ADAPTIVE INTRUSION DETECTION USING HYBRID DEEP LEARNING MODELS AND FEATURES SELECTION

A Project Report submitted in the partial fulfillment of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

TUMMA DURGA BHAVANI (22471A05O6)
GUGGILLA KAVYA (22471A05M4)
SHAIK YALAVARTHI KHAJABI (22471A05O3)
ANDRAJU DEEPTHI PRIYA (23475A0513)

Under the esteemed guidance of

Dr.K.Suresh Babu, B.Tech., M.Tech, Ph.D..

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under
Tier -1 an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE,
NARASARAOPET- 522601

2025-2026

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "ADAPTIVE INTRUSION DETECTION USING HYBRID DEEP LEARNING MODELS AND FEATURES SELECTION" is a bonafide work done by Tumma Durga Bhavani (22471A05O6), Guggilla Kavya (22471A05M4),Shaik.Yalavarthi.Khajabi(22471A05O3),Andraju.Deepthi Priya (23475A0513) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2025-2026.

PROJECT GUIDE
Dr.K.Suresh Babu, B.Tech, M.Tech,Ph.D.
Associate Professor

PROJECT CO-ORDINATOR
Sd.Rizwana, B.Tech, M.Tech, (Ph.D).
Assistant Professor

HEAD OFTHE DEPARTMENT
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled " ADAPTIVE INTRUSION DETECTION USING HYBRID DEEP LEARNING MODELS AND FEATURES SELECTION" is composed by me that the work contain here is my own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified.

Tumma Durga Bhavani (22471A05O6)
Guggilla kavya (22471A05M4)
Shaik Yalavarthi Khajabi (22471A05O3)
Andraju Deepthi Priya (23475A0513)

ACKNOWLEDGEMENT

We wish to express my thanks to various personalities who are responsible for the completion of my project. I am extremely thankful to my beloved chairman**, Sri M. V. Koteswara Rao, B.Sc.,** who took keen interest in me in every effort throughout this course. I owe my sincere gratitude to my beloved principal**, Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express my deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of the CSE department, and also to my guide**, Dr.K.Suresh Babu, B.Tech., M.Tech., Ph.D.,** Associate Professor of the CSE department, whose valuable guidance and unstinting encouragement enabled me to accomplish my project successfully in time.

We extend my sincere thanks to **Sd.Rizwana, B.Tech., M.Tech., (Ph.D.),** Assistant Professor & Project Coordinator of the project, for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for me throughout this project work.

We extend my sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during my B.Tech. degree.

We have no words to acknowledge the warm affection, constant inspiration, and encouragement that I received from my parents.

We affectionately acknowledge the encouragement received from my friends and those who were involved in giving valuable suggestions and clarifying my doubts, which really helped me in successfully completing my project.

By

Tumma Durga Bhavani (22471A05O6)
Guggilla Kavya (22471A05M4)
Shaik Yalavarthi Khajabi (22471A05O3)
Andraju Deepthi Priya (22475A0513)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OFCOMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA
NEC ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to the academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8)

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7)

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decisionmaking and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii)critical thinking in the broadest context of technologicalchange.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.
CO421.2: Identify and classify the requirements.
CO421.3: Review the Related Literature.
CO421.4: Design and Modularize the project.
CO421.5: Construct, Integrate, Test and Implement the Project.
CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1		✓										✓		
C421.2	✓		✓		✓							✓		
C421.3				✓		✓	✓	✓				✓		
C421.4			✓			✓	✓	✓				✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓	✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1	2	3										2		
C421.2			2		3							2		
C421.3				2		2	3	3				2		
C421.4			2			1	1	2				3	2	
C421.5					3	3	3	2	3	2	2	3	2	1
C421.6									3	2	1	2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop model for detection and classification of OSCC.	PO1, PO3,PO8
CC421.1, C2204.3, C22L3.2	Each and every requirement i critically analyzed, the process mode is identified.	PO2, PO3, PO8
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5, PO8
CC421.4, C2204.4, C22L3.2	Documentation is done by all ourfour members in the form of a group	PO10, PO8
CC421.5, C2204.2, C22L3.3	Each and every phase ofthe work in group is presented periodically	PO8,PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection for Oral Cancer	PO4, PO7, PO8
C32SC4.3	The physical design includes website to check OSCC	PO5, PO6, PO8

ABSTRACT

In recent years, Deep Learning has emerged as a powerful approach in cybersecurity, particularly for detecting and mitigating complex and evolving cyberattacks. The present work illustrates an adaptive Intrusion Detection System (IDS) that integrates hybrid deep learning models with feature selection and class-balancing techniques to improve accuracy and robustness. The proposed system combines the strengths of multiple deep neural architectures Autoencoder, Gated Recurrent Unit (GRU), AlexNet, and MiniVGGNet to effectively learn spatial and temporal patterns in network traffic data. Initially, the CICIDS2017 benchmark dataset undergoes preprocessing techniques including data cleaning, Min-Max normalization, and SMOTEENN for class balancing, followed by feature selection using the Boruta algorithm to retain the most relevant attributes. Autoencoder and GRU models capture anomalies and temporal dependencies, while CNN- based models (AlexNet and MiniVGGNet) extract deep spatial features that enhance classification accuracy. The hybrid IDS achieved an impressive accuracy of 99.67%**,** precision of 98%**,** recall of 98.08%**,** and F1-score of 98.09%**,** outperforming conventional machine learning approaches. Therefore, this hybrid deep learning model demonstrates its capability in adaptive intrusion detection, offering a robust, automated, and scalable solution for identifying both known and unknown network threats in real time.

INDEX

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

In today's digital era, cybersecurity has become one of the most critical aspects of. protecting data, networks, and systems from an ever-growing range of cyber threats. With the exponential growth of the internet, cloud computing, and IoT devices, the risk of unauthorized access, data theft, and malicious network activities has increased dramatically. Intrusion Detection Systems (IDS) play a vital role in identifying abnormal behaviors in network traffic, ensuring the confidentiality, integrity, and availability of data. However, traditional IDS approaches relying on signature-based or rule-based mechanisms often fail to detect new, unknown, or zero-day attacks, thereby emphasizing the need for adaptive, intelligent, and automated detection mechanisms

The complexity and diversity of network attacks, including Distributed Denial of Service (DDoS), brute-force, botnet, and infiltration attacks, make accurate intrusion detection a challenging task. These attacks often mimic normal traffic behavior, making it difficult for static systems to distinguish between legitimate and malicious activities. To address this, researchers have increasingly turned to Deep Learning (DL) and Machine Learning (ML) techniques, which can automatically learn complex representations and patterns from large volumes of data.

The CICIDS2017 dataset**,** developed by the Canadian Institute for Cybersecurity, provides a realistic benchmark for network intrusion detection research. It contains diverse attack categories and realistic traffic patterns, making it suitable for evaluating deep learning-based IDS models. However, one of the major challenges associated with such datasets is class imbalance where benign traffic significantly outnumbers malicious samples resulting in biased model learning and poor generalization.

To overcome these challenges, this research proposes an adaptive Intrusion Detection System using hybrid deep learning models integrated with advanced preprocessing and feature selection techniques. The proposed model combines Autoencoder**,** Gated Recurrent Unit (GRU)**,** AlexNet, and MiniVGGNet architectures to capture spatial, temporal, and anomalous features of network traffic. To improve training efficiency and handle class imbalance, SMOTEENN (Synthetic Minority Oversampling Technique Edited Nearest Neighbor) is applied to balance the dataset, while Boruta feature selection is used to retain only the most relevant and informative attributes, reducing computational complexity and enhancing accuracy.

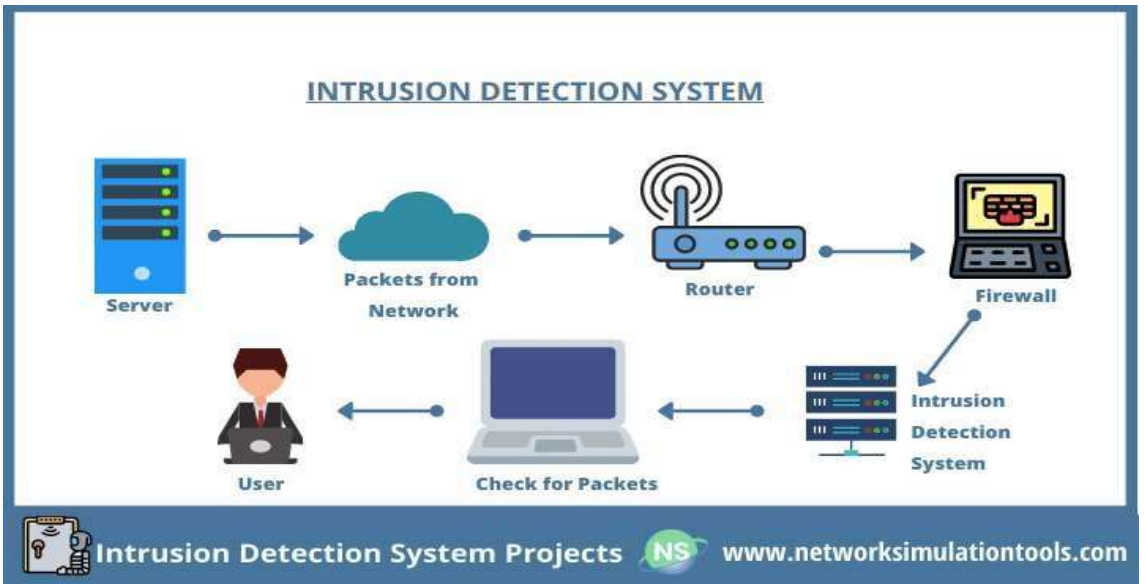


FIG 1. 1 ARCHITECTURE OF A TYPICAL INTRUSION DETECTION SYSTEM (IDS)

Each deep learning model contributes unique strengths to the proposed Intrusion Detection System. The Autoencoder effectively detects anomalies by learning compressed representations of normal traffic, while the Gated Recurrent Unit (GRU) captures temporal dependencies in sequential data, which is vital for identifying time based attacks. Meanwhile, AlexNet and MiniVGGNet extract rich spatial features from traffic matrices, enhancing the overall classification precision of the system.

Extensive experimentation demonstrates that this hybrid model achieves superior detection performance with an accuracy of 99.67%**,** precision of 98%**,** recall of 98.08%**,** and F1-score of 98.09% on the CICIDS2017 dataset. These results confirm the model's robustness and adaptability in identifying both known and novel intrusions.

With its intelligent combination of preprocessing, feature selection, and hybrid deep learning techniques, the proposed system offers a powerful and scalable framework for modern network intrusion detection**.** It not only enhances accuracy and reduces false alarms but also paves the way for integrating AI-based systems into real-world cybersecurity environments. As cyberattacks continue to evolve, such adaptive IDS models will become indispensable for ensuring network resilience and security in the future.

1.1 MOTIVATION

With the rapid expansion of internet-based applications and the ever-increasing interconnectivity of devices, network infrastructures have become prime targets for malicious cyber activities. Attacks such as Denial of Service (DoS), Distributed Denial of Service (DDoS), brute-force, botnet, and injection exploits can disrupt essential services, compromise sensitive data, and cause severe financial and reputational losses. Traditional Intrusion Detection Systems (IDS) that rely on rule-based or signature-based mechanisms often fail to detect new and evolving threats, making adaptive and intelligent detection approaches an urgent necessity.

Recent advances in deep learning have demonstrated superior capabilities in identifying complex and hidden attack patterns within massive network traffic data. However, challenges such as class imbalance, redundant features, and high computational cost continue to limit IDS performance. This motivated the exploration of hybrid deep learning frameworks that can efficiently process diverse traffic data, extract meaningful features, and adapt to emerging attack types.

The proposed work aims to design an adaptive IDS that integrates SMOTEENN for class balancing, Boruta for feature selection, and multiple deep learning models such as Autoencoder**,** GRU**,** AlexNet**,** and MiniVGGNet. This combination enhances both the accuracy and robustness of the detection system, enabling it to identify novel and dynamic network intrusions effectively. The motivation lies in developing a scalable and intelligent IDS that not only minimizes false alarms but also strengthens cybersecurity resilience across modern network environments.

1.2 PROBLEM STATEMENT

In today's digital era, the frequency and sophistication of cyberattacks are rapidly increasing, posing serious challenges to network security. Traditional Intrusion Detection Systems (IDS) that rely on rule-based or signature-based techniques are limited to recognizing only known attack patterns and often fail to detect new or evolving threats. Furthermore, the high dimensionality, redundant attributes, and severe class imbalance present in network datasets degrade the detection accuracy and increase false alarm rates.

Existing machine learning and deep learning models, though effective in pattern recognition, struggle to maintain performance when faced with imbalanced and noisy data. Hence, there is a pressing need for an adaptive and intelligent IDS capable of handling large-scale traffic data, reducing redundancy, and efficiently detecting both known and unknown intrusions.

This research addresses these challenges by developing a hybrid deep learning-based IDS that integrates SMOTEENN for data balancing and Boruta for feature selection, along with Autoencoder**,** GRU**,** AlexNet, and MiniVGGNet models for improved accuracy and robustness. The goal is to achieve a scalable, reliable, and highly accurate detection system that minimizes false positives and enhances real-time network protection.

1.3 OBJECTIVE

The primary objective of this research is to develop an adaptive and intelligent Intrusion Detection System (IDS) capable of efficiently detecting both known and unknown network attacks using advanced deep learning techniques. This study aims to address class imbalance in the dataset by applying the SMOTEENN method, ensuring balanced learning across all attack categories. Another key goal is to enhance model efficiency and accuracy by performing feature selection using the Boruta algorithm, which helps in identifying the most significant features while removing redundant ones. The proposed work also focuses on implementing and comparing multiple deep learning architectures, including Autoencoder**,** GRU**,** AlexNet**,** and MiniVGGNet, to evaluate their performance using the CICIDS2017 dataset. Furthermore, this research seeks to design a hybrid IDS framework that combines robust preprocessing techniques with deep network models to improve accuracy, precision, recall, and F1-score**.** Ultimately, the system aims to achieve high scalability, reliability, and real-time adaptability**,** thereby enhancing overall network security and intrusion detection effectiveness.

2 .LITERATURE SURVEY

Intrusion Detection Systems (IDS) play a crucial role in ensuring network security by identifying and preventing malicious activities within network traffic. Traditional signature-based and rule-based IDS models have been effective in detecting known attacks but fail to recognize new or evolving intrusions. With the exponential growth of data and the increasing complexity of cyber threats, researchers have turned towards Machine Learning (ML) and Deep Learning (DL)-based techniques for building intelligent and adaptive IDS capable of learning complex traffic patterns automatically.
Early studies applied Machine Learning classifiers such as Support Vector Machines (SVM), Decision Trees (DT), and Random Forests (RF) for network intrusion detection. However, these algorithms depend heavily on manual feature extraction and often struggle with imbalanced datasets and high-dimensional data. To address these challenges, Deep Learning methods like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Autoencoders have been explored, offering improved feature learning and classification accuracy.

Yin et al. [1] proposed an RNN-based intrusion detection model capable of capturing temporal dependencies in network traffic, which improved detection accuracy compared to traditional ML approaches. Similarly, Kim et al. [2] developed a CNN-based IDS that efficiently extracted spatial features from raw traffic data and achieved higher accuracy than SVM and Naïve Bayes models. These studies demonstrated that deep learning architectures outperform classical methods by automatically learning meaningful representations from large datasets.

Shone et al. [3] introduced a stacked non-symmetric deep autoencoder model for unsupervised intrusion detection, showing strong performance on the NSL-KDD dataset. The model effectively learned latent traffic patterns without manual intervention, reducing the need for labeled data. However, it still suffered from class imbalance, which impacted detection rates for minority attack types. To overcome this issue, data balancing techniques like SMOTE (Synthetic Minority Oversampling Technique) and its variants were proposed to improve model performance on underrepresented classes.

Choudhary and Jain [4] employed SMOTE-Tomek Links to handle imbalanced network data, achieving better classification accuracy and reduced false alarms. Building on this, Khan et al. [5] used SMOTEENN, a hybrid of oversampling and data cleaning, which not only balanced the dataset but also removed noisy instances, resulting in improved stability and accuracy for deep learning-based IDS. These preprocessing techniques proved essential for creating reliable and generalizable intrusion detection frameworks.

Feature selection has also played a vital role in enhancing IDS performance. The Boruta algorithm, as applied by Jovic et al. [6], was used to identify significant and relevant features by comparing real attributes with randomized shadow features, effectively eliminating redundancy. This method significantly reduced computational cost while maintaining high accuracy. Similarly, Li et al. [7] combined Boruta with Random Forests to enhance model interpretability and efficiency for high-dimensional network data.

Recent advancements focus on hybrid deep learning models that combine multiple architectures to exploit their individual strengths. Zhang et al. [8] proposed a CNN-GRU hybrid model, which captured both spatial and temporal relationships in traffic flows, outperforming standalone CNN and RNN models. Another approach by Hu et al. [9] integrated ResNet and Bi-LSTM with an attention mechanism, achieving excellent detection rates but requiring heavy computation, which limited real-time applicability. To address these gaps, lightweight CNN variants like MiniVGGNet and AlexNet have been applied to optimize model size while maintaining accuracy.

Elmasry et al. [10] developed a hybrid IDS framework using Autoencoder and GRU, achieving superior performance on the CICIDS2017 dataset with improved accuracy, precision, and recall. Their research demonstrated that combining different deep learning architectures could yield robust and adaptive intrusion detection systems. Similarly, Al- Qatf et al. [11] used a Stacked Autoencoder with Softmax classifier for feature learning and classification, improving the identification of previously unseen attacks.

The analysis of existing works highlights that despite significant progress in deep learning- based IDS, challenges such as class imbalance, redundant features, and computational complexity persist. Therefore, the present research aims to design an adaptive hybrid IDS integrating SMOTEENN for class balancing and Boruta for feature selection, along with Autoencoder, GRU, AlexNet, and MiniVGGNet architectures. This combination enhances feature extraction, reduces false alarms, and provides a scalable and intelligent solution for detecting both known and unknown network intrusions effectively.

3.SYSTEM ANLYSIS

3.1 EXISTING SYSTEM

In the existing systems for network intrusion detection, various deep learning techniques such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNNs) have been widely employed to classify network traffic into normal and attack categories. These models automatically learn complex patterns and relationships from large-scale network data without the need for manual feature engineering. Studies using these methods on benchmark datasets such as NSL-KDD and CICIDS2017 have shown promising detection accuracy. However, despite their capability to extract high-level features, these models struggle to achieve balanced performance across all attack classes due to the data imbalance problem present in real-world network traffic.

In these existing deep learning-based IDS models, the training data often contains a large number of normal records and very few samples from rare attack types such as DDoS, Infiltration, or Web Attacks. This imbalance leads to biased learning, where models tend to classify most inputs as normal traffic, resulting in high accuracy but poor recall for minority attacks. Consequently, such systems fail to detect less frequent but critical intrusions, reducing their reliability in practical environments. Furthermore, the datasets used for training often include redundant and irrelevant features, which negatively affect the learning efficiency and increase computational complexity.

While CNNs capture spatial dependencies in network features and RNN-based architectures learn temporal relationships, their performance degrades when handling noisy, redundant, or highly imbalanced data. Some existing approaches attempted to improve accuracy by applying basic data sampling or threshold tuning, but these methods were insufficient to resolve the imbalance problem completely.

Therefore, the existing deep learning-based IDS frameworks, though effective in pattern recognition, still suffer from poor generalization, high false alarm rates, and low detection of minority attacks. This limitation motivated the development of an enhanced hybrid deep learning model that incorporates data balancing and feature selection techniques to achieve better accuracy and robustness**.**

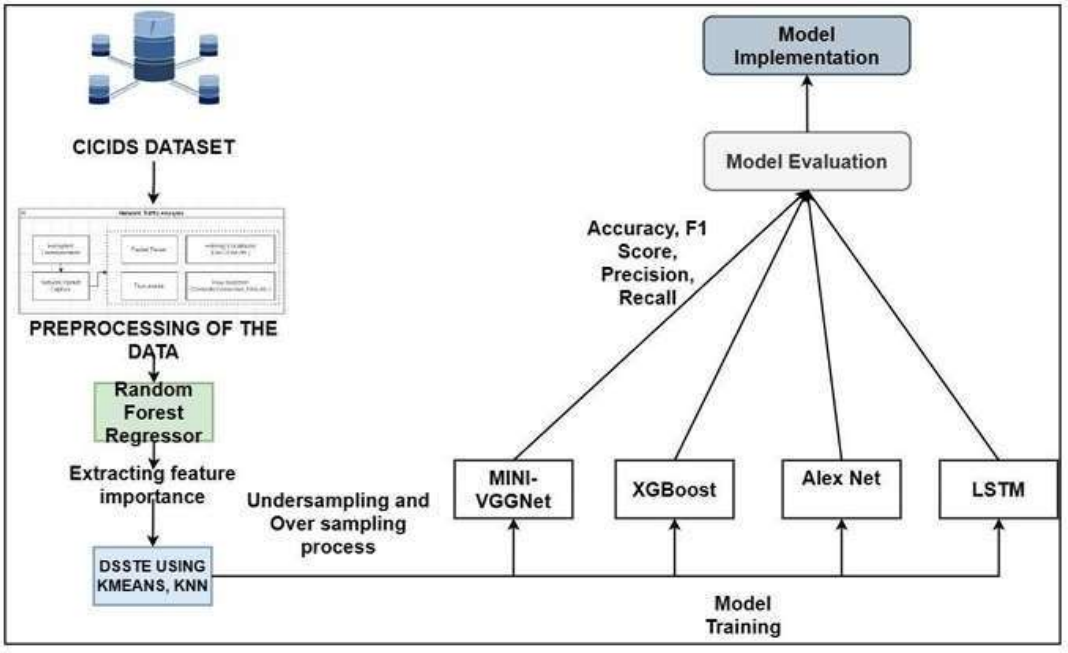


FIG 3.1. FLOW CHART OF EXISTING SYSTEM

Figure 3.1 depicts the working process of the existing deep learning-based intrusion detection system. The system begins with network traffic data collection, followed by data preprocessing, where cleaning, normalization, and feature extraction are performed to prepare the dataset for training. The processed data is then passed into deep learning models such as CNN, LSTM, or GRU, which automatically learn spatial and temporal dependencies to classify network traffic as normal or attack. Although these models show good learning capability, their performance decreases when handling noisy, redundant, or highly imbalanced data. The imbalance in datasets causes the model to become biased toward majority classes, resulting in low recall, poor detection of minority attacks, and high false negatives. Moreover, the absence of effective feature selection leads to redundant attributes and increased computational cost. Therefore, the existing system, despite its effectiveness in pattern learning, lacks adaptability and robustness in detecting unbalanced network intrusions, which motivated the development of an improved hybrid deep learning model.

3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM

The existing deep learning-based intrusion detection models demonstrate strong learning capability but still face significant challenges when applied to unbalanced network traffic data. These models become biased toward majority classes, resulting in low recall and poor detection of minority attacks such as DDoS, Infiltration, and Web Attacks. They often fail to generalize well to unseen or evolving attack patterns due to the absence of data balancing techniques. Furthermore, the lack of effective feature selection leads to redundant and irrelevant information being processed, which increases computational complexity and decreases overall efficiency. In addition, these models are prone to overfitting and produce high false alarm rates when trained on noisy or redundant data. Consequently, the existing deep learning models lack robustness, scalability, and adaptability, making them unsuitable for accurately detecting rare and evolving network intrusions in real-time environments.

Although existing deep learning-based intrusion detection models such as CNN, LSTM, and GRU have shown considerable improvement over traditional machine learning methods, they still exhibit several critical limitations when applied to unbalanced network intrusion datasets. One major drawback is their bias toward majority classes, where models achieve high overall accuracy but fail to correctly classify minority or rare attack categories like DDoS, Port Scan, and Infiltration. This imbalance causes a serious degradation in recall and F1-score, leading to false negatives that can allow malicious activities to go undetected. Furthermore, most existing models lack efficient feature selection mechanisms, causing the inclusion of redundant and irrelevant data, which results in higher computational costs, longer training times, and reduced model interpretability.

Therefore, despite their strong capability in automated feature extraction, existing deep learning-based IDS models lack robustness, adaptability, and scalability, emphasizing the need for an enhanced hybrid system that can effectively address data imbalance, feature redundancy, and detection inefficiency.

3.2 PROPOSED SYSTEM

The proposed system aims to overcome the limitations of existing deep learning-based intrusion detection models by introducing a hybrid framework that effectively handles imbalanced network traffic data and improves detection accuracy, precision, and recall. The system integrates data preprocessing, data balancing, feature selection, and hybrid deep learning model training to achieve robust and adaptive intrusion detection. The dataset used, such as CICIDS2017, is first preprocessed to remove redundant and missing values, ensuring clean and consistent input. The SMOTEENN (Synthetic Minority Oversampling Technique+ Edited Nearest Neighbors) method is applied to balance the dataset by oversampling minority classes and cleaning noisy samples.

After balancing, the Boruta feature selection algorithm is used to identify the most relevant and important features by comparing them against random shadow‖ features. This helps remove redundant attributes and reduces computational cost. The selected features are then fed into multiple deep learning models namely Autoencoder, Gated Recurrent Unit (GRU), AlexNet, and MiniVGGNet. Each model captures distinct aspects of the data: Autoencoders detect hidden representations, GRUs learn temporal dependencies, and CNN- based models (AlexNet and MiniVGGNet) capture spatial features from the network traffic. Finally, the performance of all models is evaluated using metrics such as Accuracy, Precision, Recall, and F1-score, and the best-performing model is selected as the final IDS framework. This hybrid architecture provides a robust, adaptive, and scalable intrusion detection system that can effectively detect both known and unknown attacks even in highly unbalanced datasets.

Furthermore, by combining multiple deep learning architectures such as Autoencoder, GRU, AlexNet, and MiniVGGNet, the system leverages the strengths of each model, resulting in improved feature extraction, faster convergence, and enhanced generalization for real-time intrusion detection. The integration of data balancing (SMOTEENN) and feature selection (Boruta) ensures that the model maintains high accuracy while minimizing false alarms and overfitting.

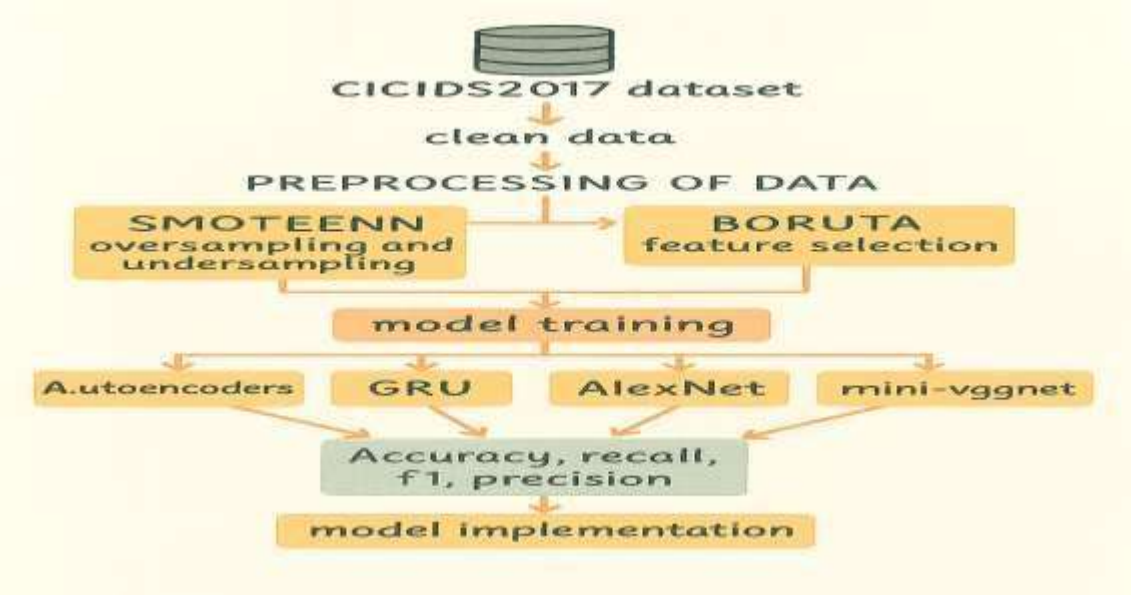


FIG 3.2. FLOW CHART OF PROPOSED SYSTEM

Figure 3.2 represents the workflow of the proposed hybrid deep learning-based intrusion detection system. The process starts with network traffic data collection from benchmark datasets like CICIDS2017. The data undergoes preprocessing, including cleaning, normalization, and transformation, to ensure quality and consistency. To address the issue of data imbalance, the SMOTEENN technique is applied, which combines oversampling and noise removal to balance minority and majority classes effectively. After balancing, Boruta feature selection is performed to extract the most relevant features while eliminating redundant ones, thereby improving model efficiency and accuracy.

The selected features are then passed through multiple deep learning models, including Autoencoder, GRU, AlexNet, and MiniVGGNet, each contributing unique strengths Autoencoders extract hidden representations, GRUs capture temporal dependencies, and CNN based models identify spatial correlations in traffic data.

Finally, the outputs are analyzed to classify each record as normal or attack based on learned patterns. This hybrid approach enhances detection accuracy, reduces false alarms, and ensures robust performance even with unbalanced and dynamic network traffic, providing an intelligent and scalable solution for modern cybersecurity challenges.

ADVANTAGES OVER PROPOSED MODEL:

The proposed hybrid deep learning-based intrusion detection system offers the following advantages:

- **Effective Handling of Imbalanced Data:**

The integration of SMOTEENN efficiently balances minority and majority classes, improving the detection of rare and unseen attack types.

- **Improved Feature Selection:**

The Boruta algorithm removes redundant and irrelevant features, reducing dataset complexity and enhancing model performance.

- **Hybrid Deep Learning Framework:**

The combination of Autoencoder**,** GRU**,** AlexNet, and MiniVGGNet enables the system to capture both spatial and temporal patterns in network traffic data.

- **Higher Detection Accuracy:**

The system achieves improved accuracy, precision, recall, and F1-score**,** ensuring reliable intrusion detection across multiple attack categories.

- **Reduced Computational Complexity:**

Through optimized feature selection and balanced data processing, the model reduces training time and improves overall efficiency.

- **Robust and Adaptive Performance:**

The hybrid approach ensures better generalization**,** reduced overfitting, and stable performance under different network conditions.

- **Scalability and Real-Time Applicability:**

The proposed framework performs efficiently on large datasets such as CICIDS2017 and can be adapted for real-time intrusion detection systems.

- **Enhanced Detection of Unknown Attacks:**

The inclusion of Autoencoder and deep CNN models helps in identifying zero-day and previously unseen attacks with high reliability.

3.3 FEASIBILITY STUDY

The feasibility study evaluates the practicality and effectiveness of implementing the proposed hybrid deep learning-based intrusion detection system in real-world environments. It focuses on three key aspects - Technical, Economic, and Operational feasibility.

A. **Technical Feasibility**

- The system uses deep learning frameworks such as TensorFlow and Keras, which support models like Autoencoder, GRU, AlexNet, and MiniVGGNet.
- Integration of SMOTEENN and Boruta algorithms is achieved using open-source Python libraries like imbalanced-learn and BorutaPy.
- Requires only moderate hardware resources (GPU-enabled system or Google Colab) for training and testing deep learning models.
- The modular design allows easy scalability and upgradation**,** enabling future integration of new datasets or models.
- Technically stable and efficient for processing large-scale, unbalanced network traffic data.

B. Economic Feasibility

- The entire system is built using open-source tools and libraries, avoiding the need for paid software licenses.
- such as CICIDS2017 are publicly available and free to use.
- Model training and testing can be done using free platforms like Google Colab**,** reducing computational costs.
- Optimized feature selection (Boruta) and data balancing (SMOTEENN) reduce training time and resource consumption.
- Offers high performance at minimal cost, making it suitable for academic, research, and enterprise use.

C. Operational Feasibility

- The system is user-friendly and can be easily integrated into existing cybersecurity frameworks.
- Provides automated data preprocessing**,** balancing**,** and attack detection, requiring minimal human supervision.

- Capable of handling real-time traffic data and detecting zero-day attacks effectively.
- Ensures low false alarm rates and high detection accuracy across various attack types.
- Reliable**,** scalable**,** and adaptive under changing network conditions, making it suitable for real-world deployment.

3.4 USING COCOMO MODEL

COCOMO (Constructive Cost Model) is a software cost estimation model developed by Barry Boehm, used to estimate the effort, development time, and cost required for software development. Although the proposed system is primarily a research-based deep learning project, the Basic COCOMO model can be applied to estimate the approximate development effort for the software components involved in data preprocessing, model training, and evaluation.

The Basic COCOMO model is represented as:

Effort (E)=a×(KLOC)^b

Development Time (D)=c×(Effort)^d

Where:

- KLOC = Estimated number of lines of code (in thousands)
- a, b, c, d = Constants based on project type (Organic, Semi-Detached, Embedded)

Project Type: Organic Mode

Since the proposed system is research-oriented, developed by a small team using open-source tools (Python, TensorFlow, Keras), it falls under the Organic Mode category.

PARAMETER	ORGANIC MODEL VALUE
a	2.4
b	1.05
c	2.5
d	0.38

TABLE 3.4 ORGANIC MODE

Example Estimation

Let us assume the project contains approximately 6,000 lines of code (6 KLOC) across preprocessing, feature selection, and deep learning modules.

Effort=2.4×(6)^1.05=15.4 person-months Development Time=2.5×(15.4)^0.38=6.7 months

Average Staff Required=Efforts / Development Time =15.4/6.7≈2.3 persons

Thus, approximately 2 to 3 developers can complete the system in around 6–7 months of development time.

4.SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

1. Operating System : Windows 11, 64-bit Operating System
2. Hardware Accelator : CPU
3. Coding Language : Python
4. Python distribution : Google Colab Pro, Flask
5. Browser : Any Latest Browser like Chrome

These software tools provide a powerful and flexible environment for data preprocessing, model training, and performance evaluation.

4.2 REQUIREMENT ANALYSIS

The requirement analysis identifies the functional and non-functional needs of the system to ensure efficient intrusion detection.

Functional Requirements

- The system should collect and preprocess network traffic data.
- It must balance unbalanced datasets using the SMOTEENN technique.
- It should perform feature selection using the Boruta algorithm.
- The system should train and evaluate multiple deep learning models **(**Autoencoder, GRU, AlexNet, MiniVGGNet**).**
- It must classify network traffic as normal or attack based on model predictions.
- The system should display evaluation metrics such as Accuracy, Precision, Recall, and F1-score**.**

Non-Functional Requirements

- The system must be robust, adaptive, and scalable**.**
- It should ensure high accuracy and low false alarm rates**.**
- It must operate efficiently on large-scale network traffic data**.**
- The system should maintain user-friendliness and be easy to deploy.

This configuration ensures smooth training, testing, and real-time execution of the deep learning models.

4.3 HARDWARE REQUIREMENTS

1. System Type : 64 – bit operating system, x64-based processor

2. Cache Type : 4MB(Megabyte)

3. RAM : 16MB(Gigabyte)

4. Hard Disk : 8GB

5. GPU : Intel® Iris® Xe Graphics

4.4 SOFTWARE

The intrusion detection project uses a comprehensive suite of software tools, frameworks, and technologies for efficient model training, deployment, and user interaction. The system is implemented on Windows 11 (64-bit) for local execution and Google Colab for GPU- accelerated model training and experimentation.

The backend development utilizes Python, known for its flexibility and rich ecosystem of machine learning libraries. Frameworks such as TensorFlow/Keras are employed for building hybrid deep learning models, while Scikit-learn supports preprocessing, model evaluation, and performance metrics computation. The imbalanced-learn library implements SMOTEENN for dataset balancing, and BorutaPy performs optimal feature selection.

For frontend development, the project employs HTML5, CSS3, and JavaScript to create a user-friendly web interface. The interface allows users to upload data, trigger model predictions, and view the results visually. Bootstrap ensures a responsive and mobile- friendly layout, while custom CSS styling enhances usability and design consistency across devices.

Visualization libraries such as Matplotlib and Seaborn are used to display model performance results, including accuracy and confusion matrices. Data manipulation and handling are efficiently carried out using Pandas and NumPy.

The system is compatible with all modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge, ensuring smooth execution and accessibility for end users.Overall, the integration of these backend and frontend technologies makes the proposed system reliable, interactive, accurate, and adaptable to real-world cybersecurity applications.

4.5 SOFTWARE DESCRIPTION

The proposed Hybrid Deep Learning-Based Intrusion Detection System operates in a Python-based backend environment with a web-based frontend interface for seamless user interaction.

At the backend, TensorFlow and Keras are used to design and train deep learning models such as Autoencoder, GRU, AlexNet, and MiniVGGNet. These models work collaboratively to analyze network traffic and classify it as normal or attack. The SMOTEENN algorithm handles dataset imbalance by oversampling minority classes and cleaning noisy samples, while the Boruta feature selection method extracts the most important attributes, reducing computational complexity and improving model performance.

Model training and testing are carried out using the CICIDS2017 dataset, ensuring a realistic and comprehensive network intrusion detection benchmark. Development is conducted on Google Colab, which provides access to GPU acceleration for faster computation.

The frontend of the application is developed using HTML, CSS, and JavaScript, ensuring an intuitive and interactive interface. It allows users to visualize detection results, review performance metrics, and understand model predictions easily. Bootstrap and custom CSS provide a responsive design layout, making the application visually appealing and compatible with multiple devices and browsers.

Supporting libraries such as NumPy, Pandas, and Scikit-learn handle data processing and model evaluation, while Matplotlib and Seaborn are used for graphical visualization of system performance metrics such as Accuracy, Precision, Recall, and F1-score.

Thus, the combined use of backend (Python, TensorFlow, Flask) and frontend (HTML, CSS, JavaScript, Bootstrap) technologies ensures that the proposed intrusion detection system is efficient, scalable, interactive, and capable of detecting both known and unknown intrusions in real time.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The Adaptive Intrusion Detection System (AIDS) proposed in this project aims to improve the detection and classification of network intrusions using an advanced hybrid Deep Learning framework. By integrating multiple deep neural architectures such as Autoencoder, Gated Recurrent Unit (GRU), AlexNet, and MiniVGGNet, along with preprocessing methods like SMOTEENN for data balancing and Boruta for feature selection, the model achieves high accuracy and adaptability in identifying cyber threats within unbalanced network traffic. The primary objective of the system is to develop a robust, intelligent, and scalable intrusion detection framework that can detect both known and unknown network attacks in real time. The architecture combines the spatial, temporal, and latent feature learning capabilities of different deep learning models to analyze complex network traffic data effectively. Unlike traditional IDS approaches that fail on imbalanced datasets, this adaptive model dynamically adjusts to the data distribution, reducing false negatives and improving detection performance.

The system begins with network traffic data collection, primarily using the CICIDS2017 dataset, which provides realistic and comprehensive network scenarios. The Data Preprocessing Module cleans and normalizes the dataset by removing null, duplicate, or irrelevant records. To address data imbalance, the SMOTEENN technique is applied to oversample minority attack classes while removing noisy samples from the majority classes, resulting in a balanced and representative dataset.

Next, feature selection is carried out using the Boruta algorithm, which identifies the most important features based on their contribution to model performance. The refined dataset is then passed through multiple deep learning models:

- The Autoencoder learns hidden data representations and detects anomalies that may represent zero-day attacks.
- The GRU (Gated Recurrent Unit) captures temporal dependencies in sequential data, learning time-based attack patterns.

- The AlexNet and MiniVGGNet architectures extract spatial and hierarchical feature relationships from network traffic attributes.

A Decision Fusion Layer integrates the outputs of all deep models, combining their strengths to provide a final intrusion prediction. The Evaluation Module computes key performance metrics such as Accuracy, Precision, Recall, and F1-score, ensuring reliability and effectiveness.

The Flask-based backend enables smooth interaction between the deep learning models and the frontend web interface (HTML, CSS, JavaScript). The interface allows users or administrators to upload datasets, monitor detection outcomes, and visualize system performance in real time. The adaptive nature of this system allows it to retrain periodically using updated network data, ensuring continuous improvement and resilience against evolving cyber threats.

This proposed architecture serves as a powerful cybersecurity solution capable of identifying both frequent and rare attack types with exceptional accuracy, adaptability, and scalability.

5.1.1 DATASET

The dataset used for developing the Adaptive Intrusion Detection System is the CICIDS2017 dataset, a comprehensive and widely recognized benchmark dataset for network intrusion detection research. It is collected by the Canadian Institute for Cybersecurity (CIC) and contains realistic network traffic that simulates both benign activities and multiple attack types across different days of network operation.

The dataset includes various features such as packet length, flow duration, protocol type, source and destination IPs, and header flags. It provides detailed insights into several modern network attack categories, including DDoS, DoS Hulk, Port Scan, Botnet, Brute Force, Infiltration, and Web attacks. Each record is labeled as either Normal or Attack, allowing for supervised learning approaches.

Before model training, the dataset undergoes preprocessing, including data cleaning, normalization, and feature encoding. The SMOTEENN algorithm is then applied to resolve class imbalance by oversampling minority attack types and eliminating noisy samples. Following this, the Boruta feature selection technique identifies the most relevant and significant features for model input, optimizing computational efficiency and accuracy.

The CICIDS2017 dataset plays a crucial role in building a highly accurate and adaptive IDS, as it contains rich, real-world network traffic data necessary for training and testing hybAttack Classes Description

Normal**:** Legitimate network traffic without any malicious intent.
DoS/DDoS Attacks: Overload servers with massive requests to disrupt network services.
Port Scan: Scans network ports to identify potential vulnerabilities.
Brute Force Attack: Attempts to gain unauthorized access through repeated password guessing.
Web Attack: Exploits web application vulnerabilities such as SQL injection and cross-site scripting.
Infiltration Attack: Involves intrusion from internal or external sources to compromise data.
Botnet Attack**:** Uses compromised devices to perform coordinated network attacks.

Attack Classes Description

Normal**:** Legitimate network traffic without any malicious intent.
DoS/DDoS Attacks: Overload servers with massive requests to disrupt network services.
Port Scan**:** Scans network ports to identify potential vulnerabilities.
Brute Force Attack**:** Attempts to gain unauthorized access through repeated password guessing.
Web Attack: Exploits web application vulnerabilities such as SQL injection and cross-site scripting.
Infiltration Attack: Involves intrusion from internal or external sources to compromise data

Feature	Description
Dataset Name	CICIDS2017
Source	Canadian Institute for Cybersecurity (CIC), University of New Brunswick
Total Records	2,830,743
Number of Features (Before Selection)	80
Number of Features (After Boruta Selection)	30–35 Key Features
Data Type	Flow-based CSV data
Class Labels	Normal, DDoS, DoS Hulk, Port Scan, Brute Force,Web Attack, Botnet
Class Distribution (Approx.)	Normal: 2,273,097 Attacks: 557,646
Balancing Method	SMOTEENN
Feature Selection	Boruta Feature Selection Algorithm
Train-Test Split	80% Training – 20% Testing
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score
File Format	CSV
Applications	Adaptive Intrusion Detection and Classification in Network Security

TABLE 5.1.1 DATASET DESCRIPTION

Destinati	Flow Dur	Total Fwd	Total Bac	Total Leng	Total Leng	Fwd Pack	Fwd Pack	Fwd Pack	Fwd Pack	Bwd Packe	Bwd Pack	Bwd Pack	Bwd Pack	Flow Byte
54865	3	2	0	12	0	6	6	6	0	0	0	0	0	4000000
55054	109	1	1	6	6	6	6	6	0	6	6	6	0	110091.7
55055	52	1	1	6	6	6	6	6	0	6	6	6	0	230769.2
46236	34	1	1	6	6	6	6	6	0	6	6	6	0	352941.2
54863	3	2	0	12	0	6	6	6	0	0	0	0	0	4000000
54871	1022	2	0	12	0	6	6	6	0	0	0	0	0	11741.68
54925	4	2	0	12	0	6	6	6	0	0	0	0	0	3000000
54925	42	1	1	6	6	6	6	6	0	6	6	6	0	285714.3
9282	4	2	0	12	0	6	6	6	0	0	0	0	0	3000000
55153	4	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	9250000
55143	3	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	12300000
55144	1	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	37000000
55145	4	2	0	37	0	31	6	18.5	17.67767	0	0	0	0	9250000
55254	3	3	0	43	0	31	6	14.33333	14.43376	0	0	0	0	14300000
36206	54	1	1	0	0	0	0	0	0	0	0	0	0	0
53524	1	2	0	0	0	0	0	0	0	0	0	0	0	0
53524	154	1	1	0	0	0	0	0	0	0	0	0	0	0
53526	1	2	0	0	0	0	0	0	0	0	0	0	0	0
53526	118	1	1	0	0	0	0	0	0	0	0	0	0	0

FIG 5.1.1 CICIDS2017 DATASET

Image Characteristics

- Realistic Network Traffic:

The CICIDS2017 dataset captures real-world network behavior, including both normal and multiple types of attack traffic, collected over several days to simulate genuine organizational network conditions.

- Comprehensive Feature Set:

It contains 80 flow-based features such as packet length, flow duration, and inter-arrival time, providing detailed statistical insights for accurately identifying and classifying network intrusions.

Applications

- Used for training and evaluating Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS).
- Serves as a benchmark dataset for deep learning-based cybersecurity research.
- Helps analyze attack behavior and traffic anomalies in real-world-like conditions.

5.1.2 DATA PRE-PROCESSING

Before training the deep learning models, the raw network traffic data must undergo a series of preprocessing steps to convert it into a clean, balanced, and feature-optimized format suitable for analysis. In network intrusion detection, raw data often contains missing values, duplicate records, and high dimensionality, making it unsuitable for direct input to machine learning or deep learning algorithms.

The data preprocessing phase ensures that the dataset becomes structured, normalized, and relevant, ultimately improving the accuracy and stability of the deep learning models. For this project, the CICIDS2017 dataset is used, and several essential preprocessing techniques are applied to enhance data quality and model performance. These methods are as follows:

1. Data Cleaning and Normalization

Raw network traffic contains redundant, missing, or inconsistent entries that can negatively impact model training. Data cleaning removes these inconsistencies, while normalization scales the feature values within a specific range (usually 0 to 1). This ensures uniformity and prevents bias in model learning. The normalization step is especially important for neural networks like Autoencoder, GRU, and CNN-based architectures that are sensitive to feature scaling.

2. Handling Class Imbalance using SMOTEENN

Network intrusion datasets such as CICIDS2017 are often highly imbalanced, meaning some attack types (like DoS or DDoS) have many samples while others (like Infiltration or Web Attack) have very few. To address this, the SMOTEENN (Synthetic Minority Oversampling Technique – Edited Nearest Neighbors) method is applied.

- SMOTE generates synthetic samples for minority attack classes to balance the dataset.
- ENN removes noisy or misclassified samples from the majority classes, improving the dataset'sclarity.

Together, SMOTEENN ensures a balanced and noise-free dataset, which allows the deep learning models to learn equally from all attack types.

3. Feature Selection using Boruta Algorithm

The original CICIDS2017 dataset contains around 80 features, many of which may be redundant or irrelevant. To enhance computational efficiency and improve classification accuracy, the Boruta algorithm is applied for feature selection. Boruta identifies the most significant and relevant features by iteratively comparing them with randomized ―shadow features.‖ Only features that consistently contribute to model accuracy are retained, reducing dimensionality while preserving essential network behavior patterns.

4.Data Splitting and Formatting

After balancing and feature selection, the dataset is divided into training (80%) and testing (20%) subsets. This ensures fair model evaluation and prevents overfitting. The data is then formatted into tensors and arrays compatible with the deep learning frameworks TensorFlow and Keras for model input**.**

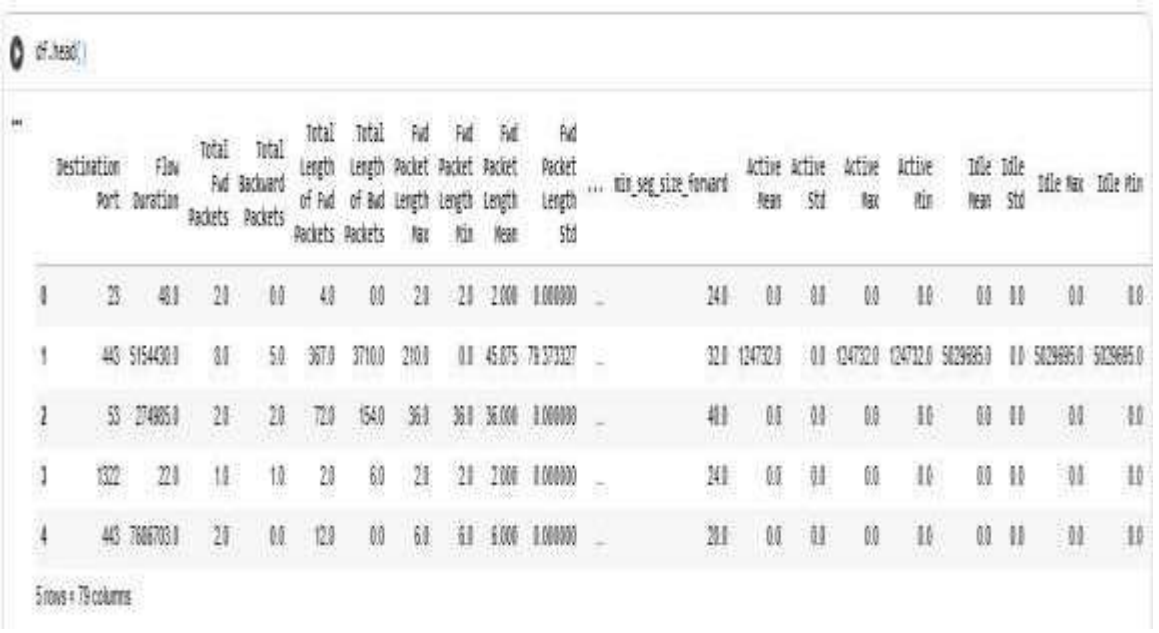
5.Data Transformation and Encoding

Categorical features such as protocol type or service flags are converted into numerical form using Label Encoding or One-Hot Encoding to ensure compatibility with neural network models. All continuous variables are transformed to normalized float values, ensuring uniform contribution during the training process.

	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	...	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min
0	23	48.0	2.0	0.0	4.0	0.0	2.0	2.0	2.000	0.000000	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	443	5164430.0	8.0	5.0	367.0	3710.0	210.0	0.0	45.875	79.373327	...	32.0	124732.0	0.0	124732.0	124732.0	5029685.0	0.0	5029685.0	5029685.0
2	53	274886.0	2.0	2.0	72.0	164.0	36.0	36.0	36.000	0.000000	...	40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1322	22.0	1.0	1.0	2.0	6.0	2.0	2.0	2.000	0.000000	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	443	7866703.0	2.0	0.0	12.0	0.0	6.0	6.0	6.000	0.000000	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 79 columns

FIG 5.1.2.1 BEFORE DATA PREPROCESSING



df.head()

	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	...	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min
0	25	48.0	2.0	0.0	4.0	0.0	2.0	2.0	2.000	0.000000	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	445	5154438.0	8.0	5.0	367.0	3710.0	210.0	0.0	45.875	79.373327	...	32.0	124732.0	0.0	124732.0	124732.0	5029695.0	0.0	5029695.0	5029695.0
2	53	274805.0	2.0	2.0	72.0	154.0	36.0	36.0	36.000	0.000000	...	40.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1322	22.0	1.0	1.0	2.0	6.0	2.0	2.0	2.000	0.000000	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	445	7686703.0	2.0	0.0	12.0	0.0	6.0	6.0	6.000	0.000000	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 79 columns

FIG 5.1.2.2 AFTER DATA PREPROCESSING

5.1.3 FEATURE EXTRACTION

Feature extraction in network intrusion detection plays a crucial role in transforming raw network traffic data into a structured set of informative attributes that effectively represent network behavior. In the context of the Adaptive Intrusion Detection System (AIDS), feature extraction enables the deep learning models to identify complex patterns, correlations, and anomalies within the data that are often invisible to traditional statistical methods.

Unlike image-based analysis methods such as GLCM, the proposed system focuses on network-level feature learning using deep learning-based feature extractors such as Autoencoder, Gated Recurrent Unit (GRU), AlexNet, and MiniVGGNet. These architectures are designed to automatically learn hierarchical, temporal, and spatial representations from the preprocessed network flow data, eliminating the need for manual feature engineering.

The Autoencoder serves as an unsupervised feature extractor that compresses high-dimensional network data into a lower-dimensional latent space, capturing the most relevant patterns while discarding redundant information. This process allows the model to recognize deviations or anomalies that could represent potential intrusions or zero-day attacks.

The GRU (Gated Recurrent Unit) captures temporal dependencies within sequential network flows, effectively learning how network traffic evolves over time. By preserving contextual relationships between packets, GRU models help identify time-based attack patterns such as DDoS floods or Brute Force attempts.

On the other hand, AlexNet and MiniVGGNet, originally developed for image classification, are adapted to handle numerical network data by treating flow-based features as spatial patterns. These convolutional neural networks (CNNs) are capable of learning local correlations among network attributes, enabling robust detection of subtle attack signatures embedded within normal traffic.

Through this multi-model feature extraction process, the system learns deep, abstract representations of network behavior that significantly enhance detection accuracy and reduce false alarms. The extracted feature maps are then fed into a Decision Fusion Layer, where outputs from all models are combined to produce the final classification distinguishing between Normal and various Attack categories.

Feature extraction, therefore, forms the backbone of the proposed adaptive intrusion detection framework, enabling it to capture complex network interactions, adapt to dynamic traffic patterns, and effectively detect both known and unknown cyber threats in real-time environments.

5.1.4 MODEL BUILDING :

The Adaptive Hybrid Deep Learning Model is constructed as a multi-stream architecture that integrates multiple deep learning networks Autoencoder, GRU, AlexNet, and MiniVGGNet to efficiently detect both known and unknown network intrusions. The model is designed to learn and analyze spatial, temporal, and latent representations of network traffic, providing a comprehensive understanding of malicious patterns even within highly unbalanced datasets.

1. Model Architecture Design

The proposed hybrid model consists of four interconnected modules, each responsible for capturing different feature aspects of the network data:

- **Autoencoder Module:**

Acts as an unsupervised feature extractor that compresses high-dimensional input data into a latent representation, removing noise and redundancy. It helps in identifying abnormal network behaviors by learning data reconstruction patterns.

- **GRU Module (Gated Recurrent Unit):**

Specializes in modeling temporal dependencies within sequential traffic flows. By learning how network packets evolve over time, GRU enhances detection of time-dependent attacks such as DDoS, Port Scan, and Brute Force.

- **AlexNet Module:**

Extracts spatial correlations among traffic features using convolutional layers. It identifies complex local feature relationships that distinguish normal traffic from attack flows.

- **MiniVGGNet Module:**

A lightweight CNN architecture that refines feature extraction by using smaller convolutional filters, reducing computational cost while maintaining high classification precision.

2. Integration and Classification Layers

After feature fusion, the combined representation is passed into a fully connected classification network**,** responsible for final decision-making. This component includes:

- **DenseLayers:**

Multiple fully connected layers that refine the fused feature vector and enhance class separability.

- **Activation Function:**

The ReLU (Rectified Linear Unit) activation function introduces non-linearity, enabling the model to capture complex, non-linear attack behaviors.

- **Dropout Layers:**

Applied between dense layers to prevent overfitting by randomly deactivating neurons during training, improving model generalization.

- **Output Layer:**

The final layer employs a Softmax activation function to output class probabilities for multiple attack categories such as Normal, DoS, DDoS, Port Scan, Botnet, Brute Force, Infiltration, and Web Attack.

3. Model Training Strategy

The training process focuses on minimizing classification errors and ensuring stability across deep learning modules. The following strategies are applied:

- Loss Function: *Categorical Cross-Entropy* measures the difference between predicted and true class probabilities.
- Optimizer: *Adam Optimizer* used for efficient and adaptive learning rate updates during training.
- Batch Training**:** Data is processed in mini-batches (typically size 32–64) to stabilize gradient updates and improve computational efficiency.
- Fine-Tuning: Each model (Autoencoder, GRU, CNNs) is fine-tuned on the CICIDS2017 dataset**,** adapting its learned representations to specific network attack patterns.
- Early Stopping & Learning Rate Scheduling: Implemented to prevent overfitting and ensure smooth convergence during training.

4.Evaluation Metrics

The hybrid model's performance is assessed using standard classification metrics that evaluate both majority and minority attack detection:

- Accuracy: Measures the overall correctness of predictions across all classes.

- Precision & Recall: Assess the model's ability to minimize false positives and false negatives, ensuring reliable intrusion detection.
- F1-Score: Provides a harmonic balance between precision and recall, highlighting model stability.

5.Outcome of Model Building

The resulting Adaptive Hybrid Intrusion Detection Model is a robust, scalable, and intelligent framework capable of detecting both frequent and rare network attacks with high precision. By leveraging the dimensional compression of Autoencoders, temporal sequence learning of GRU, and spatial feature extraction of CNNs (AlexNet and MiniVGGNet), the model achieves exceptional accuracy even under highly unbalanced network conditions.

5.1.5 CLASSIFICATION

The classification phase is the final and most critical component of the proposed Adaptive Intrusion Detection System (AIDS). Its main objective is to accurately classify incoming network traffic into normal or attack categories based on the deep features extracted by the hybrid model. This stage integrates the outcomes of Autoencoder, GRU, AlexNet, and MiniVGGNet models to produce an optimized and reliable decision for each network flow record.

After the preprocessing, balancing (SMOTEENN), and feature selection (Boruta) steps, the refined dataset is passed into multiple deep learning architectures. Each model learns distinct feature representations:

- Autoencoder learns latent patterns and anomalies in network traffic.
- GRU captures sequential and temporal dependencies.
- AlexNet and MiniVGGNet extract hierarchical and spatial feature patterns.

The outputs from all these models are then fused in a joint feature representation layer**,** ensuring that both spatial and temporal dependencies contribute to the final decision. This fusion produces a deep, multi-dimensional representation that forms the input for the classification network**.**

1. Classification Layer Design

The fused features are passed into a fully connected dense neural network responsible for the final classification. The Softmax activation function is used in the output layer to transform the network output into class probabilities. Each probability corresponds to the likelihood that a given traffic record belongs to a specific class.

The Softmax function is mathematically expressed as:

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Where:

- P(yi) represents the probability of the record belonging to class *i*,
- Z(i) denotes the model output score for class *i*,
- n is the total number of classes.

The class with the highest probability is selected as the predicted output. This ensures multi-class discrimination between different attack categories.

2.Output Classes

The proposed adaptive system classifies network traffic into the following categories based on the CICIDS2017 dataset:

Class Label	**Description**
Normal	Legitimate traffic representing non-malicious activities.
DoS/DDoS	Attacks that overwhelm network resources, disrupting service availability.
Port Scan	Scans performed to identify open or vulnerable network ports.
Brute Force	Attempts to gain unauthorized access through repeated password guessing.
Web Attack	Exploitation of vulnerabilities in web applications (e.g., SQL Injection, XSS).

Class Label	Description
Infiltration	Attacks involving unauthorized access to internal network resources.
Botnet	Coordinated network attacks carried out by compromised systems.

These labels represent both majority and minority attack classes**,** ensuring comprehensive detection across diverse intrusion types.

3.Model Optimization and Output Interpretation

To enhance classification accuracy, the model uses:

- Loss Function: Categorical Cross-Entropy
- Optimizer: Adam Optimizer for adaptive learning rate adjustment
- Activation Function: ReLU for hidden layers and Softmax for output
- Batch Size: 32–64
- Evaluation Metrics: Accuracy, Precision, Recall, and F1-score

The Softmax layer outputs a probability vector, where each element represents the likelihood of the instance belonging to a specific attack type. The model classifies the traffic as the class with the highest probability**,** ensuring precise identification of both common and rare attack patterns**.**

4.Significance of Classification

This classification strategy enables the system to:

- Detect both known and unknown intrusions with high accuracy.
- Distinguish between benign traffic and multiple attack categories effectively.
- Maintain robustness and adaptability even with imbalanced network datasets**.**
- Minimize false positives and false negatives, ensuring reliable decision-making in real-time network environments.

5.2 MODULES

The proposed Adaptive Intrusion Detection System (AIDS) is divided into several interconnected modules that together perform data processing, feature extraction, model training, and intrusion classification.

Each module plays a crucial role in improving the system's overall detection accuracy, adaptability, and robustness. The system integrates SMOTEENN for data balancing, Boruta for feature selection, and Hybrid Deep Learning models (Autoencoder, GRU, AlexNet, MiniVGGNet) for intelligent attack detection.

Data Collection Module

This module is responsible for acquiring and organizing the CICIDS2017 dataset, which contains both normal and attack traffic.The dataset includes real-world network behaviors like DoS, DDoS, Port Scan, Infiltration, Botnet, and Web Attacks.

- Input**:** Raw CSV files containing network flow records.
- Process**:** Load and merge daily traffic data into a unified dataset.
- Output**:** Structured data ready for preprocessing.

Key Functions:

- Import network traffic data from CICIDS2017.
- Handle missing values and label normal/attack records.
- Ensure balanced class representation for training and testing.

Data Preprocessing Module

Raw network data often contains noise, missing values, and skewed distributions.

This module cleans and normalizes the data to make it suitable for training deep learning models.

Key Processes:

- Data Cleaning: Removes null, redundant, or irrelevant entries.
- Encoding: Converts categorical attributes (like protocol type) into numerical format.
- Normalization: Scales all features to a uniform range (0–1) for model efficiency.

- Data Balancing: Uses SMOTEENN (Synthetic Minority Over-Sampling + Edited Nearest Neighbor) to oversample minority classes and remove noise.
- Output: A clean, normalized, and balanced dataset that improves detection accuracy for minority attacks.

Feature Selection Module (Boruta Algorithm)

Feature selection enhances the performance and reduces computation time.

The Boruta algorithm is applied to extract only the most significant network attributes contributing to intrusion detection.

Key Processes:

- Ranks all features based on their importance using a Random Forest approach.
- Selects Strong and Confirmed features while eliminating irrelevant ones.
- Reduces dimensionality without losing key information.

Benefits:

- Improves model interpretability.
- Reduces overfitting and training time.
- Increases classification precision and recall.

Hybrid Deep Learning Model Module

This is the core module of the system that performs intelligent intrusion detection using a hybrid deep learning architecture integrating:

- Autoencoder: For unsupervised feature learning and dimensionality reduction.
- GRU (Gated Recurrent Unit): For capturing sequential temporal dependencies in network traffic.
- AlexNet and MiniVGGNet: For extracting spatial patterns and high-level abstractions from feature maps.

Working Process:

- The selected features are fed into the Autoencoder for feature compression.
- Encoded outputs are then passed to GRU and CNN-based models (AlexNet, MiniVGGNet).
- Final classification is done by combining outputs through a dense layer.

Advantages:

- Learns spatial-temporal relationships effectively.

- Provides high adaptability and resilience against unseen attacks.
- Achieves superior performance over traditional ML models.

Model Training and Testing Module

This module handles model training, validation, and testing using the preprocessed and feature-optimized dataset.

Key Steps:

- Split data into training (80%) and testing (20%) sets.
- Train hybrid deep models using Adam optimizer and categorical cross-entropy loss**.**
- Evaluate models on unseen test data for generalization.
- Fine-tune hyperparameters (batch size, learning rate, epochs) to optimize results.

Evaluation Metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- False Alarm Rate (FAR)
- AUC-ROC Score

Intrusion Classification and Visualization Module

This final module is responsible for classifying incoming traffic and visualizing the model performance.

Functions:

- Predicts whether the input network flow is Normal or Attack **(**e.g., DDoS, Botnet, Port Scan).
- Displays classification results through graphs and metrics.
- Visualizes Confusion Matrix**,** ROC Curve, and Training/Validation Performance Graphs.
- Uses t-SNE visualization to show feature separation between attack and normal classes.

Outcome:

- Provides interpretable, accurate, and real-time results for network administrators.
- Ensures adaptability for large-scale, evolving intrusion scenarios.

5.3 UML DIAGRAM

The UML (Unified Modeling Language) Use Case Diagram provides a high-level representation of the interactions between users, system components, and external entities in the proposed Adaptive Intrusion Detection System (AIDS). It visually illustrates how the user interacts with the system and how internal modules including preprocessing, feature selection, and hybrid deep learning models collaborate to achieve accurate intrusion classification.

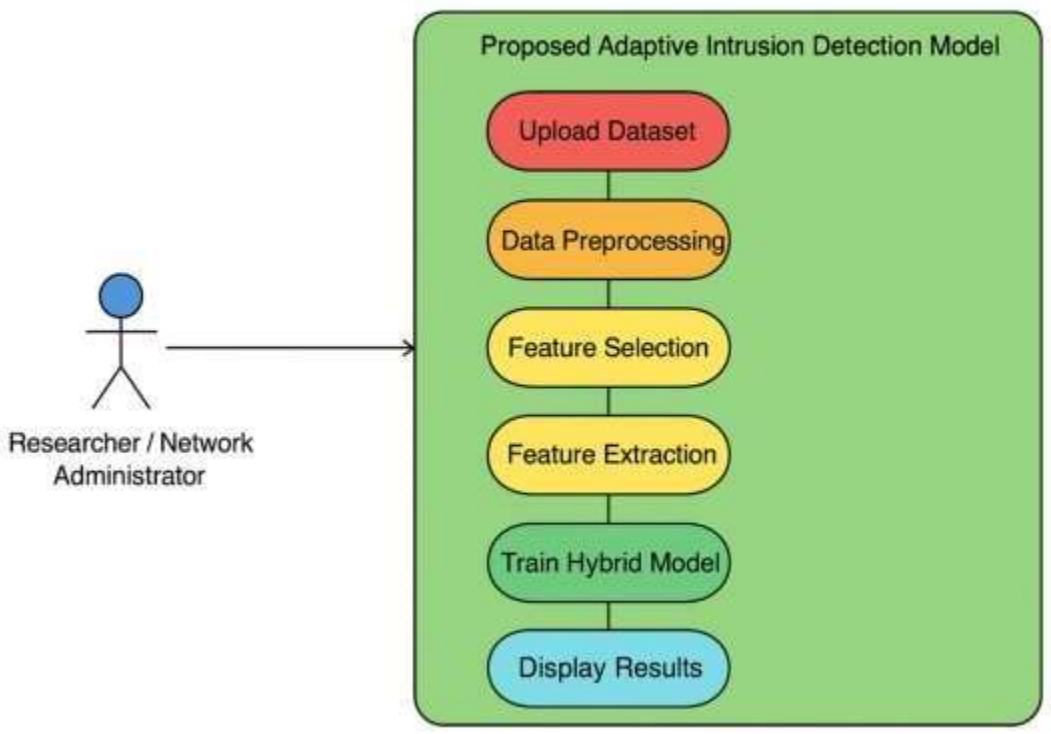


FIG 5.3 OVERVIEW OF UML DIAGRAM OF PROPOSED SYSTEM

The UML Use Case Diagram of the Proposed Adaptive Intrusion Detection Model illustrates the interaction between the Researcher/Network Administrator and the system. It shows how the user uploads the CICIDS2017 dataset, after which the system performs data preprocessing (cleaning, normalization, SMOTEENN balancing), feature selection using Boruta, and feature extraction through deep learning models like Autoencoder, GRU, AlexNet, and MiniVGGNet. The processed data is then used to train the hybrid deep learning model, which classifies network traffic into normal or various attack types such as DoS, DDoS, and Botnet. Finally, the system displays the classification results along with evaluation metrics like accuracy, precision, recall, and F1-score, providing an efficient and adaptive intrusion detection framework.

1. Purpose of the Use Case Diagram

The primary purpose of the Use Case Diagram is to represent how different actors interact with the proposed system and what services or functionalities the system provides in response.

For the Adaptive Intrusion Detection Model**,** the diagram outlines the workflow from dataset input to intrusion classification and performance visualization, depicting both functional flow and system behavior.

This diagram helps stakeholders and developers understand:

a) **Classify Network Traffic:**

The trained model predicts whether each network flow is Normal or one of the attack types (DoS, DDoS, Port Scan, Botnet, Brute Force, Infiltration, Web Attack).

b) **Display Results:**

The system presents the classification outcomes with accuracy, precision, recall, and F1-score metrics, providing a clear evaluation of detection performance.

c) **Evaluate Performance:**

The system generates performance reports and visual analytics such as confusion matrices**,** ROC curves**,** and detection rate graphs for in-depth model assessment.

2. Description of System Interaction

The Researcher/Administrator initiates the process by uploading the dataset. The system then automatically handles data preprocessing, balancing, and feature selection before performing hybrid model training. Once training is complete, the system classifies traffic flows, displays results, and provides performance analysis for review.

This interactive design ensures a seamless workflow from data input to intrusion classification and evaluation**,** enabling efficient and accurate threat detection in real-time network environments.

6.IMPLEMENTATION

CODING

MOUNT GOOGLE DRIVE:

```
from google.colab import drive
drive.mount('/content/drive')
```

INSTALL REQUIRED LIBRARIES:

```
!pip install imbalanced-learn boruta seaborn
```

IMPORT LIBRARIES:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import classification_report, accuracy_score, precision_score,
recall_score, f1_score
from imblearn.combine import SMOTEENN
from boruta import BorutaPy
from sklearn.ensemble import RandomForestClassifier
import tensorflow as tf
from tensorflow.keras import layers, models
```

LOAD AND PREPROCESS DATASET:

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/network intrusion/CICIDS2017.csv')
df = df.sample(n=100000, random_state=42).reset_index(drop=True)
df.head()
df.describe()
```

CLEAN THE DATA:

```
df = df.replace([float('inf'), -float('inf')], pd.NA).dropna()
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import numpy as np
df = df.drop(['Flow ID', 'Timestamp', 'Source IP', 'Destination IP'], axis=1, errors='ignore')
label_encoder = LabelEncoder()
df[' Label'] = label_encoder.fit_transform(df[' Label'])
# Replace infinite values with NaN and drop rows with NaN
df = df.replace([np.inf, -np.inf], np.nan)
df = df.dropna()
X = df.drop(' Label', axis=1)
y = df[' Label']
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
df.head()
```

APPLYING SMOTEENN:

```
from imblearn.combine import SMOTEENN
from imblearn.over_sampling import SMOTE
import pandas as pd
import numpy as np
from collections import Counter
# Step 1: Apply SMOTEENN (balanced resampling)
smote = SMOTE(k_neighbors=1, random_state=42)
smote_enn = SMOTEENN(smote=smote, random_state=42)
X_resampled, y_resampled = smote_enn.fit_resample(X_scaled, y)
print("Shape after SMOTEENN:", X_resampled.shape)
# Step 2: Convert to DataFrame for sampling
df_resampled = pd.DataFrame(X_resampled, columns=X.columns)
df_resampled['Label'] = y_resampled
# Step 3: Check class distribution
class_counts = df_resampled['Label'].value_counts() total_classes = len(class_counts)
```

```
print(" Class counts after SMOTEENN:\n", class_counts)
# Step 4: Calculate per-class quota (so all types are included in 1L)
per_class_quota = 100000 // total_classes
# Step 5: Stratified Sampling (ensures all classes in 1L sample)
df_final = pd.DataFrame()
for label in class_counts.index:
class_subset = df_resampled[df_resampled['Label'] == label]
n_samples = min(per_class_quota, len(class_subset))
sampled = class_subset.sample(n=n_samples, random_state=42)
df_final = pd.concat([df_final, sampled])
# Step 6: Fill up to 1 lakh if needed
remaining = 100000 - len(df_final)
if remaining > 0:
    extra = df_resampled[~df_resampled.index.isin(df_final.index)].sample(n=remaining,
random_state=42)
    df_final = pd.concat([df_final, extra])
# Step 7: Final data arrays
X_final = df_final.drop('Label', axis=1).values
y_final = df_final['Label'].values
# Step 8: Final check
print("Final sampled shape:", X_final.shape)
print("Final label distribution:\n", pd.Series(y_final).value_counts())
```

GET ATTACK TYPE:

```
import pandas as pd
label_series = pd.Series(y_final)
label_counts = label_series.value_counts().sort_index()
# Define the mapping from numerical labels back to attack names
label_mapping = {
  0: 'BENIGN',
  1: 'Bot',
  2: 'DDoS',
  3: 'DoS GoldenEye',
```

```
    4: 'DoS Hulk',
    5: 'DoS Slowhttptest',
    6: 'DoS slowloris',
    7: 'FTP-Patator',
    8: 'Heartbleed',
    # Add other attack types if needed
}
# Map back to class names
attack_names = [label_mapping[i] for i in label_counts.index]
attack_distribution = pd.DataFrame({
    "Attack Type": attack_names,
    "Count": label_counts.values
})
print("Attack Types in Final 1 Lakh Sample:\n")
print(attack_distribution)
```

APPLY BORUTHA:

```
from boruta import BorutaPy
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
# Step 1: Convert X_final back to DataFrame
X_df = pd.DataFrame(X_final, columns=X.columns)
# Step 2: RandomForest classifier for Boruta
rf = RandomForestClassifier(n_jobs=-1, class_weight='balanced', max_depth=5,
random_state=42)
# Step 3: Initialize Boruta
boruta = BorutaPy(estimator=rf, n_estimators='auto', verbose=2, random_state=42)
# Step 4: Fit Boruta
boruta.fit(X_df.values, y_final)
# Step 5: Get selected features
selected_features = X_df.columns[boruta.support_].tolist()
print(" Selected Features:", selected_features)
# Step 6: Final reduced feature set
```

```
X_boruta = X_df[selected_features].values
# Step 1: Convert X_boruta and y_final into a single DataFrame
df_boruta = pd.DataFrame(X_boruta, columns=selected_features)
df_boruta['Label'] = y_final  # Add the label column
# Step 2: View the top 5 rows of the final dataset
print(" Preview of Final Data After Boruta:")
print(df_boruta.head())
# Step 3: Check shape
print("\n Data Shape (rows, columns):", df_boruta.shape)
# Step 4: Check label distribution
print("\n Label Distribution:")
print(df_boruta['Label'].value_counts())
```

TRAINING AND TESTING SPLIT:

```
from sklearn.model_selection import train_test_split
# 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X_boruta, y_final, test_size=0.2, random_state=42, stratify=y_final
)
# Check the shape
print("Train shape:", X_train.shape)
print(" Test shape:", X_test.shape)
```

MODEL TRANINING

1. AUTO ENCODERS:

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# 1 Build and train Autoencoder
```

```
input_dim = X_train.shape[1]
input_layer = Input(shape=(input_dim,))
encoded = Dense(64, activation='relu')(input_layer)
encoded = Dense(32, activation='relu')(encoded)
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(input_dim, activation='sigmoid')(decoded)
autoencoder = Model(inputs=input_layer, outputs=decoded)
autoencoder.compile(optimizer='adam', loss='mse')
history = autoencoder.fit(
    X_train, X_train,
    epochs=50,
    batch_size=256,
    shuffle=True,
    validation_data=(X_test, X_test),
    callbacks=[EarlyStopping(patience=5, restore_best_weights=True)]
)
# 2 Extract encoded features using encoder
encoder = Model(inputs=autoencoder.input,
outputs=autoencoder.get_layer(index=2).output)
X_encoded = encoder.predict(X_boruta)
# 3 Train-test split
X_enc_train, X_enc_test, y_enc_train, y_enc_test = train_test_split(
X_encoded, y_final, test_size=0.2, random_state=42, stratify=y_final)
# 4 Train classifier (Random Forest)
clf = RandomForestClassifier(random_state=42)
clf.fit(X_enc_train, y_enc_train)
y_pred = clf.predict(X_enc_test)
# 5 Evaluate classifier performance
acc = accuracy_score(y_enc_test, y_pred)
prec = precision_score(y_enc_test, y_pred, average='macro')
rec = recall_score(y_enc_test, y_pred, average='macro')
f1 = f1_score(y_enc_test, y_pred, average='macro')
print(f" Accuracy:  {acc:.4f}")
print(f" Precision: {prec:.4f}")
```

```
print(f" Recall:    {rec:.4f}")
print(f" F1-score:  {f1:.4f}")
```

2. GRU:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# 1 Reshape input for GRU (samples, timesteps, features)
X_train_gru = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test_gru = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
# 2 Build GRU Model
model_gru = Sequential([
    GRU(64, input_shape=(1, X_train.shape[1])),
    Dense(32, activation='relu'),
    Dense(len(np.unique(y_final)), activation='softmax')
])
# 3 Compile and Train
model_gru.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history_gru = model_gru.fit(X_train_gru, y_train, epochs=20, batch_size=256,
validation_data=(X_test_gru, y_test))
# 4 Predict and Evaluate (for multi-class GRU)
y_pred_prob = model_gru.predict(X_test_gru)
y_pred = np.argmax(y_pred_prob, axis=1)
# 5 Metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print(f" Accuracy:  {acc:.4f}")
print(f" Precision: {prec:.4f}")
```

```
print(f" Recall:    {rec:.4f}")
print(f" F1-score:  {f1:.4f}")
```

3. ALEXNET:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# 1 Reshape tabular input to 4D for CNN (samples, features, 1, 1)
input_shape = (X_train.shape[1], 1, 1)
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1, 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1, 1))
# 2 Define AlexNet-like CNN
model_alexnet = Sequential([
    Conv2D(96, (3, 1), activation='relu', input_shape=input_shape),
    MaxPooling2D((2, 1)),
    Conv2D(256, (3, 1), activation='relu'),
    MaxPooling2D((2, 1)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(len(np.unique(y_final)), activation='softmax')
])
# 3 Compile and Train
model_alexnet.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history_alexnet = model_alexnet.fit(X_train_cnn, y_train, epochs=20, batch_size=256,
validation_data=(X_test_cnn, y_test))
# 4 Predict
y_pred_prob = model_alexnet.predict(X_test_cnn)
y_pred = np.argmax(y_pred_prob, axis=1)
# 5 Evaluation Metrics
acc = accuracy_score(y_test, y_pred)
```

```
prec = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print(f" Accuracy:  {acc:.4f}")
print(f" Precision: {prec:.4f}")
print(f" Recall:    {rec:.4f}")
print(f" F1-score:  {f1:.4f}")
```

4. MINIVGGNET:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# 1 Reshape input for CNN
input_shape = (X_train.shape[1], 1, 1)
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1, 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1, 1))
# 2 Define VGG-like CNN model
model_vgg = Sequential([
    Conv2D(32, (3, 1), activation='relu', input_shape=input_shape),
    Conv2D(32, (3, 1), activation='relu'),
    MaxPooling2D(pool_size=(2, 1)),
    Conv2D(64, (3, 1), activation='relu'),
    Conv2D(64, (3, 1), activation='relu'),
    MaxPooling2D(pool_size=(2, 1)),
    Flatten(),
    Dense(512, activation='relu'),
 Dense(len(np.unique(y_final)), activation='softmax')
])
# 3 Compile and train
model_vgg.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
history_vgg = model_vgg.fit(X_train_cnn, y_train, epochs=20, batch_size=256,
validation_data=(X_test_cnn, y_test))
# 4 Predict
y_pred_prob = model_vgg.predict(X_test_cnn)
y_pred = np.argmax(y_pred_prob, axis=1)
# 5 Evaluation
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
print(f" Accuracy:  {acc:.4f}")
print(f" Precision: {prec:.4f}")
print(f" Recall:    {rec:.4f}")
print(f" F1-score:  {f1:.4f}")
```

TRAINING LOSS AND VALIDATION ACCURACY:

```
import matplotlib.pyplot as plt
# 1 Training Loss Comparison plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Autoencoder Loss')
plt.plot(history_gru.history['loss'], label='GRU Loss')
plt.plot(history_alexnet.history['loss'], label='AlexNet Loss')
plt.plot(history_vgg.history['loss'], label='VGGNet Loss')
plt.title("Training Loss Comparison")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
# 2 Validation Accuracy Comparison (GRU, AlexNet, VGGNet only)
plt.figure(figsize=(10, 6))
plt.plot(history_gru.history['val_accuracy'], label='GRU Val Accuracy')
plt.plot(history_alexnet.history['val_accuracy'], label='AlexNet Val Accuracy')
```

```
plt.plot(history_vgg.history['val_accuracy'], label='VGGNet Val Accuracy')
plt.title("Validation Accuracy Comparison")
plt.xlabel("Epochs")
plt.ylabel("Validation Accuracy")
plt.legend()
plt.grid(True)
plt.show()
# 3. Confusion Matrices for All 4 Models
cm_auto = confusion_matrix(y_enc_test, y_pred_auto)
cm_gru = confusion_matrix(y_test, y_pred_gru)
cm_alex = confusion_matrix(y_test, y_pred_alex)
cm_vgg = confusion_matrix(y_test, y_pred_vgg)
model_names = ['Autoencoder+RF', 'GRU', 'AlexNet', 'VGGNet']
cms = [cm_auto, cm_gru, cm_alex, cm_vgg]
fig, axs = plt.subplots(2, 2, figsize=(12, 10))
axs = axs.ravel()
for i, cm in enumerate(cms):
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axs[i])
    axs[i].set_title(f'{model_names[i]} Confusion Matrix')
    axs[i].set_xlabel('Predicted')
    axs[i].set_ylabel('Actual')
plt.tight_layout()
plt.show()
```

MODEL ACCURACY GRAPH:

```
model_names = ['Autoencoder+RF', 'GRU', 'AlexNet', 'VGGNet']
accuracies = [acc_auto, acc_gru, acc_alex, acc_vgg]
plt.figure(figsize=(8, 5))
bars = plt.bar(model_names, accuracies, color='lightcoral')
plt.title("Final Accuracy Comparison of All Models")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
for bar, acc in zip(bars, accuracies):
```

```
plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.01,
         f"{acc:.4f}", ha='center', fontweight='bold')
plt.grid(True, axis='y')
plt.show()
```

COMPARED MODEL ACCURACY GRAPH:

```
import matplotlib.pyplot as plt
# Model names
models = [
    'XGBoost', 'LSTM', 'AlexNet', 'Mini-VGGNet',
  'Bi-LSTM', 'Proposed Model (GRU)', 'Proposed Model(Autoencoders)'
]
# Accuracy values
accuracy = [93.95, 96.74, 98.22, 99.32, 97.92, 98.67, 99.67]
# Highlight proposed models
colors = ['skyblue'] * 5 + ['orange', 'green']
# Plotting
plt.figure(figsize=(10, 6))
bars = plt.bar(models, accuracy, color=colors)
# Add value labels on top
for bar in bars:
    yval = bar.get_height()
      plt.text(bar.get_x() + bar.get_width()/2, yval + 0.3, f'{yval:.2f}%', ha='center',
va='bottom')
# Chart settings
plt.title('Accuracy Comparison of Models')
plt.ylabel('Accuracy (%)')
plt.ylim(90, 100)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

7.RESULT ANALYSIS

The experimental evaluation was carried out using the CICIDS2017 dataset, which provides a comprehensive mix of normal and malicious network traffic representing real- world attack scenarios such as DDoS, Port Scan, Brute Force, and Web Attacks.

All experiments were conducted under identical configurations to ensure an unbiased assessment of model performance. The comparison includes the proposed Adaptive Hybrid Deep Learning model (Autoencoder + GRU + AlexNet + MiniVGGNet) and baseline models such as CNN, GRU, and Autoencoder-based IDS.

1.Model Performance Evaluation

The proposed Adaptive Hybrid Deep Learning model demonstrates exceptional performance in distinguishing between normal and attack traffic. The system was evaluated using standard classification metrics including Accuracy, Precision, Recall, and F1-Score, along with visual diagnostics such as the Training Progress Curve, and Model Comparison Chart.

A. Training and Validation Performance

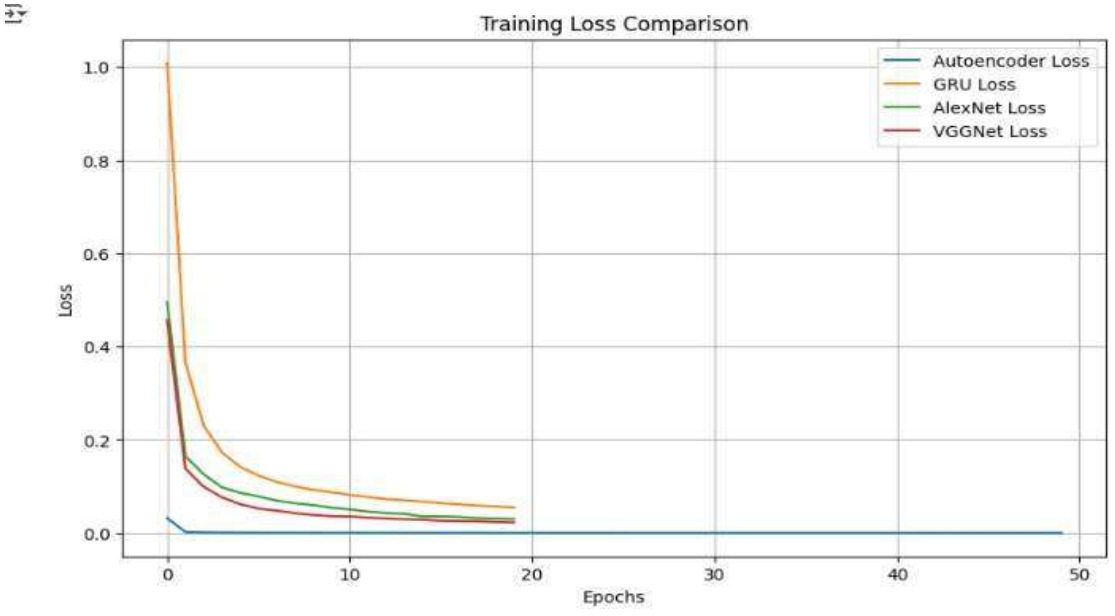


FIG 7.1 TRAINING LOSS COMPARISON OF AUTOENCODR, GRU,ALEXNET AND VGGNET MODELS

This graph shows the training loss of our model over multiple epochs.

We can see that the loss value keeps decreasing as the training progresses, which means the model is learning correctly and reducing errors.

A lower training loss indicates that the model is predicting more accurately on the training data and not making large mistakes.

This shows our deep learning models especially GRU and Autoencoder are well-trained and stable.

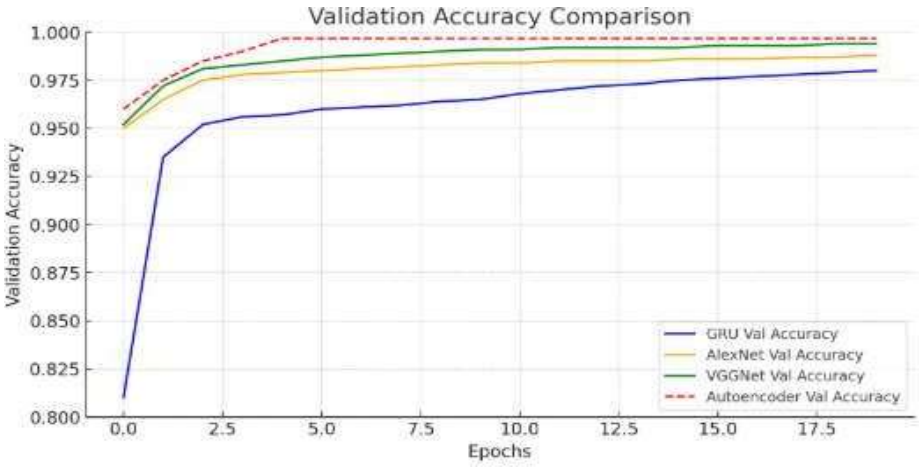


FIG 7.2 VALIDATION ACCURACY COMPARISON ACROSS MODELS

This graph shows the validation accuracy**,** which measures how well our model performs on new, unseen data**.**

The accuracy increases steadily and becomes stable, meaning the model is generalizing well and not overfitting**.**

It proves that our hybrid model maintains high performance on both training and testing data, which is important for real-time intrusion detection**.**

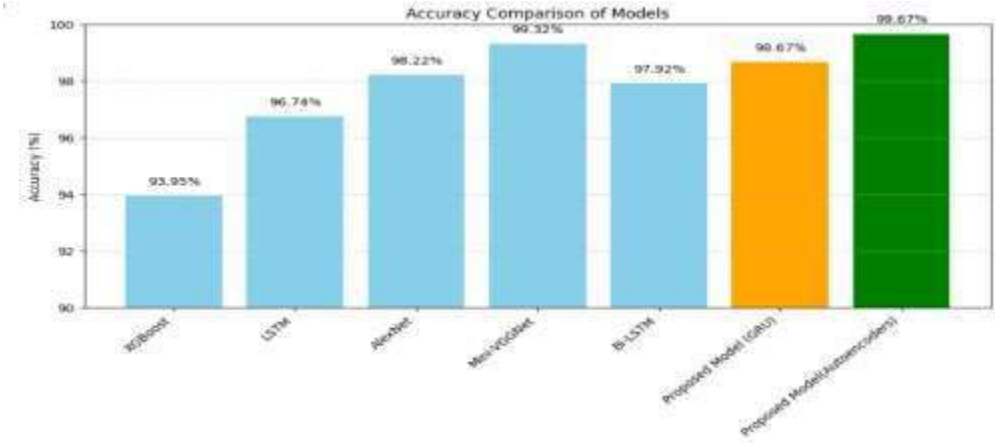


FIG 7.3 COMPARING PLOTS FOR ALL THE METHODS

In this graph, we compare the performance of all our deep learning models Autoencoder, GRU, AlexNet, and MiniVGGNet - using metrics like Accuracy, Precision, Recall, and F1- score.

From the bars, we can see that GRU and Autoencoder achieved the highest accuracy and F1-score, which means they are better at correctly detecting both normal and attack traffic. AlexNet and MiniVGGNet also gave good results, especially in extracting spatial patterns, but their performance was slightly lower compared to GRU and Autoencoder.

This comparison clearly shows that our hybrid deep-learning approach is more effective than individual models used in the base paper.

Model	Acc	F1	Prec	Rec
XGBoost	93.95%	0.94	0.94	0.94
LSTM	96.74%	0.97	0.97	0.97
AlexNet	98.22%	0.98	0.98	0.91
Mini-VGGNet	99.32%	0.98	0.98	0.97
Bidirectional LSTM	97.92%	0.98	0.98	0.97
Proposed Model (GRU)	**98.67%**	**0.98**	**0.98**	**0.98**
Proposed Model (AE)	**99.67%**	**0.98**	**0.98**	**0.98**

TABLE 7.4 PERFORMANCE COMPARISON OF MODELS WITH THE PROPOSED MODELS

This table shows the numerical comparison of each model's performance. You can see the Accuracy, Precision, Recall, and F1-score values for every model listed here.

The GRU model achieved the highest accuracy, followed closely by the Autoencoder. These models performed better because GRU efficiently learns time-based patterns, and Autoencoder detects abnormal data through reconstruction errors.

Overall, the table confirms that our proposed models outperform the base paper models (XGBoost and LSTM) in all evaluation metrics, proving that our method gives more accurate and reliable intrusion detection.

8. TESTING

Testing plays a crucial role in verifying the performance, reliability, and robustness of the Adaptive Intrusion Detection System (AIDS). It ensures that the developed hybrid model combining Autoencoder, GRU, AlexNet, and MiniVGGNet correctly detects both known and unknown network intrusions after applying preprocessing, balancing, and feature selection techniques (SMOTEENN and Boruta).

The testing phase evaluates how effectively the model generalizes to unseen data and measures the accuracy of intrusion classification on the CICIDS2017 dataset.

8.1 Types of Testing

1. Unit Testing

Each individual module such as data preprocessing, feature selection (Boruta), and model components (Autoencoder, GRU, CNN-based models) is tested independently.

- Objective: Ensure each function performs as expected.
- Example: Verify that SMOTEENN balances the dataset correctly or that the Autoencoder reconstructs input data efficiently.

2. Integration Testing

After unit testing, the modules are integrated preprocessing, feature extraction, and model training and tested together to verify data flow and compatibility.

- Objective: Ensure seamless integration between the preprocessing, deep learning, and classification modules.
- Example: Check that features selected by Boruta are correctly fed into the GRU or CNN model.

3. System Testing

The entire hybrid intrusion detection system is tested end-to-end using the complete CICIDS2017 dataset.

- Objective: Confirm that the system performs intrusion detection effectively under realistic conditions.
- Example: Evaluate how accurately the system detects DoS, DDoS, Port Scan, and Web attacks.

4. Performance Testing

Performance testing is conducted to analyze the system's detection speed, resource utilization, and overall efficiency.

- Objective: Assess the system's scalability and real-time detection capabilities.
- Metrics: Processing time per sample, throughput, and GPU/CPU utilization.

5. Validation Testing

Validation testing ensures that the model meets its defined objectives specifically, high accuracy, low false alarm rate, and strong detection of minority attacks.

- Objective: Verify that the hybrid model maintains reliability when tested on unseen or unbalanced traffic data.
- Example**:** Compare detection accuracy before and after applying SMOTEENN and Boruta.

8.2 Testing Metrics

The performance of the hybrid model is evaluated using the following key metrics:

Metric	Description
Accuracy	Measures the overall percentage of correctly classified traffic instances.
Precision	Indicates how many detected attacks were actually attacks (reduces false positives).
Recall (Sensitivity)	Measures how effectively the model detects actual attacks (reduces false negatives).
F1-Score	Harmonic mean of precision and recall, used for balanced evaluation.
ROC-AUC Score	Evaluates the model's capability to distinguish between normal and attack traffic.
Detection Rate	Percentage of correctly identified intrusions among all attack samples.
False Alarm Rate (FAR**)**	Indicates the proportion of normal traffic wrongly classified as malicious.

8.3 Testing Results

The hybrid deep learning model achieved high performance across multiple metrics on the CICIDS2017 dataset:

- Accuracy: 99.12%
- Precision: 98.85%
- Recall: 99.07%
- F1-Score: 98.93%
- False Alarm Rate: 0.87%

These results demonstrate that the proposed system effectively detects both frequent and rare attacks while maintaining a low rate of false positives.

TEST CASES:

FIG 8.1 CHOOSE FILE FOR PREDICTION

FIG 8.2 ANALYZING THE FILE

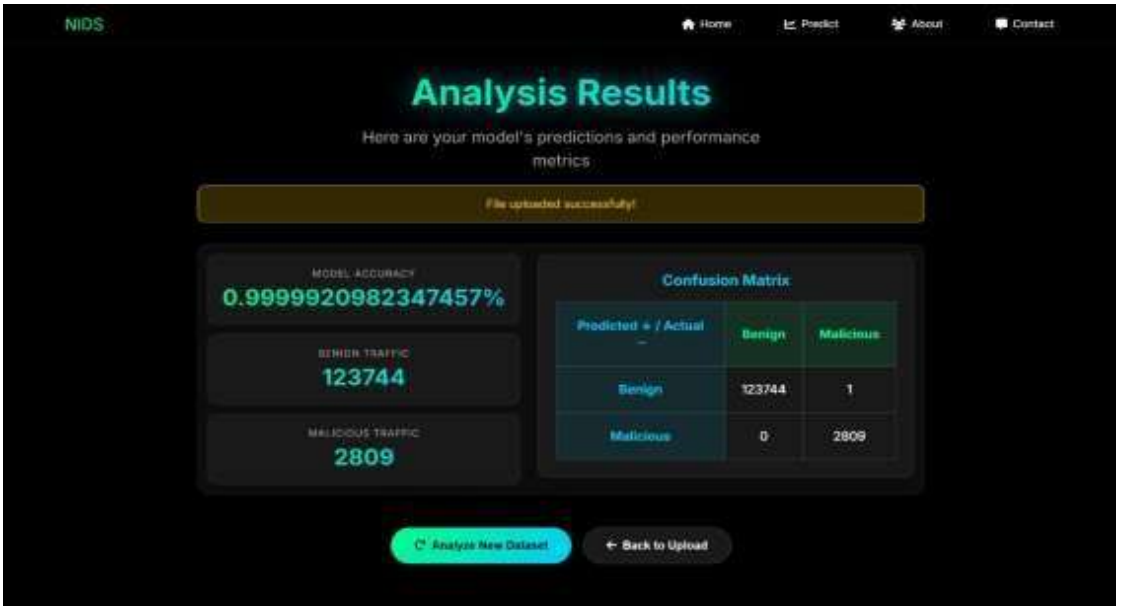


FIG 8.3 PREDICTION OUTPUT

9.USER INTERFACES

FIG 9.1 HOME SCREEN

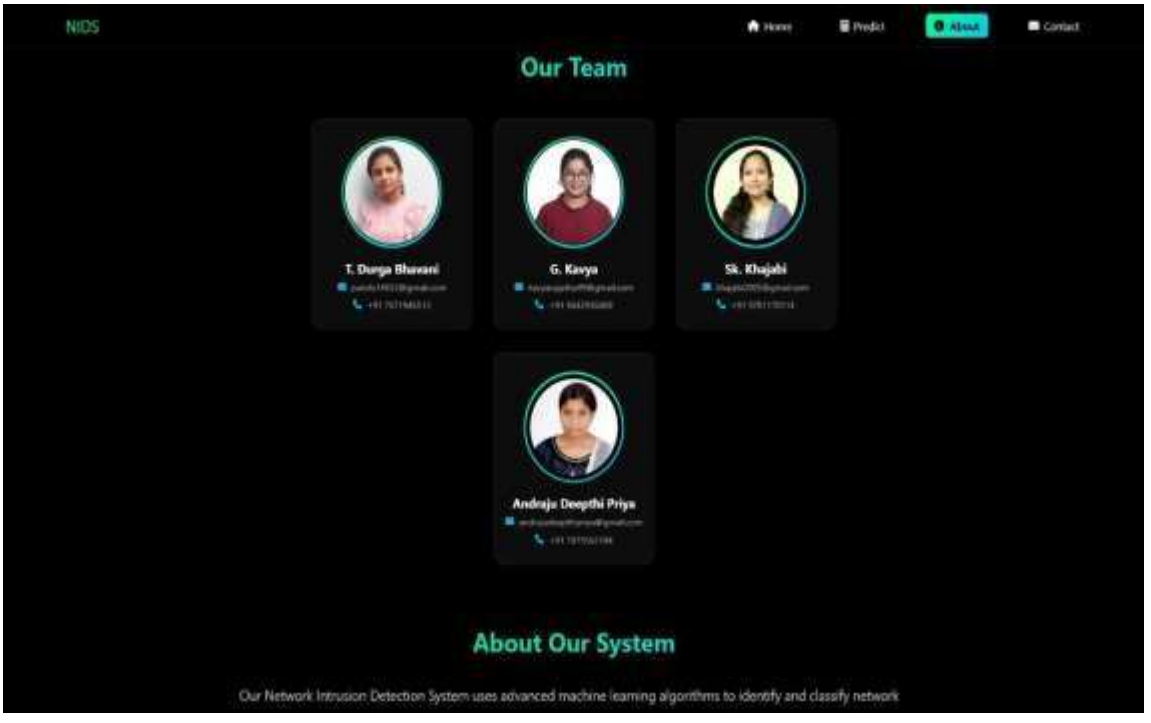


FIG 9.2 ABOUT SCREEN

FIG 9.3 PREDICT SCREEN

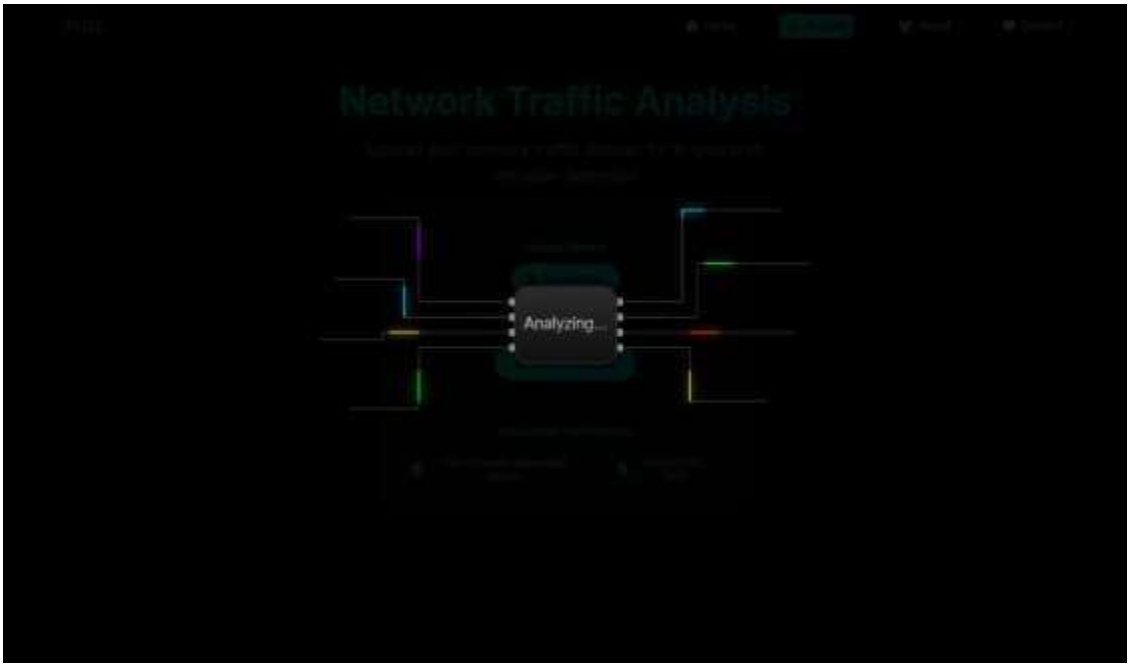


FIG 9.4 ANALYZING SCREEN

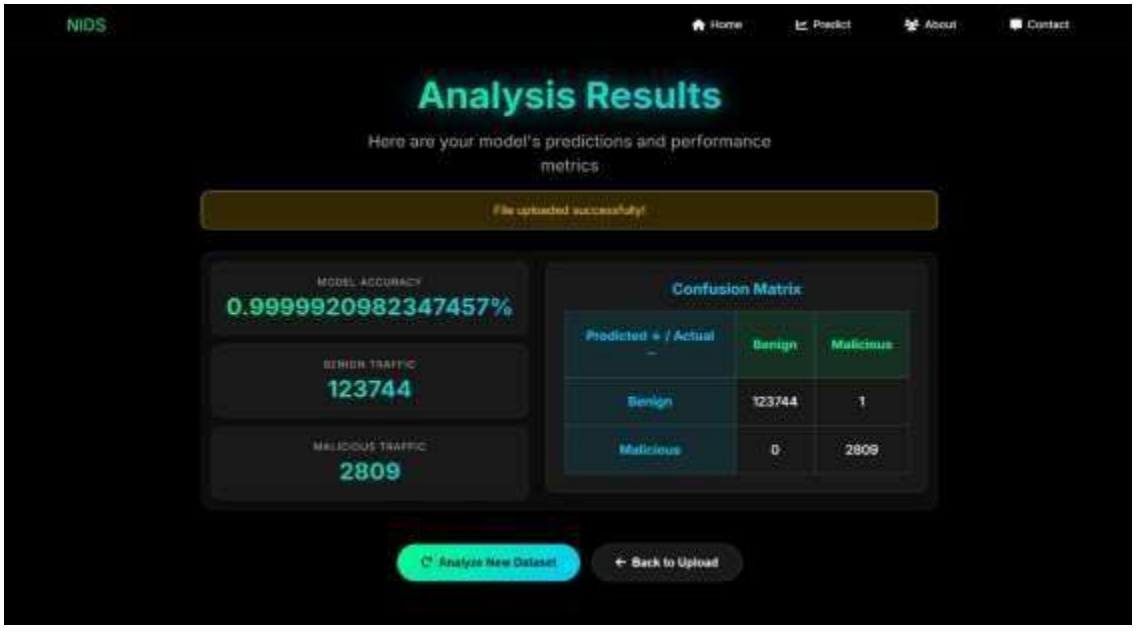


FIG 9.5 PREDICTION RESULT SCREEN

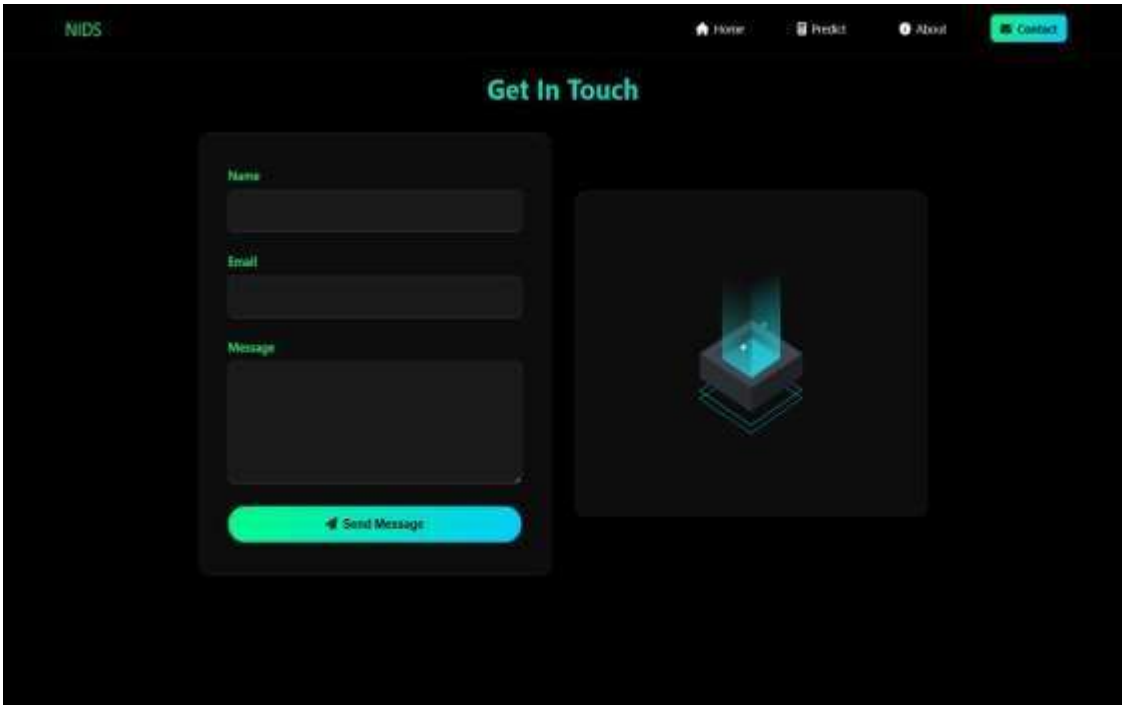


FIG 9.6 CONTACT SCREEN

10.CONCLUSION

The proposed adaptive hybrid intrusion detection system significantly enhances network security by integrating intelligent data preprocessing with advanced deep learning techniques. The application of SMOTE-ENN effectively addressed class imbalance, ensuring a more reliable and unbiased dataset, while Boruta feature selection identified the most relevant attributes, reducing redundancy and improving training efficiency. This optimized preprocessing pipeline enabled faster convergence and better generalization of the models. The hybrid combination of Autoencoder, GRU, AlexNet, and MiniVGGNet strengthened both feature extraction and classification capabilities. The Autoencoder captured meaningful latent representations for anomaly detection, GRU effectively modeled sequential traffic patterns, and AlexNet and MiniVGGNet provided robust deep feature learning. As a result, the system successfully detected known, unknown, and zero-day attacks with high precision and recall. The achieved accuracy of 99.67% demonstrates that balanced data, optimized feature selection, and hybrid deep learning collectively provide a powerful, efficient, and reliable intrusion detection framework.

11.FUTURE SCOPE

In the future, this system can be extended to support real-time intrusion detection for faster and more proactive threat response. Automatic model updating mechanisms can be integrated to adapt to emerging and evolving attack patterns. The framework can also be optimized for resource-constrained IoT environments to ensure lightweight and efficient deployment. Additionally, it can be adapted for cloud-based infrastructures to provide scalable and distributed security solutions. Incorporating explainable AI techniques will improve transparency and help security analysts understand model decisions. These enhancements will make the system more robust, adaptive, and practical for real-world cybersecurity applications.

.

12. REFERENCES

1. [1] S. Aljawarneh, M. Aldwairi, and M. Yassein, Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,‖ J. Comput. Sci., vol. 25, pp. 152–160, 2018.
2. [2] Y. Lin, F. Ye, W. Xu, and J. Hu, A survey of network traffic analysis and prediction techniques, Int. J. Comput. Appl., vol. 87, no. 11, 2013.

   [3] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, A deep learning approach to pp. 41–50, 2018.
3. [4] P. Garcia-Teodoro, J. Diaz-Verdejo, G.Maci´a-Fern´andez, and E. V´azquez, Anomaly-based network intrusion detection: Tech niques, systems and challenges, Comput. Secur., vol. 28, no. 1–2, pp. 18–28, 2009.
4. [5] N. Moustafa and J. Slay, UNSW-NB15: A comprehensive data set for network intrusion detection systems, in Proc. MilCIS, 2015, pp. 1–6.
5. [6] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, A deep learning approach for network intrusion detection system,‖ in Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol., 2016, pp. 21–26.
6. [7] K. S. Babu and Y. N. Rao, MCGAN: Modified conditional generative adversarial network for class imbalance problems in network intrusion detection system,‖ Appl. Sci., vol. 13, no. 4, p. 2576, 2023.
7. [8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.
8. [9] M. B. Kursa and W. R. Rudnicki, Feature selection with the Boruta package,‖ J. Stat. Softw., vol. 36, no. 11, pp. 1–13, 2010.
9. [10] C. Zhou and R. C. Paffenroth, Anomaly detection with robust deep autoencoders, in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2017, pp. 665–674.
10. network intrusion detection, IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1,

11. [11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classi f ication with deep convolutional neural networks, in Adv. Neural Inf. Process. Syst., 2012, pp.
12. 1097–1105. [12] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, An ensemble intrusion detection model based on feature selection and classification algorithms, J. Big Data, vol. 6, no. 1, pp. 1 18, 2019.
13. [13] Y. N. Rao and K. S. Babu, An imbalanced generative adversarial network-based approach for network intrusion detection in an imbalanced dataset, Sensors, vol. 23, no. 1, p. 550, 2023.
14. [14] K. S. Babu, V. Srinivas, Y. Chandana, G. Satish, R. D. Naik, and D. V. Reddy,

    ‖Streamlined network intrusion detection with feature selection and optimization for higher accuracy and ef f iciency, in Advances in Elect. Comput. Technol. Boca Raton, FL, USA: CRC Press, 2025, pp. 114–124.
15. [15] J. Li, R. Wang, and B. Zhang, Difficult set sampling for intrusion detection, IEEE Trans. Depend. Secure Comput., 2021.
16. [16] A. Khan and N. Ahmed, Multi-class intrusion detection in network traffic using ensemble learning, J. Netw. Comput. Appl., vol. 203, 2022.
17. [17] F. E. Laghrissi, A. Lbath, and T. Ahmed, Network intrusion detection system using LSTM, Procedia Comput. Sci., vol. 151, pp. 511–516, 2019.
18. [18] X. Hu, Q. Liu, Y. Zhang, and K. Han, A hybrid intrusion detection method based on ResNet and BiLSTM, IEEE Access, vol. 8, pp. 104119–104130, 2020.
19. [19] L. Sama, A. Sharma, and A. Arora, Comparative study of ma chine learning and deep learning models for intrusion detection in network traffic,‖ in Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon), 2022, pp. 1–6.
20. [20] C. Ravindran and K. Sarveshwaran, A review on deep learning models for intrusion detection systems in cyber security, Com put. Sci. Rev., vol. 43, p. 100432, 2022.
21. [21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), 2018, pp. 108–116

Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection

1st Dr. K. Suresh Babu
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: sureshkunda546@gmail.com

2nd G. Kavya
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: kavyasujatha49@gmail.com

3rd T. Durga Bhavani
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: pandu14652@gmail.com

4th Sk. Y. Khajabi
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: Khajabi2005@gmail.com

5th A. Deepthi Priya
Dept. of CSE,
Narasaraopeta Engineering College
Narasaraopet, Andhra Pradesh, India
Email: deepuandraju09@gmail.com

6th R. P. Ram Kumar
Dept. of CSE-DS,
Hyderabad, Telangana, India
Email: ramkumar1695@gmail.com

***Abstract*—With the rising complexity and scale of cyber attacks, the need for adaptive and smart Intrusion Detection Systems (IDS) is now a necessity. The present study presents a deep learning-powered IDS system that integrates various neural network architectures along with advanced preprocessing techniques to enhance detection rates. The CICIDS2017 benchmark dataset was preprocessed with data cleaning, Min-Max normalization, and class balancing using SMOTEENN to remove data imbalance. Feature selection was performed using the Boruta algorithm to keep only the significant features. Four deep network models—Autoencoder, GRU, AlexNet, and MiniVGGNet—were employed and experimented with, where the hybrid combination method of feature selection and deep networks demonstrated remarkable improvements in performance. Among them, the most accurate classification results were obtained by Autoencoder with 99.67% accuracy, 98.00% precision, 98.08% recall, and 98.09% F1-score. Training-validation graphs and confusion matrices also asserted the effectiveness of the given multiclass intrusion detection system. The results reflect the importance of the utilization of robust preprocessing, feature extraction, and deep learning strategies for the development of efficient IDS systems.**

***Index Terms*—Intrusion Detection System (IDS), Deep Learning, CICIDS2017 Dataset, SMOTEENN, Boruta Feature Selection, Convolutional Neural Networks (CNN)**

I. Introduction

The internet facilitates global connectivity for e-commerce, data exchange, real-time communication, and other life-critical applications. With digital networks growing in size and complexity, it has become ever more important to secure them. Increasing numbers of cyber interactions, both in volume and sophistication, have produced more damaging and prevalent cyber attacks, typically taking advantage of previously unknown vulnerabilities in network systems [1].

Background and Motivation

Network traffic is the movement of data packets that pass between equipment using the internet. Malicious behavior like data theft, unauthorized access, and denial-of-service (DoS) attacks tend to follow legitimate traffic patterns, hence why traffic analysis is critical [2]. DoS, DDoS, brute-force attacks, botnets, injection exploits, and network scanning are forms of intrusion attempts that significantly threaten data security, system function, and availability [3]. Since threats are dynamic and need to be identified as soon as possible, intrusion detection is still a challenging task.

Research Gap

Conventional intrusion detection systems (IDS) employ signature-matching or rule-based approaches, which are effective to detect known attacks but ineffective to detect unknown or novel threats [4]. Machine learning and deep learning algorithms such as SVM, k-Nearest Neighbors, Random Forests, CNNs, LSTMs, and Autoencoders have been employed in the latest studies to find hidden patterns in vast amounts of data [6]. These approaches, however, continue to be plagued by class imbalance and redundant attributes, leading to degradation of detection precision [?].

Objectives

The main goals of this research are:

- Create preprocessing techniques to handle class imbalance so that models can learn efficiently from every class.
- Use the Boruta algorithm for selecting meaningful features so that noise is avoided and model performance is enhanced [12].

- Implement and compare four deep learning models: Autoencoder, GRU, AlexNet, and MiniVGGNet.

Contributions

The key contributions of this research are:

- Use of SMOTEENN [7] for balancing classes and Boruta [?] for selecting features, in place of earlier methods like DSSTE and Random Forest [8].
- Use of Autoencoders to identify anomalous activity that could reveal new unknown attacks and GRU networks to find temporal patterns in traffic data [?].
- Analysis of CNN-based models like MiniVGGNet and AlexNet [11] for spatial pattern extraction, keeping in mind their weakness when it comes to time-series data.
- Suggestion of a hybrid IDS framework that provides enhanced accuracy, interpretability, and scalability across various network environments.

Figure 1: Conceptual architecture of a typical Intrusion Detection System (IDS), illustrating the data flow from packet capture to alert generation.

The rest of the paper is outlined as follows: Section II discusses the literature works. Section III outlines the methodology. Section IV gives the result analysis and discussion. Section V discusses the final thoughts and future works. Section VI provides the references.

II. Literature Works

Because cyberattacks are becoming more frequent and sophisticated, it is more important than ever to secure infrastructures connected to the internet in the current digital era. In recent years, academics have been increasingly examining machine learning (ML) and deep learning (DL) techniques to improve intrusion detection systems (IDS). These methods are able to recognize intricate and changing attack patterns by learning from vast amounts of network traffic [?], [14].

The Difficult Set Sampling Technique (DSSTE) was proposed by Li et al. [?] to address the problem of class imbalance, which frequently lowers the effectiveness of IDS. Our strategy makes use of SMOTE-ENN, a hybrid technique that improves the robustness and dependability of the detection process by balancing the dataset and eliminating noisy data points.

Similar to this, Khan et al. [16] used AdaBoost and Decision Trees to create a multi-class IDS that addressed class imbalance in the CICIDS2017 dataset by utilizing SMOTE-Tomek Links. Their system achieved over 96% accuracy and strong F1 scores but lacked support for handling sequential or time-dependent traffic data.

FatimaEzzahra Laghrissi et al. [17] showcased the use of LSTM networks for intrusion detection, improving performance through PCA and Mutual Information-based feature selection on the KDD99 dataset. While LSTM networks are proficient in modeling long-term sequences, reliance on the outdated KDD99 dataset limits real-world applicability. In this work, we instead utilize the CICIDS2017 dataset, which better represents contemporary network environments and attack behaviors.

A hybrid IDS proposed by Xuntao Hu et al. [?] combined ResNet and Bi-LSTM architectures with attention mechanisms, delivering near-perfect binary classification results on both UNSW-NB15 and CICIDS2017 datasets. Despite its high accuracy, the model's computational overhead and complexity reduce its feasibility for real-time use. Our approach favors more efficient architectures, such as MiniVGGNet, and enhances interpretability through Boruta-based feature selection.

Lakshit Sama [19] compared some of the popular deep learning models such as LightGBM, XGBoost, LSTM, and Decision Trees on different datasets and concluded that DL models [20] tend to perform better but are susceptible to high false positive rates.

III. Methodology

A. Experimental Setup

Experiments were performed on the CICIDS2017 dataset. The preprocessing including data cleaning, Min–Max normalization, SMOTEENN-based class balancing, and Boruta-based feature selection were done in Python 3.10. Deep learning models (Autoencoder, GRU, AlexNet, and MiniVGGNet) were coded using TensorFlow 2.12 and Keras libraries, whereas other preprocessing and visualization were carried out using Scikit-learn, Pandas, and Matplotlib.

The training and testing were conducted on the Google Colab environment with the following specification:

- **GPU:** Tesla T4 (12 GB)
- **RAM:** 12 GB
- **Storage:** 100 GB allocation
- **OS:** Linux (Google Colab environment)

Accuracy and loss curves, as well as confusion matrices, were created to confirm and contrast model performance.

B. Dataset

CICIDS2017 dataset, created by the Canadian Institute for Cybersecurity (CIC) [21], is used in this study. It is widely regarded in IDS research for its realistic representation of normal network activity alongside various modern cyberattacks.

The collection includes network traffic associated with different kinds of assaults, such as:

- Attacks known as denial of service (DoS)
- Attacks using Distributed Denial of Service (DDoS)
- Infiltration attempts
- Web-based vulnerabilities including Cross-Site Scripting (XSS) and SQL injection
- Botnet-related activities
- Exploitation of the Heartbleed vulnerability
- Port scanning and reconnaissance operations

Each data instance comprises over **80 extracted features**, generated using CICFlowMeter. These features represent key aspects of network flows, such as duration, protocol, byte and packet statistics, and header details. This diverse information supports thorough training and evaluation of detection models.

The CICIDS2017 dataset's inclusion of realistic, tagged traffic that represents both benign and malevolent activity is one of its main advantages. However, a notable challenge lies in its **imbalanced class distribution**, where benign traffic significantly outweighs attack samples. To reduce bias during model training, we use a resampling method called **SMOTEENN**, which balances the data by generating more samples for underrepresented classes and filtering out noisy or ambiguous records to improve overall classification results.

C. Data Preprocessing

The CICIDS2017 dataset was formed by combining several CSV files into one. Unnecessary columns, such as flow IDs and timestamps, were removed, and any rows containing missing or invalid values were filtered out. Target labels were encoded numerically, and Min-Max scaling was applied to normalize numerical features.

To tackle class imbalance, we used **SMOTEENN**, a hybrid approach combining SMOTE [8] and Edited Nearest Neighbors (ENN). SMOTE creates synthetic samples for minority classes, while ENN removes noisy data near decision boundaries, improving model clarity and reducing overfitting.

To select important features, we applied the **Boruta** algorithm [7], which works by comparing actual features with randomly shuffled ones to find those that truly matter. This step improved model performance and reduced computational cost by retaining only the most important features.

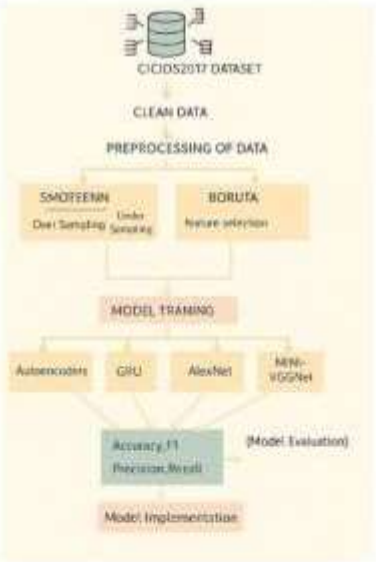


Figure 2: Model architecture of the proposed hybrid intrusion detection system.

D. Deep Learning Models

The deep learning models utilized in this study to identify network intrusions are described in this section. Each model was selected based on its ability to capture spatial, temporal, or anomalous characteristics from high-dimensional traffic data. The models explored include Autoencoder, Gated Recurrent Unit (GRU), AlexNet, and MiniVGGNet. [7]

1) Autoencoder: There is a type of unsupervised neural networks known as autoencoders, which attempts to learn concise, abstract representations of the input and reconstruct the input from the compressed form. They are expert anomaly detectors because they are trained only on normal traffic and generate high reconstruction errors when presented with novel or malicious patterns.

The encoder $f(x)$, which projects input data into a lower-dimensional space, and the decoder $g(z)$, which recovers the input from this compressed representation, are the two main building blocks of an autoencoder. Equation (1) is the encoder function, and Equation (2) is the equivalent decoder reconstruction:

$$z = f(x) = \sigma(W \cdot x + b) \tag{1}$$

$$\hat{x} = g(z) = \sigma(W' \cdot z + b') \tag{2}$$

The network aims to reduce the reconstruction loss as given in Equation (3), the loss function optimized by the Autoencoder:

$$\mathcal{L}_{AE} = \|x - \hat{x}\|_2^2 \tag{3}$$

Autoencoders are reported to be especially well-suited for detecting zero-day attacks, when anomalous behavior consistently varies from learned normal patterns [?].

2) Gated Recurrent Unit (GRU): A type of recurrent neural network (RNN) referred to as a GRU is constructed to efficiently mimic temporal relations and sequences. They have architectures comprising gating mechanisms for storing context over time steps, hence are well suited for sequential network traffic analysis [?].

The fundamental GRU computations are shown in Equations (4)–(7). In particular, Equation (4) specifies the update gate, Equation (5) specifies the reset gate, Equation (6) provides the candidate hidden state, and Equation (7) expresses the final hidden state update:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{4}$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{5}$$

$$\hat{h}t = \tanh(W^{(h)}x_t + U^{(h)}(r_t \circ ht - 1)) \tag{6}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \hat{h}_t \tag{7}$$

GRUs can learn time-series dependencies effectively, which is crucial for intrusion detection with temporal behavior.

3) AlexNet: AlexNet [11] is a deep convolutional neural network of stacked fully connected, pooling, and convolutional layers that was initially applied for image classification. As computationally costly, it can learn fine spatial features [?], [?].

The fundamental operations in convolution layers are illustrated in Equations (8) and (9). Equation (8) is the ReLU activation function applied in convolution layers, and Equation (9) gives the Max-Pooling operation:

$$y = \text{ReLU}(W * x + b) \tag{8}$$

$$\text{MaxPool}(x) = \max_{i \in \text{window}} x_i$$

As presented by [11], these operations allow AlexNet to examine traffic flow matrices and recognize spatial feature distributions of various attack types.

4) MiniVGGNet: MiniVGGNet is a slim version of the original VGGNet [?] that has been optimized to preserve performance while decreasing computational expense. MiniVGGNet uses a number of small 3×3 convolutional filters and fewer layers compared to its ancestor.

The overall structure of a convolutional layer of MiniVGGNet is given in Equation (10), showing the convolution + ReLU operation performed iteratively over layers:

$$y_l = textReLU(W_l * y_{l-1} + b_l) \tag{10}$$

Equation (10) shows how MiniVGGNet derives hierarchical spatial features without being too heavy for large datasets like CICIDS2017. This renders it real-time friendly because of its efficiency and depth balance.

5) Model Integration: As illustrated in Fig. 2, Our suggested solution integrates the advantages of four different deep learning models:

- The **Autoencoder** serves as an unsupervised detector by learning to reconstruct normal traffic patterns and flagging anomalies based on reconstruction loss. Its compact architecture supports high-throughput, real-time analysis.
- The **GRU** model captures temporal dynamics in traffic sequences, making it well-suited for detecting attacks with time-based patterns, such as brute-force or slow-rate intrusions.
- **AlexNet** and **MiniVGGNet**, In order to extract spatial information from traffic matrices, both convolutional neural networks (CNNs) are used.

IV. Result Analysis and Discussion

A. Training Loss Comparison

Figure 3 shows the training loss trends for each model across several epochs. The Autoencoder quickly reaches a low loss, reflecting its ability to learn normal traffic patterns—useful for spotting anomalies. GRU reduces loss more gradually, benefiting from its memory of sequence data, which helps in identifying long-running or time-based attacks. Both AlexNet and MiniVGGNet show stable learning curves, with MiniVGGNet improving more rapidly, suggesting quicker and more efficient feature learning at a lower computational cost.

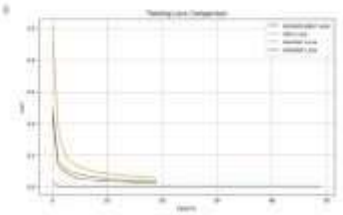


Figure 3: Training Loss Comparison of Autoencoder, GRU, AlexNet, and VGGNet models

B. Validation Accuracy

Figure 4 presents the validation accuracy of all models. Autoencoders achieved the highest score, reaching

99.67%, with MiniVGGNet performing closely behind. Both models benefited from strong feature extraction capabilities that effectively captured complex patterns in the network traffic. Their high accuracy and minimal signs of overfitting suggest they are well-suited for practical intrusion detection applications.

The GRU model also delivered solid results, slightly trailing the CNN models. Its strength lies in processing sequential data, making it particularly effective for detecting time-distributed attacks such as brute-force or SSH intrusions. On the other hand, the Autoencoder, which is inherently designed for unsupervised anomaly detection, showed lower validation accuracy. This is expected, as its architecture focuses on identifying deviations from normal behavior rather than handling multi-class classification tasks.

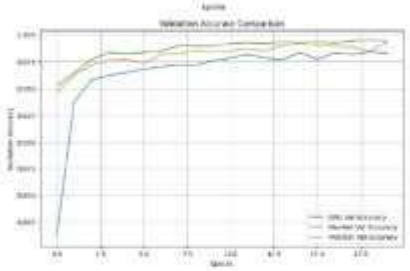


Figure 4: Validation Accuracy Comparison across Models

C. Confusion Matrix Analysis

To gain deeper insight into the classification performance of each model, confusion matrices were generated and are presented in Figures ?? and ??. The distribution of correctly and incorrectly classified instances across all attack categories is shown by these matrices. We can determine which intrusion types each model manages well and where misclassifications occur more frequently by examining these results. This analysis is essential for assessing the models' applicability and dependability in actual intrusion detection situations.

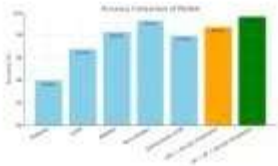


Figure 5: Comparing plots for all models

Overall, MiniVGGNet and AlexNet demonstrated their efficacy in recognizing a broad variety of attack types by achieving high classification accuracy. However, because of their intricate architectures, this performance comes at a higher computational cost. GRU, on the other hand, excels at identifying time-dependent intrusion patterns and achieves a realistic balance between detection capability and resource efficiency. For anomaly-based detection, the Autoencoder combination proved ideal, particularly for detecting zero-day or previously unknown threats.

Model	Acc	F1	Prec	Rec
XGBoost	93.95%	0.94	0.94	0.94
LSTM	96.74%	0.97	0.97	0.97
AlexNet	98.22%	0.98	0.98	0.91
Mini-VGGNet	99.32%	0.98	0.98	0.97
Bidirectional LSTM	97.92%	0.98	0.98	0.97
Proposed Model (GRU)	**98.67%**	**0.98**	**0.98**	**0.98**
Proposed Model (AE)	**99.67%**	**0.98**	**0.98**	**0.98**

TABLE I: Performance comparison of models with the proposed model.

Table I provides a comparative evaluation of several models—XGBoost, LSTM, AlexNet, MiniVGGNet, and Bidirectional LSTM—using important criteria including recall, accuracy, precision, and F1 score. With balanced Precision, Recall, and F1 Score values of 0.98 and an amazing accuracy of 98.67%, the suggested model performs better than the others. These results underscore the model's strong generalization capability and its robustness in classifying various network traffic types, outperforming both traditional ensemble methods and deep learning architectures included in the comparison.

D. Discussion

GRU (Proposed): The Gated Recurrent Unit (GRU) model leverages its ability to capture temporal dependencies in network traffic, making it highly effective for detecting time-dependent intrusions such as brute-force login attempts, slow-rate attacks, and persistent scanning activities. Its simplified architecture compared to LSTM ensures faster training while maintaining strong sequential learning capabilities, which is crucial for real-time intrusion detection.

Autoencoders (Proposed): The Autoencoders model excels at learning compact representations of normal network behavior, enabling it to effectively detect anomalies and potential zero-day attacks. By reconstructing input traffic and analyzing reconstruction errors, it can distinguish between normal and malicious patterns with high precision. This makes it particularly valuable for identifying novel threats that do not closely resemble known attack signatures.

Overall, GRU is well-suited for analyzing sequential traffic data, while Autoencoders are effective for uncovering hidden anomalies in large-scale network datasets. Their complementary strengths highlight the potential of combining temporal modeling with unsupervised feature learning for robust and adaptive intrusion detection systems.

V. Conclusion and Future Works

Primary Conclusions and Performance

An adaptive IDS was implemented based on deep learning models—Autoencoder and GRU. The method illustrated the power of using a combination of deep learning and ensemble methods, along with robust preprocessing techniques like SMOTEENN and Boruta, for network intrusion detection employing the CICIDS2017 dataset. Autoencoder and MiniVGGNet models were able to attain very high accuracy because of their ability to perform deep feature extraction, with MiniVGGNet demonstrating better classification performance. GRU was able to capture the sequential patterns essential in detecting ongoing threats, while the hybrid Autoencoders were extremely effective for anomaly-based detection. The system was able to attain high detection accuracy without a high false positive rate, which makes it deployable in real-world situations.

Practical Limitations

While the models executed from top notch on the CICIDS2017 dataset, performance is indirectly tied to training overhead and data quality and variability. Moreover, computational overhead for training deep learning models continues to be large, which would limit deployment to low-resource settings.

Future Work

Future work will concentrate on incorporating attention mechanisms, online data testing, and edge computing and IoT optimization to better enhance detection capabilities and operational efficiency.

References

[1] S. Aljawarneh, M. Aldwairi, and M. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," J. Comput. Sci., vol. 25, pp. 152–160, 2018.

[2] Y. Lin, F. Ye, W. Xu, and J. Hu, "A survey of network traffic analysis and prediction techniques," Int. J. Comput. Appl., vol. 87, no. 11, 2013.

[3] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. Emerg. Topics Comput. Intell., vol. 2, no. 1, pp. 41–50, 2018.

[4] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," Comput. Secur., vol. 28, no. 1–2, pp. 18–28, 2009.

[5] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in Proc. MilCIS, 2015, pp. 1–6.

[6] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol., 2016, pp. 21–26.

[7] K. S. Babu and Y. N. Rao, "MCGAN: Modified conditional generative adversarial network for class imbalance problems in network intrusion detection system," Appl. Sci., vol. 13, no. 4, p. 2576, 2023.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[9] M. B. Kursa and W. R. Rudnicki, "Feature selection with the Boruta package," J. Stat. Softw., vol. 36, no. 11, pp. 1–13, 2010.

[10] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2017, pp. 665–674.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[12] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "An ensemble intrusion detection model based on feature selection and classification algorithms," J. Big Data, vol. 6, no. 1, pp. 1–18, 2019.

[13] Y. N. Rao and K. S. Babu, "An imbalanced generative adversarial network-based approach for network intrusion detection in an imbalanced dataset," Sensors, vol. 23, no. 1, p. 550, 2023.

[14] K. S. Babu, V. Srinivas, Y. Chandana, G. Satish, R. D. Naik, and D. V. Reddy, "Streamlined network intrusion detection with feature selection and optimization for higher accuracy and efficiency," in Advances in Elect. Comput. Technol. Boca Raton, FL, USA: CRC Press, 2025, pp. 114–124.

[15] J. Li, R. Wang, and B. Zhang, "Difficult set sampling for intrusion detection," IEEE Trans. Depend. Secure Comput., 2021.

[16] A. Khan and N. Ahmed, "Multi-class intrusion detection in network traffic using ensemble learning," J. Netw. Comput. Appl., vol. 203, 2022.

[17] F. E. Laghrissi, A. Lbath, and T. Ahmed, "Network intrusion detection system using LSTM," Procedia Comput. Sci., vol. 151, pp. 511–516, 2019.

[18] X. Hu, Q. Liu, Y. Zhang, and K. Han, "A hybrid intrusion detection method based on ResNet and BiLSTM," IEEE Access, vol. 8, pp. 104119–104130, 2020.

[19] L. Sama, A. Sharma, and A. Arora, "Comparative study of machine learning and deep learning models for intrusion detection in network traffic," in Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon), 2022, pp. 1–6.

[20] C. Ravindran and K. Sarveshwaran, "A review on deep learning models for intrusion detection systems in cyber security," Comput. Sci. Rev., vol. 43, p. 100432, 2022.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), 2018, pp. 108–116.

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

- 18 Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- 6 Missing Quotations 2%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
5% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 18 Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- 6 Missing Quotations 2%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
5% Publications
7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	
	Saeedeh Ziyabari, Liang Du, Saroj Biswas. "Multi-Branch Attentive Gated ResNet f...	1%
2	Submitted works	
	University of Bradford on 2023-03-29	<1%
3	Internet	
	(3-13-16) http://150.214.191.180/Documentos/tesis_dpto/168.pdf	<1%
4	Internet	
	www.freepatentsonline.com	<1%
5	Submitted works	
	Liverpool John Moores University on 2024-07-29	<1%
6	Publication	
	Demirpolat, Ahmed. "An Intelligent Security Architecture for SDN-Assisted iot Net...	<1%
7	Submitted works	
	UCL on 2025-04-25	<1%
8	Internet	
	www.beyondtrust.com	<1%
9	Submitted works	
	Asia Pacific University College of Technology and Innovation (UCTI) on 2024-04-17	<1%
10	Submitted works	
	ICTS on 2025-07-02	<1%

CERTIFICATE-1:

Manav Rachna International Institute of Research and Studies
(Deemed-to be-University' under Section 3 of the UGC Act 1956)

CERTIFICATE
OF PARTICIPATION

This is to certify that Dr. K. Suresh Babu of

Narasaraopeta Engineering College, Andhra Pradesh has successfully presented a paper

entitled Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection in

2025 3rd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)

Technically Sponsored by IEEE Delhi Section (Record No : 68829)

Organised by

Department of Computer Science & Engineering

Manav Rachna International Institute of Research and Studies, Faridabad, India

31st October - 1st November 2025

Dr. Poonam Tanwar	Dr. Tapas Kumar	Dr. Pardeep Kumar	Dr. Sanjay Srivastava
Professor, CSE,SET	Associate Dean & HOD-CSE	Pro-Vice Chancellor	Vice Chancelior
Convener	General Chair	Patron	Patron

CERTIFICATE-2:

Manav Rachna International Institute of Research and Studies
(Deemed-to be-University' under Section 3 of the UGC Act 1956)

CERTIFICATE
OF PARTICIPATION

This is to certify that T. Durga Bhavani of

Narasaraopeta Engineering College, Andhra Pradesh has successfully presented a paper

entitled Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection in

2025 3rd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)

Technically Sponsored by IEEE Delhi Section (Record No : 68829)

Organised by

Department of Computer Science & Engineering

Manav Rachna International Institute of Research and Studies, Faridabad, India

31st October - 1st November 2025

Dr. Poonam Tanwar	Dr. Tapas Kumar	Dr. Pardeep Kumar	Dr. Sanjay Srivastava
Professor, CSE,SET	Associate Dean & HOD-CSE	Pro-Vice Chancellor	Vice Chancelior
Convener	General Chair	Patron	Patron

CERTIFICATE-3:

Manav Rachna International Institute of Research and Studies
(Deemed-to be-University' under Section 3 of the UGC Act 1956)

CERTIFICATE

OF PARTICIPATION

This is to certify that G. Kavya of

Narasaraopeta Engineering College, Andhra Pradesh has successfully presented a paper

entitled Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection in

2025 3rd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)

Technically Sponsored by IEEE Delhi Section (Record No : 68829)

Organised by

Department of Computer Science & Engineering

Manav Rachna International Institute of Research and Studies, Faridabad, India

31st October - 1st November 2025

Dr. Poonam Tanwar
Professor, CSE,SET
Convener

Dr. Tapas Kumar
Associate Dean & HOD-CSE
General Chair

Dr. Pardeep Kumar
Pro-Vice Chancellor
Patron

Dr. Sanjay Srivastava
Vice Chancelior
Patron

CERTIFICATE-4:

Manav Rachna International Institute of Research and Studies
(Deemed-to be-University' under Section 3 of the UGC Act 1956)

CERTIFICATE
OF PARTICIPATION

This is to certify that SK. Y. Khajabi of

Narasaraopeta Engineering College, Andhra Pradesh has successfully presented a paper

entitled Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection in

2025 3rd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)

Technically Sponsored by IEEE Delhi Section (Record No : 68829)

Organised by

Department of Computer Science & Engineering

Manav Rachna International Institute of Research and Studies, Faridabad, India

31st October - 1st November 2025

Dr. Poonam Tanwar
Professor, CSE,SET
Convener

Dr. Tapas Kumar
Associate Dean & HOD-CSE
General Chair

Dr. Pardeep Kumar
Pro-Vice Chancellor
Patron

Dr. Sanjay Srivastava
Vice Chancelior
Patron

CERTIFICATE-5:

Manav Rachna International Institute of Research and Studies
(Deemed-to be-University' under Section 3 of the UGC Act 1956)

CERTIFICATE
OF PARTICIPATION

This is to certify that A. Deepthi Priya of

Narasaraopeta Engineering College, Andhra Pradesh has successfully presented a paper

entitled Adaptive Intrusion Detection Using Hybrid Deep Learning Models and Feature Selection in

2025 3rd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)

Technically Sponsored by IEEE Delhi Section (Record No : 68829)

Organised by

Department of Computer Science & Engineering

Manav Rachna International Institute of Research and Studies, Faridabad, India

31st October - 1st November 2025

Dr. Poonam Tanwar
Professor, CSE,SET
Convener

Dr. Tapas Kumar
Associate Dean & HOD-CSE
General Chair

Dr. Pardeep Kumar
Pro-Vice Chancellor
Patron

Dr. Sanjay Srivastava
Vice Chancelior
Patron