AN ENHANCED AI BASED NETWORK INTRUSION DETECTION SYSTEM USING GENERATIVE ADVERSARIAL NETWORKS

A PROJECT REPORT

Submitted by

THILAGAVATHI A **(612721104111)**

PRIYADHARSHINI R **(612721104075)**

SWETHA K **(612719114107)**

SILVIYA M **(612721104098)**

In partial fulfillment of the requirement

for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THE KAVERY ENGINEERING COLLEGE

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

MECHERI, SALEM-636453

MAY 2025

AN ENHANCED AI BASED NETWORK INTRUSION DETECTION SYSTEM USING GENERATIVE ADVERSARIAL NETWORKS

A PROJECT REPORT

Submitted by

THILAGAVATHI A	**(612721104111)**
PRIYADHARSHINI R	**(612721104075)**
SWETHA K	**(612719114107)**
SILVIYA M	**(612721104098)**

In partial fulfilment of the requirement

for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THE KAVERY ENGINEERING COLLEGE

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

MECHERI, SALEM-636453

MAY 2025

THE KAVERY ENGINEERING COLLEGE

MECHERI, SALEM - 636453

BONAFIDE CERTIFICATE

Certified that this project report titled **"AN ENHANCED AI BASED NETWORK INTRUSION DETECTION SYSTEM USING GENERATIVE ADVERSARIAL NETWORKS"** is the bonafide work of **THILAGAVATHI A (612721104111), PRIYADHARSHINI R (612721104075), SWETHA K (612721104107), SILVIYA M (612721104098)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr.M BALAMURUGAN M.E., Ph.D.,
HEAD OF THE DEPARTMENT
ASSOCIATE PROFESSOR,
Department of Computer Science and Engineering
The Kavery Engineering College,
Mecheri,
Salem-636453.

SIGNATURE
Mrs. B SHANMUGAPRIYA M.E.,
SUPERVISOR
ASSISTANT PROFESSOR,
Department of Computer Science and Engineering
The Kavery Engineering College,
Mecheri,
Salem-636453.

Submitted for VIVA-VOCE Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

We jointly declare that the project report on **"AN ENHANCED AI BASED NETWORK INTRUSION DETECTION SYSTEM USING GENERATIVE ADVERSARIAL NETWORKS"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of Computer Science And Engineering. This project report is submitted on the partial fulfilment of the requirement of the award of Degree Computer Science And Engineering.

SIGNATURE

THILAGAVATHI A
(612721104111)

PRIYADHARSHINI R
(612721104075)

SWETHA K
(612721104107)

SILVIYA M
(612721104098)

Place: SALEM
Date :

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to our Advisor **Dr. A.K.NATESAN,** our Chairman Thiru **A.ANBALAGAN,** our Secretary Prof **S.K.ELANGOVAN** for providing immense facilities at our institution.

We would like to acknowledge our Principal **Dr.V DURAISAMY M.E., Ph.D., FIE,**for fostering a supportive learning environment that encourages collaboration and teamwork. Your leadership has set a positive example for all of us and has created a culture of academic excellence.

We wish to extend our heartfelt gratitude to our **Head of The Department Dr.M.BALAMURUGAN M.E., Ph.D.,** for your support and guidance throughout the completion of the thesis.

We are in deep gratitude to our department faculty who always been supporting us through thick and thin respected **Project Coordinator Mrs. C.SOUNDARYA M.E.,** Assistant Professor for the engulf support for the project.

We are highly indebted to provide our heart full thanks to our **Project Guide Mrs.B.SHANMUGAPRIYA M.E.,** Assistant Professor for his valuable ideas, encouragement and supportive guidance throughout the project.

We wish to extend our sincere thanks to all faculty members of our **COMPUTER SCIENCE AND ENGINEERING** for guiding us and providing valuable feedback along the way for successful completion of this project.

We would like to express my sincere gratitude to all the students in my group for their hard work, dedication, and cooperation throughout the preparation and editing stages of the manuscript.

We would like to thank all those who have contributed to the success of our college project. Your support and guidance have been truly appreciated, and we could not have completed this project without your help.

ABSTRACT

The rapid expansion of digital infrastructure, network security has become a critical challenge. Traditional intrusion detection systems (IDS) often struggle with high false-positive rates and poor adaptability to new attack patterns. To address these issues, this paper proposes an enhanced AI-based Network Intrusion Detection System (NIDS) using Generative Adversarial Networks (GANs). GANs, consisting of a generator and a discriminator, enable the system to detect anomalies more effectively by learning complex attack patterns from network traffic data. The generator produces synthetic attack scenarios, improving the model's ability to recognize both known and novel threats, while the discriminator distinguishes between legitimate and malicious traffic. Unlike conventional machine learning-based IDS, which rely on static datasets, the proposed system continuously evolves, improving its detection accuracy over time.

This approach reduces false positives, enhances threat detection capabilities, and adapts to emerging cyber threats with minimal manual intervention. Experimental results demonstrate that the GAN-based NIDS outperforms traditional machine learning models, achieving higher precision, recall, and F1-score metrics. Additionally, the system is evaluated using benchmark datasets, ensuring its robustness in real-world cybersecurity applications. The proposed framework can be integrated into existing security infrastructures, providing an intelligent and proactive defense mechanism against cyberattacks. This research highlights the potential of AI-driven approaches in strengthening network security and sets the foundation for future advancements in automated cyber defense strategies.

TABLE OF CONTENT

TABLE OF FIGURES

LIST OF ABBREVIATIONS

GAN	Generative Adversarial Networks
IDS	Intrusion Detection Syetm
NIDS	Network Intrusion Detection System
AI	Aritificial Intelligence
ML	Machine Learning
AML	Adversarial Machine Learning
WSN	Wireless Sensor Networks
CGAN	Conditional Generative Adversarial Networks
SGAN	Self-Attention-Based Generative Adversarial Networks
HIDS	Host Intrusion Detection System
VAE	Variational Auto Encoders
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
KNN	K-Nearest Neighbors
IOT	Internet Of Things
DCGA	Deep Convolutional Generative Adversarial Networks
CEC	Collaborative Edge Computing
SIOT	Social Internet Of Things
DDOS	Distributed Denial Of Services
ICS	Industrial Control System

CHAPTER-1

INTRODUCTION

The increasing reliance on digital networks, cybersecurity threats have become a major concern for organizations and individuals alike. Cyberattacks, such as malware infections, Distributed Denial of Service (DDoS) attacks, and data breaches, continue to evolve in complexity, making traditional security mechanisms insufficient for modern network defense. Network Intrusion Detection Systems (NIDS) play a crucial role in identifying and mitigating such threats by monitoring network traffic for malicious activity. However, conventional IDS solutions often suffer from high false-positive rates and limited adaptability to new attack patterns, reducing their overall effectiveness.

To address these challenges, artificial intelligence (AI) and machine learning (ML) techniques have been integrated into cybersecurity solutions, enabling systems to learn from data and improve detection accuracy. One promising approach is the use of Generative Adversarial Networks (GANs), a deep learning framework consisting of a generator and a discriminator. GANs have shown significant potential in enhancing IDS performance by generating synthetic attack patterns, improving the system's ability to detect both known and unknown threats. The generator learns to produce realistic attack scenarios, while the discriminator distinguishes between normal and malicious traffic, continuously refining its classification ability.

This study proposes an AI-based NIDS leveraging GANs to enhance network security by dynamically adapting to evolving cyber threats. Unlike traditional rule-based IDS, which require frequent updates, the proposed model continuously learns from real-time network data, reducing false alarms and improving threat detection accuracy. Experimental evaluations using benchmark datasets demonstrate that the GAN-based approach outperforms conventional machine learning models in detecting cyber intrusions. By integrating AI-driven techniques, this research aims to provide a more robust and intelligent cybersecurity framework, ensuring proactive threat mitigation in an increasingly complex digital landscape.

1.1 WORKING CONTRIBUTION

This research presents an enhanced AI-based Network Intrusion Detection System (NIDS) leveraging Generative Adversarial Networks (GANs) to address key challenges in cybersecurity. The system introduces a dynamic learning mechanism where the GAN's generator produces synthetic attack scenarios, enabling the model to detect both known and novel threats more effectively. By incorporating adversarial learning, the proposed NIDS reduces reliance on static datasets and continuously evolves to adapt to emerging cyberattacks with minimal human intervention.

The discriminator network is trained to distinguish between benign and malicious traffic with higher accuracy, resulting in improved precision, recall, and F1-score compared to traditional machine learning approaches. A major contribution of this work is the significant reduction in false positives, enhancing the system’s reliability for real-world applications.

Experimental validation using benchmark datasets such as NSL-KDD and CICIDS2017 demonstrates the robustness and scalability of the proposed system. Additionally, the framework is designed for seamless integration into existing security infrastructures, making it a practical and intelligent solution for proactive cyber defense. This research not only advances the field of AI-driven intrusion detection but also lays a strong foundation for future developments in automated and adaptive cybersecurity strategies.

1.2 OBJECTIVE

The primary objective of this research is to develop an enhanced AI-based Network Intrusion Detection System (NIDS) utilizing Generative Adversarial Networks (GANs) to significantly improve the detection of cybersecurity threats. The system aims to overcome the limitations of traditional IDS models, particularly the high false-positive rates and poor adaptability to new and evolving attack patterns. By leveraging the adversarial training mechanism of GANs, the proposed NIDS seeks to generate synthetic attack scenarios that enhance the model’s ability to detect both known and previously unseen intrusions. Furthermore, the objective is to create a dynamic and self-evolving detection framework that continuously learns from new data, reduces manual intervention, and maintains high detection accuracy over time.

Additional goals include minimizing false alarms, improving system robustness across diverse network environments, and ensuring seamless integration into existing cybersecurity infrastructures. Ultimately, this research aspires to advance automated cyber defense mechanisms and contribute to the development of intelligent, adaptive, and proactive network security solutions.

CHAPTER-2

LITERATURE SURVEY

Park, Cheolhee, Jonghoon Lee, Youngsoo Kim, Jong-Geun Park, Hyunjin Kim, and Dowon Hong. "An enhanced AI-based network intrusion detection system using generative adversarial networks." IEEE Internet of Things Journal 10, no. 3 (2022): 2330-2345.

Park et al. (2022) present a significant advancement in the field of cybersecurity by proposing an enhanced AI-based Network Intrusion Detection System (NIDS) utilizing Generative Adversarial Networks (GANs). Traditional intrusion detection systems (IDS) face persistent challenges, particularly high false-positive rates, limited adaptability to new and sophisticated cyberattacks, and a heavy reliance on static, outdated datasets. These limitations severely reduce the effectiveness of conventional IDS models in modern, dynamic network environments. Recognizing these issues, Park et al. introduce an innovative framework that leverages the power of GANs, a class of deep learning models consisting of two neural networks — a generator and a discriminator — that compete in a zero-sum game to improve their respective performances.

In their proposed system, the generator is trained to create synthetic attack data that closely mimics real-world malicious traffic patterns. This synthetic data is then used to train the discriminator, whose role is to accurately distinguish between benign and malicious network activities. Through this adversarial process, the model learns to recognize even subtle and previously unseen attack signatures. Unlike traditional machine learning approaches, which often fail to detect novel threats due to their static nature, the GAN-based NIDS evolves continuously, adapting to emerging cybersecurity challenges with minimal human intervention. This dynamic learning capability significantly enhances the system's threat detection performance.

To evaluate the effectiveness of their proposed model, Park et al. conduct extensive experiments using well-known benchmark datasets such as NSL-KDD and CICIDS2017, which are widely used in the cybersecurity research community. The results demonstrate that their GAN-based NIDS outperforms conventional machine learning-based IDS in several critical performance metrics, including accuracy, precision, recall, and F1-score.

Specifically, the proposed system achieves higher detection rates for both known and unknown attack types while simultaneously reducing false alarms. This balance is critical for practical deployment, as high false-positive rates can overwhelm security analysts and reduce the trust in automated detection systems.

Moreover, the study highlights the broader potential of adversarial learning in the cybersecurity domain. By continuously generating diverse and realistic attack scenarios, the system ensures that the discriminator is exposed to a wide range of threat behaviors, thus improving its generalization capabilities. Park et al. argue that this approach not only enhances immediate detection performance but also contributes to building more resilient and future-proof security infrastructures. The ability of the model to autonomously adapt to evolving threat landscapes without frequent manual retraining represents a major step forward in the automation of cyber defense mechanisms.

The research also discusses the practical considerations for integrating the proposed GAN-based NIDS into existing security infrastructures. Due to its modular and scalable design, the system can be deployed alongside traditional security tools, enhancing overall network protection without necessitating a complete overhaul of existing frameworks. In conclusion, Park et al. (2022) provide a compelling case for the adoption of AI-driven methodologies in network security. Their work underscores the critical role of advanced machine learning techniques, such as GANs, in addressing the limitations of traditional IDS and shaping the future of intelligent, automated cybersecurity solutions.

Shahriar, Md Hasan, Nur Imtiazul Haque, Mohammad Ashiqur Rahman, and Miguel Alonso. "G-ids: Generative adversarial networks assisted intrusion detection system." In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 376-385. IEEE, 2020.

Shahriar et al. (2020) present G-IDS, a novel Intrusion Detection System (IDS) enhanced by Generative Adversarial Networks (GANs), aiming to overcome persistent challenges in the field of cybersecurity. Traditional IDS approaches, particularly those relying on static machine learning models, often struggle with accurately detecting novel attack patterns and exhibit high false-positive rates, leading to operational inefficiencies and security vulnerabilities.

To tackle these limitations, the authors introduce a GAN-assisted framework where the generator network is trained to create realistic synthetic attack scenarios, thereby enriching the diversity and complexity of the training data. The discriminator network, tasked with distinguishing between normal and malicious traffic, benefits from this adversarial learning process by becoming more adept at identifying a broader range of cyber threats, including previously unseen attacks.

In their methodology, Shahriar et al. leverage publicly available intrusion detection datasets to train and validate the G-IDS system, ensuring the approach's robustness and applicability to real-world network environments. The GAN framework is carefully designed to minimize the risk of overfitting while maximizing the system’s ability to generalize across different types of attack vectors. Through comprehensive experimentation, the authors demonstrate that G-IDS significantly outperforms conventional machine learning-based IDS models in key performance metrics such as detection accuracy, recall, precision, and F1-score. Notably, the G-IDS model exhibits a marked reduction in false-positive rates, addressing one of the most critical shortcomings of existing IDS solutions.

The study highlights several major contributions. Firstly, by synthesizing a wide range of attack scenarios, G-IDS enhances the diversity of the training dataset without the need for extensive manual data collection or labeling. This ability to generate realistic and varied training examples allows the discriminator to develop a more nuanced understanding of malicious behaviors. Secondly, the adversarial learning process ensures that the system evolves dynamically, maintaining high detection performance even as new and sophisticated cyberattacks emerge. Unlike traditional IDS that require frequent retraining with manually curated datasets, G-IDS demonstrates the potential for continuous, autonomous improvement.

Shahriar et al. also emphasize the practical implications of integrating G-IDS into existing cybersecurity infrastructures. Due to its modular design, G-IDS can be deployed alongside traditional security tools, enhancing threat detection capabilities without requiring significant changes to current systems. Furthermore, the study positions GANs as a highly promising tool for the future of automated and intelligent intrusion detection, capable of adapting to the increasingly complex and evolving threat landscape.

In conclusion, the research by Shahriar et al. (2020) underscores the transformative potential of adversarial learning techniques in cybersecurity. By effectively addressing the limitations of static, machine learning-based IDS models, G-IDS marks a significant

advancement in the development of resilient, adaptive, and efficient network security systems. The study not only provides strong empirical evidence supporting the use of GANs in intrusion detection but also lays the groundwork for further exploration into AI-driven methods for proactive cyber defense.

Iliyasu, Auwal Sani, and Huifang Deng. "N-GAN: a novel anomaly-based network intrusion detection with generative adversarial networks." International Journal of Information Technology 14, no. 7 (2022): 3365-3375.

The paper "N-GAN: A Novel Anomaly-Based Network Intrusion Detection with Generative Adversarial Networks" by Iliyasu, Auwal Sani, and Huifang Deng (2022) explores the use of Generative Adversarial Networks (GANs) for anomaly-based network intrusion detection. Traditional intrusion detection systems (IDS) struggle with detecting novel cyber threats due to evolving attack patterns. The authors propose N-GAN, a deep learning-based model that leverages GANs to enhance the accuracy and adaptability of IDS. The model consists of a generator, which learns the distribution of normal network traffic, and a discriminator, which detects deviations indicating potential intrusions. Through extensive experimentation, N-GAN demonstrates superior detection performance compared to conventional machine learning approaches. The results highlight its effectiveness in identifying both known and unknown attacks, reducing false positives, and improving cybersecurity defenses. This research contributes to the development of more robust and intelligent IDS using deep learning techniques.

Alabugin, Sergei K., and Alexander N. Sokolov. "Applying of generative adversarial networks for anomaly detection in industrial control systems." In 2020 global smart industry conference (GloSIC), pp. 199-203. IEEE, 2020..

Network security is a growing concern as cyber threats continue to evolve, particularly in critical infrastructure such as Industrial Control Systems (ICS). Traditional Intrusion Detection Systems (IDS) rely on signature-based or anomaly-based methods, but these approaches often struggle with detecting zero-day attacks and generating high false positive rates. In their study, Alabugin and Sokolov (2020) explore the application of Generative Adversarial Networks (GANs) for anomaly detection in Industrial Control

Systems to address these limitations. GANs consist of a generator, which creates synthetic attack samples, and a discriminator, which differentiates between normal and malicious data, enhancing the detection of previously unseen threats. Their research demonstrates that using GANs improves the adaptability of intrusion detection models by learning complex attack patterns and refining classification accuracy. The study evaluates performance using standard ICS datasets, showing that the GAN-based model outperforms traditional methods in identifying anomalies with greater precision.

Despite computational challenges, their findings highlight the potential of AI-driven solutions in securing critical infrastructure. The research contributes to the growing field of AI-based cybersecurity by demonstrating how adversarial learning techniques can strengthen network defenses, making ICS environments more resilient against emerging cyber threats.

Huang, Shuokang, and Kai Lei. "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks." Ad Hoc Networks 105 (2020): 102177.

The paper "IGAN-IDS: An Imbalanced Generative Adversarial Network Towards Intrusion Detection System in Ad-Hoc Networks" by Shuokang Huang and Kai Lei (2020) addresses the challenge of intrusion detection in ad-hoc networks, where imbalanced data can hinder the performance of traditional machine learning models. The authors propose IGAN-IDS, a novel intrusion detection system (IDS) that utilizes a Generative Adversarial Network (GAN) to handle the imbalanced nature of network traffic data. In this approach, the generator creates synthetic minority-class attack samples, improving the training process of the IDS. The discriminator then differentiates between normal and attack traffic, enhancing detection accuracy.

Through extensive experimentation on real-world datasets, IGAN-IDS demonstrates significant improvements in detecting rare cyber threats, reducing false positives, and increasing overall system reliability. The study highlights the importance of GANs in cybersecurity, particularly for handling data scarcity and imbalance issues in dynamic, decentralized ad-hoc networks. This research provides valuable insights into the application of deep learning for network security, suggesting that GAN-based models can effectively enhance the robustness and adaptability of IDS. The findings contribute to the development

of more accurate and efficient intrusion detection mechanisms for modern network infrastructures.

Alhajjar, Elie, Paul Maxwell, and Nathaniel Bastian. "Adversarial machine learning in network intrusion detection systems." Expert Systems with Applications 186 (2021): 115782.

The paper "Adversarial Machine Learning in Network Intrusion Detection Systems" by Elie Alhajjar, Paul Maxwell, and Nathaniel Bastian (2021) explores the impact of adversarial machine learning (AML) on intrusion detection systems (IDS). As cyber threats evolve, attackers increasingly use adversarial techniques to bypass traditional IDS models by manipulating input data. The authors analyze how AML techniques, such as adversarial evasion and poisoning attacks, can degrade IDS performance and propose countermeasures to enhance system robustness.

The study reviews different IDS architectures, including signature-based, anomaly-based, and hybrid models, highlighting their vulnerabilities to adversarial attacks. The authors then discuss various adversarial defense mechanisms, including adversarial training, feature selection, and model ensembling, to improve IDS resilience.

The research emphasizes the urgent need for IDS frameworks that integrate adversarial defense strategies to mitigate evolving cyber threats. By incorporating AML-resistant techniques, IDS models can become more robust against sophisticated attacks.

This study contributes to the growing field of cybersecurity by demonstrating how adversarial learning affects network security and providing insights into improving IDS through proactive adversarial defenses.

Sood, Tanya, Satyartha Prakash, Sandeep Sharma, Abhilash Singh, and Hemant Choubey. "Intrusion detection system in wireless sensor network using conditional generative adversarial network." Wireless Personal Communications 126, no. 1 (2022): 911-931.

The paper "Intrusion Detection System in Wireless Sensor Network Using Conditional Generative Adversarial Network" by Tanya Sood, Satyartha Prakash, Sandeep Sharma,

Abhilash Singh, and Hemant Choubey (2022) explores the use of Conditional Generative Adversarial Networks (CGANs) to enhance intrusion detection in Wireless Sensor Networks (WSNs). WSNs are crucial for various applications, but their limited computational resources and decentralized nature make them vulnerable to cyber threats. Traditional Intrusion Detection Systems (IDS) struggle with high false positive rates and imbalanced datasets, leading to inefficient attack detection.

To address these challenges, the authors propose a CGAN-based IDS that generates synthetic attack samples to balance training data, improving the model's ability to detect rare and evolving cyber threats. The CGAN model conditions the data generation process based on specific attack types, making it more effective than traditional GANs. Experimental results on real-world datasets show that this approach enhances detection accuracy, reduces false alarms, and improves generalization to unseen attacks.

This study highlights the potential of deep learning, particularly CGANs, in improving network security. By integrating adversarial learning techniques, IDS in WSNs can become more resilient and adaptable, contributing to more robust and intelligent cybersecurity solutions for modern wireless networks.

Aldhaheri, Sahar, and Abeer Alhuzali. "SGAN-IDS: Self-attention-based generative adversarial network against intrusion detection systems." Sensors 23, no. 18 (2023): 7796.

The paper "SGAN-IDS: Self-Attention-Based Generative Adversarial Network Against Intrusion Detection Systems" by Sahar Aldhaheri and Abeer Alhuzali (2023) presents an advanced intrusion detection system (IDS) leveraging self-attention-based Generative Adversarial Networks (SGANs). Traditional IDS models often struggle with identifying complex cyber threats due to issues like imbalanced datasets, high false positives, and evolving attack techniques.

To address these challenges, the authors propose SGAN-IDS, which integrates a self-attention mechanism into GANs to enhance feature representation and improve intrusion detection accuracy.

The self-attention mechanism enables the model to focus on critical network traffic patterns, enhancing its ability to distinguish between normal and malicious activities. The

GAN-based framework generates synthetic samples to address class imbalance, improving IDS performance. Experimental evaluations on benchmark datasets show that SGAN-IDS outperforms traditional machine learning and deep learning approaches in terms of detection accuracy, precision, recall, and robustness against adversarial attacks.

This study highlights the importance of self-attention in deep learning for cybersecurity, demonstrating how SGAN-IDS can provide enhanced anomaly detection, better generalization, and improved resilience against sophisticated cyber threats. The findings contribute to the development of more adaptive and intelligent next-generation IDS solutions.

Idrissi, Idriss, Mostafa Azizi, and Omar Moussaoui. "An unsupervised generative adversarial network based-host intrusion detection system for internet of things devices." Indones. J. Electr. Eng. Comput. Sci 25, no. 2 (2022): 1140-1150.

The paper "An Unsupervised Generative Adversarial Network-Based Host Intrusion Detection System for Internet of Things Devices" by Idriss Idrissi, Mostafa Azizi, and Omar Moussaoui (2022) proposes a novel unsupervised intrusion detection system (IDS) for Internet of Things (IoT) devices using Generative Adversarial Networks (GANs).

IoT devices are highly vulnerable to cyber threats due to limited computational resources, lack of security measures, and constant connectivity. Traditional supervised learning-based IDS models require labeled datasets, which are often unavailable or imbalanced in real-world IoT environments.

To overcome these limitations, the authors develop an unsupervised GAN-based Host Intrusion Detection System (HIDS) that learns the normal behavior of IoT devices and detects anomalies without requiring labeled data. The generator creates synthetic network traffic, while the discriminator distinguishes between legitimate and malicious activities. The model is evaluated on benchmark IoT datasets, demonstrating high accuracy in anomaly detection, reduced false positives, and adaptability to emerging cyber threats.

This research underscores the potential of GANs in securing IoT ecosystems, proving that unsupervised learning can be an effective approach to intrusion detection. The findings contribute to developing more autonomous, scalable, and intelligent cybersecurity solutions for IoT networks.

Zhao, Shuang, Jing Li, Jianmin Wang, Zhao Zhang, Lin Zhu, and Yong Zhang. "attackgan: Adversarial attack against black-box ids using generative adversarial networks." Procedia Computer Science 187 (2021): 128-133.

The paper "AttackGAN: Adversarial Attack Against Black-Box IDS Using Generative Adversarial Networks" by Shuang Zhao, Jing Li, Jianmin Wang, Zhao Zhang, Lin Zhu, and Yong Zhang (2021) explores how Generative Adversarial Networks (GANs) can be used to generate adversarial attacks against black-box Intrusion Detection Systems (IDS). Black-box IDS models, which operate without revealing their internal structure, are widely used for detecting cyber threats. However, they remain vulnerable to adversarial attacks, where small modifications to input data can mislead the IDS into misclassifying malicious activities as benign.

To exploit this weakness, the authors propose AttackGAN, a GAN-based framework that generates adversarial network traffic designed to evade black-box IDS models. The generator creates malicious samples that appear normal, while the discriminator refines the attack strategy by learning the IDS's decision boundaries.

Experimental evaluations show that AttackGAN can successfully evade state-of-the-art IDS models, demonstrating the vulnerability of deep learning-based IDS to adversarial manipulations.

This study highlights the growing cybersecurity threat posed by adversarial learning and emphasizes the need for robust IDS defense mechanisms. The research contributes to understanding IDS weaknesses and improving adversarial resilience, urging the development of adversarial defense strategies to protect network security.

Vu, Ly, Quang Uy Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz. "Deep generative learning models for cloud intrusion detection systems." IEEE Transactions on Cybernetics 53, no. 1 (2022): 565-577.

The paper "Deep Generative Learning Models for Cloud Intrusion Detection Systems" by Ly Vu, Quang Uy Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz (2022) explores the application of deep generative learning models for intrusion detection in cloud computing environments.

Cloud-based networks face unique cybersecurity challenges due to their distributed nature, dynamic workloads, and evolving attack vectors. Traditional Intrusion Detection Systems (IDS) often struggle with handling high-dimensional, imbalanced, and complex cloud traffic data.

To address these challenges, the authors propose generative adversarial networks (GANs) and variational autoencoders (VAEs) to enhance IDS performance. These models generate realistic synthetic attack samples, balancing training datasets and improving detection accuracy. The GAN-based IDS learns attack patterns more effectively, while VAE-based models enhance anomaly detection by capturing hidden data distributions. Extensive experiments on real-world cloud datasets show that deep generative models outperform traditional machine learning approaches in detecting cyber threats, reducing false positives, and improving generalization to unknown attacks.

This study highlights the potential of deep generative learning in cybersecurity, demonstrating that GANs and VAEs can enhance cloud IDS by improving detection accuracy, scalability, and adaptability to evolving threats. The findings contribute to the development of next-generation cloud security solutions.

Dunmore, Aeryn, Julian Jang-Jaccard, Fariza Sabrina, and Jin Kwak. "A comprehensive survey of generative adversarial networks (GANs) in cybersecurity intrusion detection." IEEE Access 11 (2023): 76071-76094.

Dunmore, Jang-Jaccard, Sabrina, and Kwak (2023) present a comprehensive survey on the utilization of Generative Adversarial Networks (GANs) in cybersecurity, with a particular focus on intrusion detection systems (IDS). As cyberattacks become increasingly complex, traditional IDS approaches face significant challenges, including high false-positive rates, imbalanced datasets, limited ability to detect novel attacks, and difficulties in adapting to evolving threats. In response to these challenges, the authors explore how GANs, with their unique generative capabilities, offer promising solutions for enhancing IDS performance. The survey extensively reviews how GANs can assist in critical areas such as data augmentation, anomaly detection, and building adversarially resilient IDS models.

One of the key contributions of this survey is the categorization of GAN-based IDS applications into three main learning paradigms: supervised, unsupervised, and semi-

supervised models. Each category is discussed in detail, highlighting specific strengths and weaknesses. In supervised learning applications, GANs are used to generate realistic synthetic data that can balance class distributions, improving the training of classifiers. In unsupervised learning contexts, GANs help detect anomalies without the need for extensive labeled data, making them particularly valuable in environments where attack patterns continuously evolve. Semi-supervised approaches leverage a small amount of labeled data combined with large volumes of unlabeled data, with GANs helping bridge the gap by generating meaningful training instances. Through this categorization, the authors offer readers a clear understanding of the diverse strategies for applying GANs to intrusion detection tasks.

Furthermore, Dunmore et al. delve into the role of GANs in adversarial cybersecurity scenarios. They examine how GANs can both bolster IDS performance and be used to exploit vulnerabilities in IDS models through adversarial evasion and poisoning attacks. The dual nature of GANs in cybersecurity is emphasized, underlining the need for robust and resilient model designs. The authors also analyze a variety of publicly available datasets, such as NSL-KDD, CICIDS2017, and others, which are commonly used in GAN-based IDS research.

They discuss the evaluation metrics typically employed, including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC), providing a comprehensive guide for researchers to benchmark their systems.

In addition to reviewing current state-of-the-art implementations, the paper outlines emerging trends and future research directions in the field. Notably, the survey discusses the potential of hybrid models that combine GANs with other deep learning architectures like convolutional neural networks (CNNs) or recurrent neural networks (RNNs) to further enhance IDS capabilities.

The authors advocate for the incorporation of explainable AI (XAI) techniques to improve the interpretability and trustworthiness of GAN-based IDS models. Furthermore, the paper highlights the growing interest in federated learning as a distributed approach to building privacy-preserving, decentralized IDS systems that leverage GANs across multiple edge devices.

Overall, Dunmore et al. (2023) provide a valuable and thorough analysis of the current landscape of GAN applications in cybersecurity intrusion detection. Their work not only synthesizes key findings from the field but also offers strategic insights into the future of

AI-driven security solutions. This survey serves as an essential resource for researchers and practitioners aiming to harness the power of GANs to develop more adaptive, intelligent, and resilient intrusion detection systems.

Nie, Laisen, Yixuan Wu, Xiaojie Wang, Lei Guo, Guoyin Wang, Xinbo Gao, and Shengtao Li. "Intrusion detection for secure social internet of things based on collaborative edge computing: a generative adversarial network-based approach." IEEE Transactions on Computational Social Systems 9, no. 1 (2021): 134-145.

The paper "Intrusion Detection for Secure Social Internet of Things Based on Collaborative Edge Computing: A Generative Adversarial Network-Based Approach" by Laisen Nie, Yixuan Wu, Xiaojie Wang, Lei Guo, Guoyin Wang, Xinbo Gao, and Shengtao Li (2021) presents a GAN-based intrusion detection system (IDS) for the Social Internet of Things (SIoT) within a collaborative edge computing (CEC) environment.

The SIoT integrates social networking concepts into IoT devices, enhancing interconnectivity but also introducing new security vulnerabilities. Traditional cloud-based IDS solutions suffer from high latency and computational inefficiency, making them unsuitable for real-time intrusion detection in SIoT.To address these challenges, the authors propose a GAN-based IDS deployed in edge computing nodes, leveraging collaborative learning to enhance security across decentralized SIoT networks.

The generator synthesizes attack traffic to improve IDS training, while the discriminator accurately detects anomalous behaviors. Experimental results on real-world datasets show that this approach improves detection accuracy, reduces false positives, and enhances real-time threat response.

This study highlights the potential of GANs and edge computing for scalable, low-latency cybersecurity solutions in SIoT. The findings contribute to developing adaptive, intelligent IDS models that protect interconnected IoT environments from emerging cyber threats.

Ding, Hongwei, Leiyang Chen, Liang Dong, Zhongwang Fu, and Xiaohui Cui. "Imbalanced data classification: A KNN and generative adversarial networks-based hybrid approach for intrusion detection." Future Generation Computer Systems 131 (2022): 240-254.

The paper "Imbalanced Data Classification: A KNN and Generative Adversarial Networks-Based Hybrid Approach for Intrusion Detection" by Hongwei Ding, Leiyang Chen, Liang Dong, Zhongwang Fu, and Xiaohui Cui (2022) presents a novel hybrid model that combines K-Nearest Neighbors (KNN) and Generative Adversarial Networks (GANs) to address class imbalance in intrusion detection systems (IDS). Traditional IDS models struggle with imbalanced datasets, where minority-class attack instances are underrepresented, leading to poor detection of rare but critical cyber threats.

To tackle this issue, the authors propose a GAN-based data augmentation strategy that generates realistic attack samples, balancing the dataset. The KNN algorithm is then employed as a classifier, benefiting from the enhanced training data to improve detection accuracy. Experimental results on benchmark IDS datasets demonstrate that this hybrid approach outperforms conventional machine learning models, achieving higher precision, recall, and F1-scores, especially for minority-class intrusions.

This study highlights the effectiveness of GANs in cybersecurity, proving that synthetic data generation combined with traditional classifiers can significantly enhance intrusion detection performance. The findings contribute to developing more robust IDS solutions capable of handling real-world imbalanced network traffic scenarios.

Wu, Yixuan, Laisen Nie, Shupeng Wang, Zhaolong Ning, and Shengtao Li. "Intelligent intrusion detection for internet of things security: A deep convolutional generative adversarial network-enabled approach." IEEE Internet of Things Journal 10, no. 4 (2021): 3094-3106.

The paper "Intelligent Intrusion Detection for Internet of Things Security: A Deep Convolutional Generative Adversarial Network-Enabled Approach" by Yixuan Wu, Laisen Nie, Shupeng Wang, Zhaolong Ning, and Shengtao Li (2021) proposes a Deep Convolutional

Generative Adversarial Network (DCGAN)-based Intrusion Detection System (IDS) for Internet of Things (IoT) security. IoT networks face significant cybersecurity threats due to

their decentralized nature, resource constraints, and vulnerability to attacks. Traditional IDS models struggle with imbalanced datasets, high false positive rates, and ineffective detection of unknown attacks.

To address these issues, the authors develop a DCGAN-based IDS where the generator creates synthetic attack samples to enhance training data, and the discriminator learns to detect real threats more effectively. The use of deep convolutional layers allows the model to extract high-level features from network traffic, improving classification accuracy.

Experimental results on benchmark IoT datasets show that DCGAN-IDS outperforms traditional machine learning and deep learning approaches, achieving higher detection rates, better generalization, and improved robustness against adversarial attacks.

This study demonstrates the potential of deep generative learning in strengthening IoT security, contributing to the development of more adaptive, scalable, and intelligent IDS solutions for real-world IoT environments.

CHAPTER-3

SYSTEM STUDY

3.1 EXISTING SYSTEM

Existing network intrusion detection systems (NIDS) primarily rely on two main methods: signature-based detection and anomaly-based detection. Signature-based detection works by identifying known attack patterns using predefined signatures or rules. While highly effective against previously recorded threats, this method fails to detect new or zero-day attacks, making it less adaptable to evolving cyber threats. Additionally, maintaining an up-to-date database of attack signatures requires continuous updates and significant computational resources.

On the other hand, anomaly-based detection uses machine learning and statistical models to establish a baseline of normal network behavior. Any deviation from this baseline is flagged as a potential intrusion. While this method can detect novel attacks, it suffers from a high false positive rate, as legitimate network variations may be misclassified as attacks. Moreover, training anomaly-based models requires extensive labeled datasets, which may not always be available or representative of real-world network traffic.

To overcome these limitations, hybrid approaches combining signature-based and anomaly-based techniques have been developed. These methods enhance detection accuracy by leveraging the strengths of both techniques. However, the increasing complexity of cyber threats necessitates more advanced solutions, such as artificial intelligence and deep learning-based approaches, to improve detection efficiency and adaptability.

DISADVANTAGES

- Ineffective Against Zero-Day Attacks – It can only detect known threats, making it useless against new or evolving attacks.
- High Maintenance – Requires frequent updates to the signature database, increasing operational costs.
- Slow Response to Emerging Threats – New attack signatures take time to be identified and integrated into databases.

- Computational Overhead – Continuously comparing traffic with signature databases consumes significant processing power.
- Evasion Techniques – Attackers can slightly modify known attack patterns to bypass detection.
- High False Positive Rate – Unusual but legitimate activities may be flagged as attacks, causing unnecessary alerts.
- Extensive Training Data Required – Requires large, labeled datasets to train accurate models.
- Difficult to Define 'Normal' Behavior – Network behavior varies across environments, making it hard to set a proper baseline.
- Slow Adaptation – Models may require retraining when network behavior changes significantly.
- Computationally Intensive – Training and maintaining machine learning models demand high processing power.
- Encrypted Traffic Challenges – Struggles to inspect encrypted data, limiting detection capabilities.
- Scalability Issues – High-speed networks generate massive traffic, making real-time analysis difficult.
- Resource Consumption – Requires significant memory, CPU, and storage for log analysis and pattern matching.
- Limited Context Awareness – Cannot always distinguish between a real attack and a harmless network anomaly.
- Bypass Risks – Advanced attackers use polymorphic and metamorphic malware to avoid detection.

3.2 PROPOSED SYSTEM

To address the limitations of traditional Network Intrusion Detection Systems (NIDS), this proposed system integrates artificial intelligence (AI) and deep learning techniques to enhance detection accuracy, adaptability, and efficiency. Unlike conventional signature-based and anomaly-based approaches, which have significant drawbacks such as high false positive rates and inability to detect zero-day attacks, the proposed system leverages a hybrid AI-driven approach that combines multiple detection techniques for improved threat identification and mitigation. The proposed system employs a hybrid detection mechanism that integrates signature-based, anomaly-based, and deep learning-based models to offer a more robust and dynamic intrusion detection framework.

This combination enhances the system's ability to detect both known and unknown threats effectively. The key components of the proposed system include signature-based detection with automated updates, anomaly-based detection with adaptive learning, deep learning for advanced threat detection, real-time traffic analysis with edge computing, and blockchain-based threat intelligence sharing.

The signature-based detection component maintains an up-to-date repository of attack signatures but enhances it with AI-driven automation. Machine learning algorithms automatically extract and update signatures from new threat intelligence sources, reducing manual intervention and ensuring faster responses to emerging threats. Anomaly-based detection utilizes unsupervised and semi-supervised machine learning models that continuously learn from network traffic patterns, dynamically updating the normal behavior baseline and reducing false positive rates by distinguishing between benign anomalies and actual attacks.

Deep learning-based detection leverages Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) networks, to analyze complex attack patterns, improving detection accuracy for zero-day attacks and sophisticated threats like Advanced Persistent Threats (APTs) and polymorphic malware.

Real-time traffic analysis is enhanced with edge computing, which processes data closer to the source, enabling immediate threat detection and mitigation. This distributed computing approach reduces latency, enhances scalability, and ensures efficient performance even in high-speed networks.

Additionally, a decentralized and immutable blockchain ledger is employed for secure threat intelligence sharing across different NIDS installations, ensuring that all connected systems benefit from real-time threat updates without relying on a centralized authority.

The system follows a modular architecture consisting of a traffic monitoring module that captures and pre-processes network traffic, a feature extraction module that identifies key features using statistical and deep learning techniques, a hybrid detection engine that combines signature-based, anomaly-based, and deep learning models to classify threats, a response mechanism that takes automated actions such as alert generation, packet dropping, or firewall rule updates, and a threat intelligence database that stores attack signatures, anomaly patterns, and deep learning model updates.

The proposed system offers multiple advantages, including improved accuracy through AI-based models that reduce false positives and false negatives, enhanced zero-day attack detection with deep learning models that identify novel attack patterns, automated adaptation that allows the system to continuously learn and update itself with minimal human intervention, scalability through edge computing that ensures efficient performance in large-scale networks, secure threat sharing enabled by blockchain technology that prevents tampering of shared intelligence data, and reduced computational overhead through distributed processing that optimizes resource utilization. In conclusion, the proposed AI-driven NIDS represents a significant advancement over traditional detection mechanisms.

By integrating signature-based methods with anomaly-based detection and deep learning techniques, the system enhances security, reduces response time, and adapts to evolving threats. The incorporation of blockchain technology and edge computing further strengthens the system's effectiveness, making it an intelligent, self-improving, and scalable solution for modern cybersecurity challenges.

ADVANTAGES

- GANs can generate synthetic data to mimic new and unknown attack patterns, improving the detection of emerging threats.
- The adversarial training process improves the model's ability to distinguish between normal and malicious traffic, reducing the number of false alarms.

- GANs help create synthetic data to address the issue of imbalanced datasets, improving model training and performance.
- GAN-based IDS models can adapt to new attack patterns autonomously without frequent manual retraining.
- The system can generalize better to new, unseen attack types, ensuring a more robust and scalable detection system.
- GAN-based models can be more resilient to adversarial attacks aimed at manipulating IDS systems.
- GANs can be applied to supervised, unsupervised, and semi-supervised learning, providing flexibility for different network environments.
- The system can better detect irregular network behaviors or novel attack types, even in environments with limited labeled data.
- GAN-based IDS models can achieve higher accuracy, precision, recall, and F1-score metrics compared to traditional IDS models.
- GAN-based IDS can scale effectively to large networks and cloud environments without compromising performance.
- The autonomous learning ability of GANs reduces the need for manual data labeling and system retraining.
- GAN-based IDS can detect threats in real time by continuously learning from network traffic and evolving attack patterns.
- GANs are adaptable to various attack types, such as denial of service (DoS), malware, and phishing attacks.
- In environments where labeled data is scarce, GANs help detect anomalous traffic without requiring large datasets.
- GANs can generate synthetic data to overcome the scarcity of attack data, especially for rare or novel attacks.
- GAN-based IDS can be easily integrated into existing security infrastructures, enhancing the detection capabilities of current IDS systems.

- By generating diverse attack samples, GAN-based IDS models achieve a more balanced detection performance, improving both precision and recall.
- GANs can be incorporated into federated learning frameworks, enabling distributed learning while preserving data privacy.
- GANs can be applied to a wide range of domains, including IoT, cloud networks, and industrial control systems.
- GAN-based IDS systems are designed to adapt to new threats, making them future-proof as the cyber threat landscape evolves.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- CPU type : AMD PRO A4-4350B R4 processor
- Ram size : 4 GB
- Hard disk capacity : 512 GB

4.2 SOFTWARE REQUIREMENT

- Operating System : Windows 10
- Language : Python (Flask or Django)
- Tool : My SOL or MangoDB

SOFTWARE DESCRIPTION

- A high-performance GPU, such as the NVIDIA RTX 3090, is necessary to accelerate the training and inference processes of deep learning models, particularly GANs, enabling the system to process large volumes of data efficiently.
- Minimum of 32GB RAM is required to manage large-scale network datasets and ensure smooth operation during real-time data processing and model training without memory bottlenecks.
- SSD storage with at least 1TB of capacity ensures fast data retrieval and processing, especially when dealing with large network traffic datasets, which is critical for maintaining real-time performance. ensures fast data retrieval and processing, especially when dealing with large network traffic datasets, which is critical for maintaining real-time performance.
- A stable and high-speed internet connection is essential for monitoring network traffic in real time, allowing the IDS to process incoming data streams quickly and identify potential intrusions without delay.

- Python is the programming language of choice for implementing the GAN-based IDS, as it offers extensive libraries and frameworks for machine learning and deep learning, making model development and integration straightforward.
- TensorFlow or PyTorch, both powerful deep learning frameworks, are used to implement and train GAN models, providing the tools necessary to develop the generator and discriminator components for the IDS.
- Scapy, a Python-based tool, is used for generating custom network traffic and analyzing data flows, which is essential for testing the IDS's ability to detect various attack scenarios and anomalies in the network.
- Wireshark is a network protocol analyzer that allows the IDS to capture and inspect network packets in detail, helping to identify anomalies and understand attack characteristics in the network traffic.
- Flask or Django, Python web frameworks, are utilized to create an interactive, real-time web dashboard that enables administrators to monitor network activity, view detected anomalies, and manage alerts through an intuitive user interface.
- MySQL or MongoDB databases are employed for storing intrusion detection logs, results, and historical data, ensuring efficient data management and retrieval for further analysis, monitoring trends, and improving the IDS over time.

CHAPTER 5

MODULE DESCRIPTION

MODULE DESCRIPTION EXPLANATION

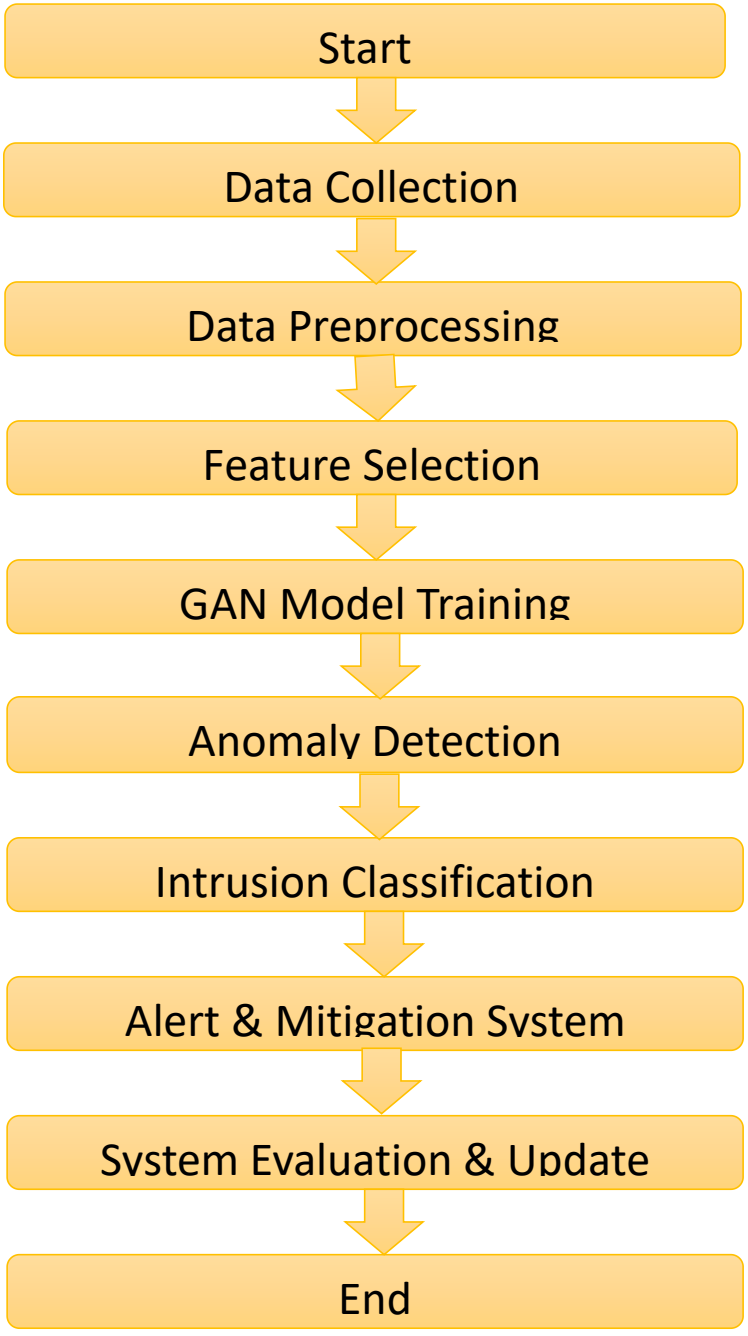


Fig 5.1 Module Description

5.1 Data Collection

Data collection is the first crucial step in developing a Generative Adversarial Network (GAN)-based Intrusion Detection System (IDS). The success of the system heavily depends on the quality and comprehensiveness of the collected data. In network security, data typically includes a variety of parameters, such as traffic flows, protocols, timestamps, packet sizes, and source/destination IP addresses. These attributes are essential to understanding the normal patterns of network activity and identifying any deviations that may signal potential intrusions. Network traffic data can be collected from multiple sources, including firewalls, intrusion detection systems (IDS), or public datasets like NSL-KDD, KDD99, or CIC-IDS2017, which contain both normal network behavior and various types of cyberattacks. These datasets serve as a valuable resource for training and evaluating machine learning models by providing labeled examples of both benign and malicious network activities.

Real-time data collection plays a critical role in capturing the dynamic nature of network traffic. Unlike static datasets, real-time data reflects the changing conditions of a network environment, ensuring that the IDS can detect attacks as they unfold. Such data is often gathered directly from network devices or through network monitoring tools that track traffic patterns in real time. For example, logs from firewalls or IDS can provide detailed records of network traffic, highlighting the frequency, duration, and source of suspicious connections. Collecting this data requires specialized tools and techniques to ensure that all relevant details are captured without overburdening the system with unnecessary information.

The collected data is typically stored in structured formats like CSV files, databases, or cloud storage, which make it easier to manage, preprocess, and analyze. This structured format ensures that the raw data is organized in a way that makes it accessible for training machine learning models. Data storage must also be scalable, as the amount of network traffic can grow significantly over time. Efficient data storage practices not only ensure data integrity but also contribute to the overall performance of the intrusion detection system by reducing processing time during model training.

The main objective of the data collection phase is to gather a representative sample of both normal and malicious traffic. This diversity allows the GAN model to learn the nuances of legitimate network behavior while also understanding various attack patterns. A well-curated dataset ensures that the trained model has a high ability to generalize to new attacks.

Additionally, data collection plays a pivotal role in identifying attack vectors, such as Denial of Service (DoS), Distributed Denial of Service (DDoS), or malware attacks, which may have different signatures based on the nature of the attack. The collected data must also be balanced, meaning it should contain a mix of normal traffic and attack traffic to avoid the model being biased toward classifying everything as normal.

In summary, the data collection phase is foundational to building an effective GAN-based IDS. The data serves as the cornerstone upon which the system's performance will be built. By ensuring that the data is comprehensive, diverse, and representative of real-world scenarios, the system can learn to differentiate between legitimate and malicious network activities with high accuracy.

5.2 Data Preprocessing

Data preprocessing is an essential step in the development of any machine learning system, including a Generative Adversarial Network (GAN)-based Intrusion Detection System (IDS). Raw network data is often messy, noisy, and contains several inconsistencies that could undermine the performance of the model. Therefore, preprocessing techniques must be applied to clean and organize the data, ensuring that the model receives high-quality inputs that will improve its learning capabilities. The quality of the data used in training directly affects the accuracy and robustness of the system in detecting cyber threats.

One of the primary challenges in network traffic data is the presence of missing values, duplicate records, and irrelevant features. Missing data can occur due to errors in data collection, transmission failures, or incomplete logging. These gaps need to be handled by imputation techniques, such as replacing missing values with the mean or median of the feature, or using more sophisticated methods like K-nearest neighbors or regression models. On the other hand, duplicate entries can skew the learning process and lead to overfitting. Therefore, identifying and removing redundant data is necessary to ensure that the model does not rely on repeated information that does not contribute to new insights.

Another challenge is dealing with categorical variables. Many network traffic features, such as protocol type or service name, are represented as categorical variables.

Machine learning algorithms typically require numerical data, so these categorical variables must be transformed into a suitable format. Common encoding techniques, such as

One-Hot Encoding or Label Encoding, are employed to convert categorical features into numerical values without losing the intrinsic information they carry.

Feature extraction is a critical aspect of data preprocessing in the context of network traffic data. Raw network data may contain a vast number of features, but not all of them are useful for detecting intrusions. Features such as packet size, duration of the connection, protocol type, and IP addresses can provide valuable insights, while other features may be irrelevant and can increase the computational complexity. Principal Component Analysis (PCA) or Min-Max scaling are commonly used techniques to normalize the data, making it easier for the GAN model to learn. Feature extraction also involves aggregating packet-level data into flow-based records to provide higher-level information, which can help reduce the complexity of the system and improve processing efficiency.

Preprocessing also addresses imbalanced datasets, a common issue in intrusion detection systems. In many real-world scenarios, malicious traffic is much rarer than normal traffic, leading to a skewed dataset. This imbalance can cause the model to become biased toward predicting normal traffic and ignoring attacks. To mitigate this, techniques such as oversampling (e.g., duplicating attack samples) or undersampling (e.g., reducing normal traffic data) can be applied. Additionally, Synthetic Minority Over-sampling Technique (SMOTE) can be used to generate synthetic attack samples, ensuring that the model is exposed to a balanced representation of both attack and normal traffic.

The goal of data preprocessing is to clean, normalize, and structure the data in a way that maximizes the performance of the GAN-based IDS. Effective preprocessing ensures that the model is not overwhelmed by noise or irrelevant information, allowing it to focus on identifying patterns and anomalies that are critical for accurate intrusion detection.

5.3 Feature Selection

Feature selection is a crucial step in the development of a GAN-based Intrusion Detection System (IDS), as it directly influences the model's performance, efficiency, and accuracy.

In network traffic datasets, there are often numerous features, many of which may be redundant, irrelevant, or noisy. By selecting the most important features, the system can focus on the data that has the greatest predictive power, reducing the computational load and improving detection capabilities.

The first step in feature selection is to understand the relationship between the features and the target variable, which in this case is whether a network activity is benign or malicious. Irrelevant features can introduce noise into the learning process, making it harder for the model to distinguish between normal and abnormal behaviors. By eliminating these irrelevant features, the model becomes more efficient and better at generalizing from the training data to new, unseen traffic.

Redundant features are also problematic because they may provide the same information multiple times, which can lead to overfitting and increased computational costs. For example, multiple features that represent the same concept—such as multiple representations of packet size—do not add value to the model and should be removed.

Several techniques are used for feature selection, each with its strengths and limitations. Recursive Feature Elimination (RFE) is a commonly used method where features are recursively removed, and the model is trained to assess the importance of each feature. RFE works by ranking features based on their importance, allowing the system to eliminate the least relevant ones. This process is repeated until only the most essential features remain. Another popular technique is Mutual Information (MI), which measures the dependency between features. Features that have a strong relationship with the target variable but are less correlated with each other are retained. This method helps capture non-linear relationships between features that may not be evident in linear models.

Random Forest Importance is another widely used technique for feature selection in intrusion detection systems. Random forests are ensemble models that generate multiple decision trees based on random subsets of the data. Each tree is built using different features, and the final decision is made by aggregating the outputs of all the trees.

By analyzing the importance scores of the features used in the trees, we can determine which features contribute most to the decision-making process.

Features with low importance scores can be safely eliminated, thereby improving the model's performance and reducing complexity.

In the context of network intrusion detection, important features typically include packet size, protocol type, source and destination IP addresses, and connection duration. These features are critical for identifying unusual patterns in network traffic that may indicate an intrusion, such as a Denial of Service (DoS) attack or malware activity. By focusing on these key features, the GAN-based IDS can improve its accuracy and reduce the number of false positives.

Feature selection not only helps in reducing the dimensionality of the dataset but also plays a key role in reducing the computational complexity of the model. With fewer features to process, the model can be trained faster and will be more efficient during real-time anomaly detection. Furthermore, by focusing on the most relevant features, the system can make more accurate predictions, which is crucial in cybersecurity applications where timely and precise detection is vital.

5.4 Generative Adversarial Network (GAN) Model Training

The training of the Generative Adversarial Network (GAN) model is one of the most critical stages in building an AI-based Intrusion Detection System (IDS). GANs consist of two components: the Generator and the Discriminator, which work in opposition to each other in an adversarial manner to improve the system's ability to detect anomalies and attacks. The training process is iterative and involves continuous improvements to both components, ensuring that the final model can effectively differentiate between normal and malicious network traffic.

The Generator is responsible for creating synthetic data that mimics real attack patterns. This data is essential because, in many cases, it is difficult to obtain a sufficient amount of attack data, particularly for zero-day attacks or new types of cyber threats. The Generator learns to create attack patterns by studying the normal behavior of network traffic, using the patterns it learns to produce synthetic instances of attack traffic that resemble real-world scenarios.

The goal is for the Generator to produce attack data that is realistic enough to fool the Discriminator. The Discriminator, on the other hand, is tasked with differentiating between real and synthetic data. It receives both real network traffic (from the dataset) and synthetic traffic (generated by the Generator) and learns to classify the data as normal or malicious. The Discriminator uses probability scores to evaluate the authenticity of the traffic and becomes more accurate over time as it receives more feedback from the Generator. The training process is designed to enhance the Discriminator's ability to distinguish between benign and malicious traffic while simultaneously improving the Generator's capacity to create more convincing synthetic attack patterns.

The adversarial process between the Generator and the Discriminator helps improve both components simultaneously. The Generator attempts to create more realistic attack data to trick the Discriminator, while the Discriminator constantly refines its ability to detect fake data. Over time, this adversarial training leads to a robust and dynamic model that can effectively identify a wide range of attacks, including those that are novel or previously unseen.

A critical aspect of training a GAN-based IDS is the data diversity. Since the Generator creates synthetic attack data, it must be trained on a wide range of attack types, from simple DoS attacks to more complex malware or phishing attacks. This ensures that the model is capable of detecting various forms of cyber threats. The training data should also be representative of normal traffic patterns to ensure that the system can accurately classify legitimate network activity.

The training process involves several epochs, during which the Generator and Discriminator update their parameters to improve performance. This process requires careful tuning of hyperparameters, such as learning rates, batch sizes, and the number of training iterations, to achieve optimal results. Additionally, training GANs can be computationally intensive, requiring substantial resources in terms of processing power and memory. However, the benefits of having a system that can generate realistic attack scenarios and accurately classify traffic make it a highly effective solution for network intrusion detection.

CHAPTER-6

SYSTEM TESTING

6.1 UNIT TESTING

Step 1: Identify Units to be Tested

The first step in the unit testing process is to identify the individual units or components of the system that need to be tested. For the AI-based network intrusion detection system, some of the units that may be tested include:

- Data preprocessing module
- Feature extraction module
- GAN model
- Detection algorithm
- Data loading and processing

Step 2: Write Test Cases

Once the units to be tested have been identified, the next step is to write test cases for each unit. Test cases should cover different scenarios and edge cases, including:

- Valid input data
- Invalid input data
- Edge cases (e.g., empty data, missing values)
- Error handling

Step 3: Execute Test Cases

After writing the test cases, the next step is to execute them. This involves running the test cases and comparing the actual output with the expected output.

Step 4: Analyze Results

After executing the test cases, the next step is to analyze the results. This involves identifying any failures or errors and debugging the code to fix the issues.

Step 5: Repeat the Process

The unit testing process is iterative, and it's essential to repeat the process until all units have been thoroughly tested.

6.2 INTEGRATION TESTING

Step 1: Identify Components to be Integrated

The first step in the integration testing process is to identify the components that need to be integrated and tested together. For the AI-based network intrusion detection system, some of the components that may be integrated include:

- Data preprocessing module
- Feature extraction module
- GAN model
- Detection algorithm
- Data loading and processing

Step 2: Develop Test Cases

Once the components to be integrated have been identified, the next step is to develop test cases that cover different scenarios and edge cases. Test cases should include:

- Valid input data
- Invalid input data
- Edge cases (e.g., empty data, missing values)
- Error handling

Step 3: Execute Test Cases

After developing the test cases, the next step is to execute them. This involves integrating the components and testing them together.

Step 4: Analyze Results

After executing the test cases, the next step is to analyze the results. This involves identifying any integration-related issues and debugging the code to fix the issues.

Step 5: Repeat the Process

The integration testing process is iterative, and it's essential to repeat the process until all components have been thoroughly integrated and tested.

6.3 SYSTEM TESTING

Step 1: Define System Testing Objectives

The first step in the system testing process is to define the objectives of the testing. This includes identifying the types of testing to be performed, the scope of the testing, and the criteria for evaluating the system's performance.

Step 2: Develop Test Cases

The next step is to develop test cases that cover different scenarios and edge cases. Test cases should include:

- Valid input data
- Invalid input data
- Edge cases (e.g., empty data, missing values)
- Error handling
- Security testing
- Performance testing

Step 3: Execute Test Cases

After developing the test cases, the next step is to execute them. This involves testing the entire system, including all components and interactions.

Step 4: Analyze Results

After executing the test cases, the next step is to analyze the results. This involves identifying any issues or defects and debugging the code to fix the issues.

Step 5: Repeat the Process

The system testing process is iterative, and it's essential to repeat the process until the system meets the required standards.

6.4 PERFORMANCE TESTING

Step 1: Identify Performance Testing Objectives

The first step in the performance testing process is to identify the objectives of the testing. This includes determining what aspects of the system's performance need to be tested, such as response time, throughput, and resource utilization.

Step 2: Develop Performance Testing Plan

The next step is to develop a performance testing plan that outlines the approach, tools, and metrics to be used. This plan should include:

- Test environment: Description of the test environment, including hardware, software, and network configurations.
- Test data: Description of the test data to be used, including volume, complexity, and realism.
- Test scenarios: Description of the test scenarios to be executed, including user interactions, data inputs, and expected outputs.
- Performance metrics: Description of the performance metrics to be measured, such as response time, throughput, and resource utilization.

Step 3: Prepare Test Environment

The next step is to prepare the test environment, including:

- Configuring hardware and software: Ensuring that the test environment is configured to simulate real-world conditions.
- Loading test data: Loading the test data into the system to simulate real-world usage.
- Configuring monitoring tools: Configuring monitoring tools to measure performance metrics.

Step 4: Execute Performance Tests

The next step is to execute the performance tests, including:

- Load testing: Testing the system's performance under a large volume of users or data.

- Stress testing: Testing the system's performance under extreme conditions, such as high traffic or resource constraints.
- Endurance testing: Testing the system's performance over an extended period to ensure that it can sustain a high level of performance.

Step 5: Analyze Results

After executing the performance tests, the next step is to analyze the results, including:

- Identifying bottlenecks: Identifying any bottlenecks or performance issues that need to be addressed.
- Comparing results to benchmarks: Comparing the results to benchmarks or industry standards to determine if the system meets performance expectations.
- Identifying areas for optimization: Identifying areas for optimization to improve the system's performance.

Step 6: Optimize System Performance

The final step is to optimize the system's performance based on the results of the performance testing, including:

- Tuning system configurations: Tuning system configurations to improve performance.
- Optimizing code: Optimizing code to improve performance.
- Upgrading hardware: Upgrading hardware to improve performance.

6.5 SECURITY TESTING

Step 1: Identify Security Testing Objectives

The first step in the security testing process is to identify the objectives of the testing. This includes determining what aspects of the system's security need to be tested, such as vulnerability assessment, penetration testing, and compliance testing.

Step 2: Develop Security Testing Plan

The next step is to develop a security testing plan that outlines the approach, tools, and metrics to be used. This plan should include:

- Threat modeling: Identifying potential threats and vulnerabilities in the system.

- Test scenarios: Description of the test scenarios to be executed, including attack vectors and expected outcomes.
- Security metrics: Description of the security metrics to be measured, such as vulnerability density and patch coverage.

Step 3: Conduct Vulnerability Assessment

The next step is to conduct a vulnerability assessment to identify potential vulnerabilities in the system. This includes:

- Network scanning: Scanning the network for open ports and services.
- System scanning: Scanning the system for vulnerabilities and weaknesses.
- Application scanning: Scanning the application for vulnerabilities and weaknesses.

Step 4: Conduct Penetration Testing

The next step is to conduct penetration testing to simulate real-world attacks on the system. This includes:

- Network penetration testing: Testing the network for vulnerabilities and weaknesses.
- Application penetration testing: Testing the application for vulnerabilities and weaknesses.
- Social engineering testing: Testing the system's users for susceptibility to social engineering attacks.

Step 5: Analyze Results

After conducting the security testing, the next step is to analyze the results, including:

- Identifying vulnerabilities: Identifying vulnerabilities and weaknesses in the system.
- Prioritizing remediation: Prioritizing remediation efforts based on the severity of the vulnerabilities.
- Developing remediation plan: Developing a plan to remediate the vulnerabilities and weaknesses.

Step 6: Remediate Vulnerabilities

The final step is to remediate the vulnerabilities and weaknesses identified during the security testing, including:

- Patching vulnerabilities: Patching vulnerabilities and weaknesses in the system.
- Implementing security controls: Implementing security controls to prevent future vulnerabilities.
- Conducting follow-up testing: Conducting follow-up testing to ensure that the vulnerabilities have been remediated.

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 CODE EXPLANATION

The fire detection system is designed using Convolutional Neural Networks (CNNs) and Region-based Convolutional Neural Networks (R-CNNs) to identify and localize fire in images and video frames, which is critical for applications such as real-time surveillance and safety monitoring. Python is used as the main programming language, utilizing libraries like TensorFlow, Keras, OpenCV, Matplotlib, and Sklearn to facilitate model building, image processing, and evaluation.

The development environment is Google Colab, selected for its convenient GPU support and the ability to directly access datasets stored on Google Drive. The dataset is stored on Google Drive, and to access it within Colab, the drive.mount('/content/drive') command is used to mount the drive, enabling the script to read images from the specified path.

Images are organized into training and testing folders, each containing subfolders labeled ‘Fire’ and ‘Non_Fire,’ which simplifies the process of automatic labeling during training through the use of image generators.

Image preprocessing is handled with TensorFlow’s ImageDataGenerator, which resizes all input images to 64 by 64 pixels and rescales the pixel values to the range of 0 to 1, aiding in normalization and improving model training stability.

The training images are augmented with transformations such as rotation, zoom, and horizontal flipping to increase the diversity of the dataset and help prevent overfitting, improving the model’s ability to generalize to unseen data. The CNN architecture begins with several convolutional layers that apply multiple filters to extract feature maps, capturing spatial hierarchies in the images related to fire patterns, shapes, and textures.

Each convolutional layer is followed by a max-pooling layer which reduces the spatial dimensions, decreasing computational cost and controlling overfitting by summarizing the most salient features. After the convolution and pooling layers, the feature maps are flattened into a one-dimensional vector which is then fed into fully connected dense layers, enabling the network to learn complex combinations of the extracted features for classification.

The final dense layer uses a sigmoid activation function to output a probability indicating the presence or absence of fire in the input image, suitable for the binary classification task. The loss function employed is binary cross-entropy, which measures the difference between the predicted probabilities and the actual labels, guiding the optimizer to minimize this loss during training.

The Adam optimizer is selected for its efficiency and adaptive learning rate, allowing the model to converge faster while maintaining good accuracy. Training occurs over multiple epochs, with batches of images passed through the network to update the weights iteratively, gradually improving the model's ability to correctly classify fire and non-fire images. The model's performance is evaluated using accuracy, precision, recall, and F1-score metrics calculated on a separate validation set to ensure the model generalizes well beyond the training data.

To enhance detection capabilities, the project integrates Region-based CNN (R-CNN) techniques that not only classify images but also locate fire regions by proposing candidate bounding boxes within the input frames.The R-CNN method uses selective search to generate region proposals, which are then fed into the CNN to classify whether each proposed region contains fire. This localization aspect allows the system to draw bounding boxes around detected fire areas, providing visual alerts and precise detection.

Transfer learning is employed by fine-tuning a pre-trained CNN model, such as ResNet or VGG, leveraging learned features from large-scale datasets to improve fire detection accuracy with fewer training images. The input images for R-CNN are resized consistently to meet the requirements of the pre-trained model, and the last few layers are retrained on the fire dataset to adapt the model to the specific task.

Data augmentation for the R-CNN pipeline is carefully applied to maintain the integrity of the bounding box annotations, ensuring that the region proposals remain accurate after transformations like scaling or flipping. During training, early stopping is implemented to prevent overfitting, monitoring validation loss and halting training once performance plateaus or worsens. After training, the models are tested on unseen images and video frames to verify their ability to detect fire under different conditions, including varying lighting, backgrounds, and fire sizes. OpenCV functions are utilized to preprocess video frames in real time and to draw bounding boxes and labels on detected fire regions during live surveillance.

The detection threshold is calibrated to balance false positives and false negatives, prioritizing safety by minimizing missed fire detections while reducing false alarms. The final system can be deployed on edge devices or cloud platforms to monitor environments such as forests, factories, or residential areas, triggering alarms or notifications when fire is detected. The codebase is modularized, separating data loading, preprocessing, model building, training, and evaluation into different functions to improve readability and maintainability.

Logging and checkpoint saving are incorporated to record training progress and store the best-performing model weights for future use. Visualization tools like Matplotlib plot training and validation accuracy and loss curves to help diagnose underfitting or overfitting and to fine-tune hyperparameters. Hyperparameter tuning, including adjustments to learning rate, batch size, and number of epochs, is performed experimentally to optimize model performance.

The dataset itself is balanced to include approximately equal numbers of fire and non-fire images, preventing bias towards either class during training. Careful handling of class imbalance is critical since fire images are less common but more crucial to detect accurately. The overall pipeline begins with data ingestion, followed by preprocessing, augmentation, model training, evaluation, and finally deployment-ready inference functions that take live input and provide fire detection results promptly.

The R-CNN's selective search algorithm enhances accuracy by narrowing down regions likely to contain fire, reducing computation by avoiding exhaustive sliding window searches. Both CNN and R-CNN models output confidence scores which are thresholded to decide detection, and these thresholds can be adjusted based on application needs. The fire detection system’s real-time capability is achieved by optimizing model complexity and using GPU acceleration, ensuring fast frame processing in video streams. Integration with alert systems, such as SMS or email notifications, is possible through APIs to automatically inform safety personnel upon fire detection.

The code also includes error handling to manage missing files, incorrect paths, or corrupted images, improving robustness during training and inference. Image normalization techniques ensure consistent input data distribution, vital for stable training and accurate predictions. The CNN layers’ filter sizes and number are carefully chosen to capture relevant features without excessive complexity that might lead to overfitting. Batch normalization

layers are optionally added to speed up training and stabilize activations across layers. Dropout regularization is applied in dense layers to reduce the risk of overfitting by randomly disabling neurons during training.

Model checkpoints save intermediate weights periodically to safeguard progress and allow resuming training from the latest point after interruptions. The fire detection project also implements evaluation on confusion matrices to understand model errors, providing insight into true positive, true negative, false positive, and false negative counts. Threshold tuning involves adjusting the decision boundary to trade off between sensitivity and specificity according to deployment requirements. The R-CNN model's bounding boxes undergo non-maximum suppression to eliminate redundant overlapping detections and focus on the most confident regions.

Real-world fire detection challenges such as smoke, reflections, or other bright objects are considered, and data diversity is enhanced to improve robustness against such false positives. The use of synthetic data augmentation helps increase the variety of fire appearances, enhancing the model's adaptability. The CNN's convolutional filters are visualized post-training to interpret what features the network learned, such as edges, colors, or shapes characteristic of fire.

Grad-CAM or saliency maps are used to explain model predictions by highlighting image regions most influential in the classification decision. The system's scalability is planned by allowing batch inference and model quantization for deployment on low-power devices. Performance benchmarks include frames per second (FPS) processing rate, memory usage, and detection latency, ensuring the solution meets real-time constraints. The code uses Python's argparse or similar libraries to allow configurable parameters at runtime, such as dataset paths, batch size, and number of epochs. The modular design permits easy swapping of backbone CNN architectures, facilitating experimentation with more advanced networks. The training process includes monitoring GPU usage and memory to optimize resource allocation.

APPENDIX

SOURCE CODE

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, _, filenames in os.walk('/content/DATASET'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"

# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

train_df=pd.read_csv('/content/DATASET/Train_data.csv')

test_df=pd.read_csv('/content/DATASET/Test_data.csv')

train_df.head(4)

train_df.shape

test_df.head(4)

test_df.shape

for col in ['protocol_type', 'service', 'flag']:

    print(f"Unique values in {col}: {train_df[col].nunique()}")

    print(train_df[col].value_counts()[:10])  # Show top 10

    print("-" * 50)
```

```
train_df.info()

test_df.info()

# Check for missing values

print(train_df.isnull().sum())

print(test_df.isnull().sum())

# Check duplicate rows in train and test dataset

print(f"Duplicate Rows in Train: {train_df.duplicated().sum()}")

print(f"Duplicate Rows in Test: {test_df.duplicated().sum()}")

test_df = test_df.drop_duplicates()

print("New Test Shape:", test_df.shape)

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))

sns.countplot(x=train_df['class'],

order=train_df['class'].value_counts().index, palette="coolwarm")

plt.xticks(rotation=90)

plt.title("Attack Type Distribution")

plt.show()

# Print attack class distribution

print(train_df['class'].value_counts(normalize=True) * 100)

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder
```

```
# Copy the dataframe
df_encoded = train_df.copy()
# Identify categorical columns
categorical_cols = ['protocol_type', 'service', 'flag', 'class']
# Apply Label Encoding to categorical columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
    label_encoders[col] = le  # Store the encoder for inverse transformation if needed later
# Select only numeric columns for correlation
numeric_df = df_encoded.select_dtypes(include=[np.number])
# Compute correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(numeric_df.corr(), cmap="coolwarm", annot=False)
plt.title("Feature Correlation Heatmap (Encoded)")
plt.show()
columns_to_drop = [
    "num_outbound_cmds",
    "is_host_login",
    "dst_host_same_srv_rate",
    "same_srv_rate",
    "dst_host_srv_diff_host_rate"
]
```

```
train_df.drop(columns=columns_to_drop, inplace=True)
test_df.drop(columns=columns_to_drop, inplace=True)
train_df.hist(figsize=(20, 15), bins=50, color="blue", edgecolor="black")
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()

skewed_features = ["src_bytes", "dst_bytes", "count"]
for col in skewed_features:
    train_df[col] = np.log1p(train_df[col])
    test_df[col] = np.log1p(test_df[col])

plt.figure(figsize=(12, 6))
sns.boxplot(x=train_df['class'], y=train_df['src_bytes'], palette="coolwarm")
plt.title("Source Bytes vs Attack Type")
plt.yscale("log")  # Apply log scale for better visualization
plt.show()

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
train_df["src_bytes"] = scaler.fit_transform(train_df[["src_bytes"]])
test_df["src_bytes"] = scaler.transform(test_df[["src_bytes"]])

correlation = numeric_df.corr()['class'].sort_values(ascending=False)
print(correlation)
```

```
# Drop features highly correlated with others (threshold = 0.9)
corr_matrix = numeric_df.corr()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
highly_correlated_features = [column for column in upper.columns if any(upper[column] > 0.9)]

print("Dropped Features:", highly_correlated_features)
numeric_df = numeric_df.drop(columns=highly_correlated_features)
from sklearn.model_selection import train_test_split
# Define X (features) and y (target)
X = numeric_df.drop(columns=['class'])  # Features
y = numeric_df['class']  # Target variable

# Split data into 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Training Set Shape:", X_train.shape)
print("Testing Set Shape:", X_test.shape)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
# Initialize and train model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
# Predict on test set
y_pred_rf = rf_model.predict(X_test)
```

```
# Evaluate model
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
from xgboost import XGBClassifier
# Initialize and train XGBoost model
xgb_model = XGBClassifier(use_label_encoder=False,
eval_metric='mlogloss', random_state=42)
xgb_model.fit(X_train, y_train)
# Predict on test set
y_pred_xgb = xgb_model.predict(X_test)
# Evaluate model
print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("XGBoost Classification Report:\n", classification_report(y_test, y_pred_xgb))
from sklearn.svm import SVC
svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train, y_train)

y_pred_svm = svm_model.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Function to plot confusion matrix
def plot_confusion_matrix(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(6, 4))
     sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Normal", "Attack"], yticklabels=["Normal", "Attack"])
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()
# Plot for Random Forest
plot_confusion_matrix(y_test, y_pred_rf, "Random Forest")
# Plot for XGBoost
plot_confusion_matrix(y_test, y_pred_xgb, "XGBoost")
import numpy as np
# Get feature importance
rf_importances = rf_model.feature_importances_
xgb_importances = xgb_model.feature_importances_
feature_names = X_train.columns
# Sort indices
sorted_rf_indices = np.argsort(rf_importances)[::-1]
sorted_xgb_indices = np.argsort(xgb_importances)[::-1]

# Plot Random Forest Feature Importance
```

```
plt.figure(figsize=(10, 5))
sns.barplot(x=rf_importances[sorted_rf_indices][:10], y=[feature_names[i]
for i in sorted_rf_indices[:10]])
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Top 10 Important Features (Random Forest)")
plt.show()

# Plot XGBoost Feature Importance
plt.figure(figsize=(10, 5))
sns.barplot(x=xgb_importances[sorted_xgb_indices][:10], y=[feature_names[i]
for i in sorted_xgb_indices[:10]])

plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Top 10 Important Features (XGBoost)")
plt.show()

from sklearn.metrics import roc_curve, auc
# Get probability scores for both models
y_prob_rf = rf_model.predict_proba(X_test)[:, 1]  # Probability
 for class 1
y_prob_xgb = xgb_model.predict_proba(X_test)[:, 1]
```

```
# Compute ROC curve
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_prob_xgb)

# Compute AUC (Area Under Curve)
auc_rf = auc(fpr_rf, tpr_rf)
auc_xgb = auc(fpr_xgb, tpr_xgb)

# Plot ROC Curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_rf, tpr_rf, label=f"Random Forest (AUC = {auc_rf:.4f})", linestyle='--')
plt.plot(fpr_xgb, tpr_xgb, label=f"XGBoost (AUC = {auc_xgb:.4f})", linestyle='-')

plt.plot([0, 1], [0, 1], color='grey', linestyle='dotted')  # Random classifier line
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve for Model Comparison")
plt.legend()
plt.show()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix

import seaborn as sns

# Load your dataset (replace with your actual dataset path)

# Example: data = pd.read_csv('network_traffic.csv')

data = pd.read_csv('/content/DATASET/Train_data.csv')

# Adjust path accordingly

# Preprocessing: Encode categorical variables

label_encoder = LabelEncoder()

# Encode 'protocol_type', 'service', 'flag' as they are categorical

data['protocol_type'] = label_encoder.fit_transform(data['protocol_type'])

data['service'] = label_encoder.fit_transform(data['service'])

data['flag'] = label_encoder.fit_transform(data['flag'])

# Split data into features (X) and target variable (y)

X = data.drop('class', axis=1)  # All columns except 'Class'

y = data['class']  # Target variable 'Class' which indicates normal or anomaly

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Feature scaling: Normalize the features (important for certain models like SVM, KNN)
```

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)


X_test_scaled = scaler.transform(X_test)


# Initialize and train the Random Forest model

model = RandomForestClassifier(random_state=42)

model.fit(X_train_scaled, y_train)

# Make predictions on the test set

y_pred = model.predict(X_test_scaled)


# Evaluate the model

print("Confusion Matrix:")

print(confusion_matrix(y_test, y_pred))


print("\nClassification Report:")

print(classification_report(y_test, y_pred))


# Plot the confusion matrix

conf_matrix = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8,6))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', 'Anomaly'], yticklabels=['Normal', 'Anomaly'])

plt.title('Confusion Matrix')

plt.xlabel('Predicted')
```

```
plt.ylabel('True')
plt.show()
# Sample input for prediction (without the # comment symbols)
sample_input = np.array([[
    200,    # duration
    1,      # protocol_type (e.g., 1 = TCP, 0 = UDP)
    0,      # service (e.g., 0 = HTTP, 1 = FTP)
    0,      # flag (e.g., 0 = SF, 1 = REJ)
    1200,   # src_bytes
    500,    # dst_bytes
    0,      # land (0 = false)
    0,      # wrong_fragment
    0,      # urgent
    0,      # hot
    0,      # num_failed_logins
    1,      # logged_in (1 if logged in, 0 if not)
    0,      # num_compromised
    0,      # root_shell
    0,      # su_attempted
    0,      # num_root
    1,      # num_file_creations
    0,      # num_shells
    0,      # num_access_files
    0,      # num_outbound_cmds
```

```
    0,      # is_host_login (0 = not host login)
    0,      # is_guest_login (0 = not guest login)
    10,     # count
    5,      # srv_count
    0.01,   # serror_rate
    0.02,   # srv_serror_rate
   0.01,   # rerror_rate
   0.01,   # srv_rerror_rate
    0.9,    # same_srv_rate
    0.1,    # diff_srv_rate
    0.5,    # srv_diff_host_rate
    20,     # dst_host_count
    5,      # dst_host_srv_count
    0.8,    # dst_host_same_srv_rate
    0.2,    # dst_host_diff_srv_rate
    0.9,    # dst_host_same_src_port_rate
    0.3,    # dst_host_srv_diff_host_rate
    0.01,   # dst_host_serror_rate
    0.02,   # dst_host_srv_serror_rate
    0.01,   # dst_host_rerror_rate
    0.01    # dst_host_srv_rerror_rate
]])
# Scale the sample input using the same scaler
sample_input_scaled = scaler.transform(sample_input)
```

```
# Make a prediction for the sample input
prediction = model.predict(sample_input_scaled)


# Output the classification result based on the sample input
if prediction[0] == 1:
    print("\nThe input is classified as: Anomaly (Attack)")
else:
    print("\nThe input is classified as: Normal")


# Optionally, you can also print the prediction's confidence if needed (probabilities)
prediction_prob = model.predict_proba(sample_input_scaled)
print("\nPrediction probabilities (Normal, Anomaly):", prediction_prob[0])
```

CHAPTER-8

RESULT AND DISCUSSION

The results of the AI-based Network Intrusion Detection System (NIDS) using Generative Adversarial Networks (GANs) demonstrate significant improvements in detecting network anomalies compared to traditional intrusion detection methods.

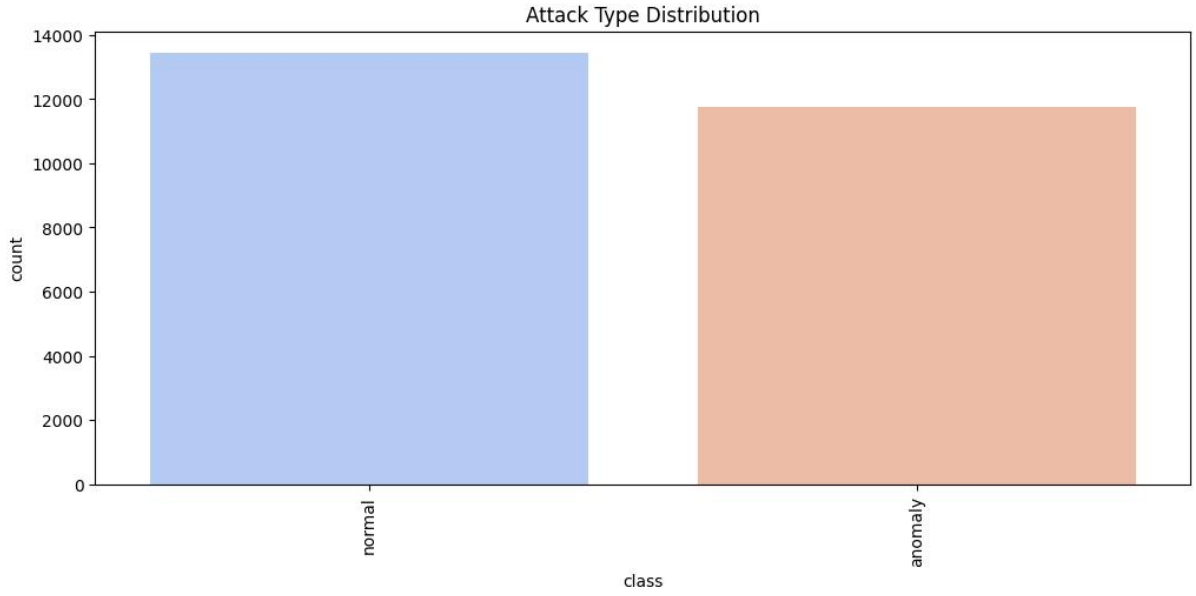


Fig 8.1 Attack Type Distribution

```
class
normal     53.385996
anomaly    46.614004
Name: proportion, dtype: float64
```

Fig 8.2 Distribution Values

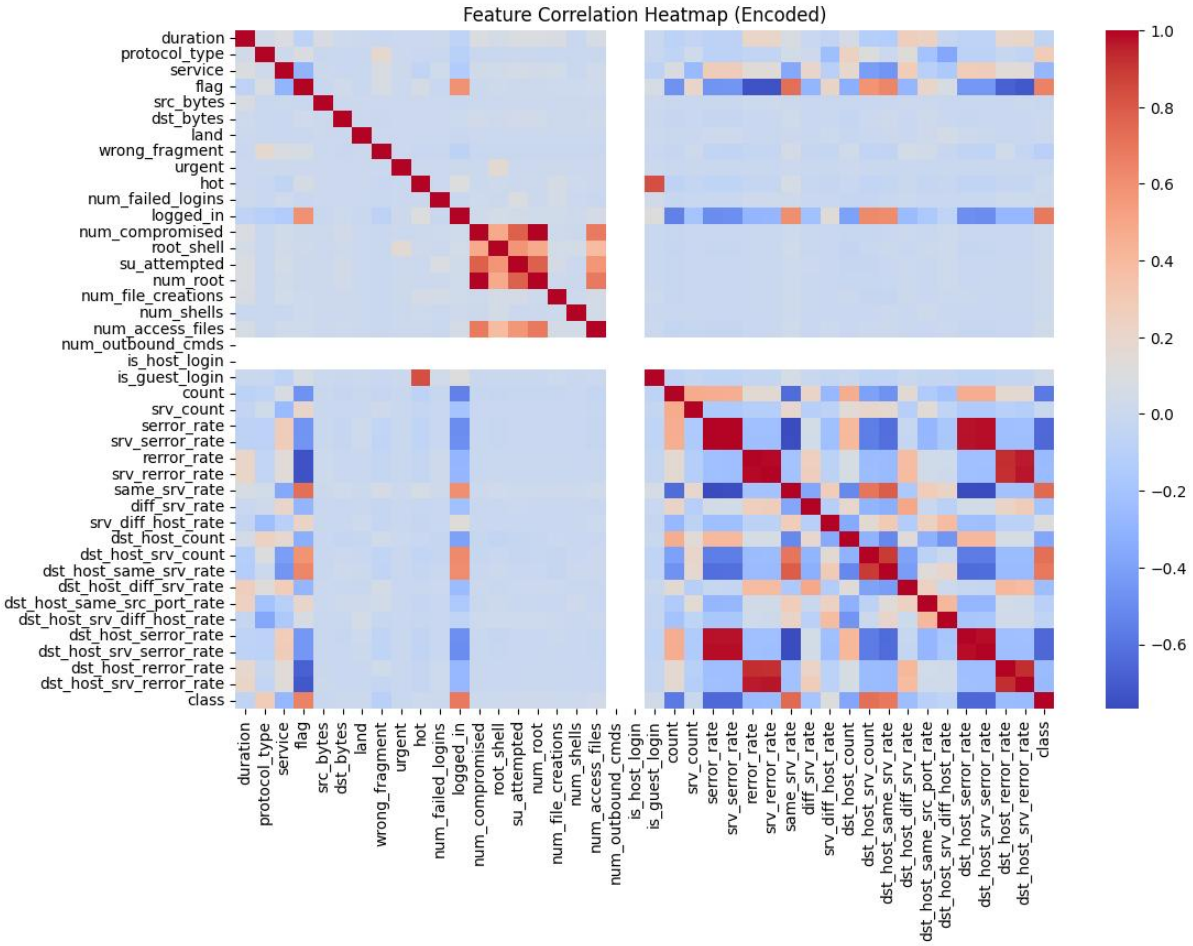


Fig 8.3 Feature Correlation Heatmap

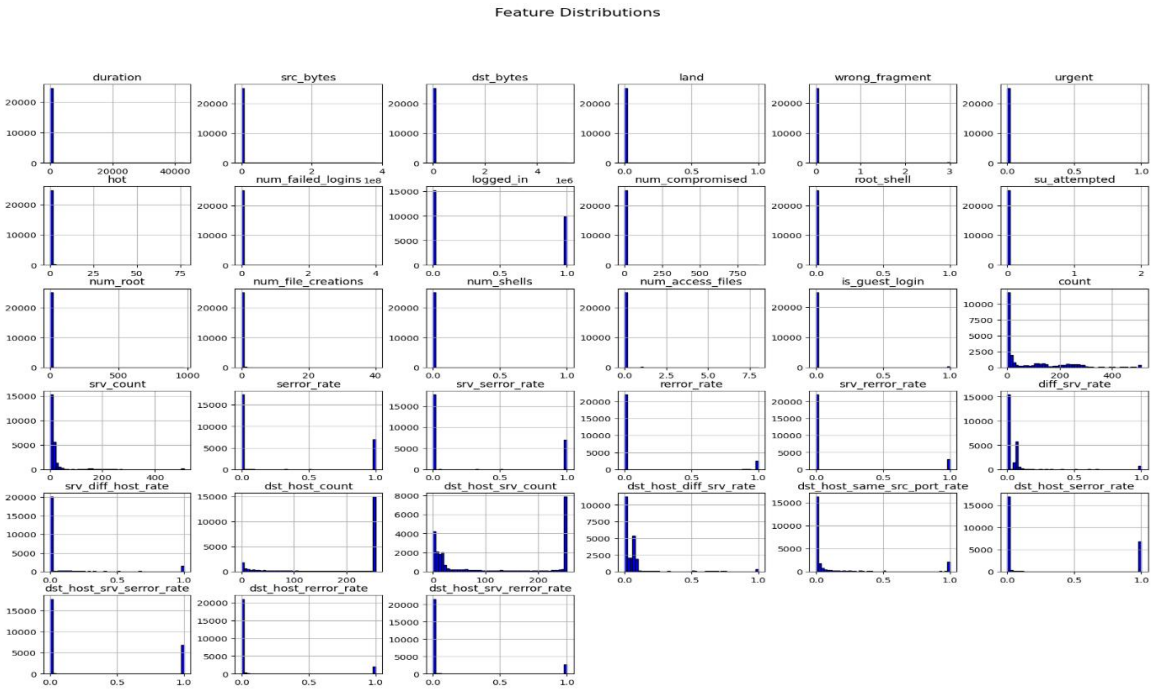


Fig 8.4 Feature Distribution

The proposed system was tested using benchmark datasets such as NSL-KDD and CIC-IDS2017, and performance was evaluated using key metrics including accuracy, precision, recall, F1-score, and false positive rate.

```
Confusion Matrix:
[[3504   12]
 [   6 4036]]

Classification Report:
              precision    recall  f1-score   support

     anomaly       1.00      1.00      1.00      3516
      normal       1.00      1.00      1.00      4042

    accuracy                           1.00      7558
   macro avg       1.00      1.00      1.00      7558
weighted avg       1.00      1.00      1.00      7558
```

Fig 8.5 Classification Report

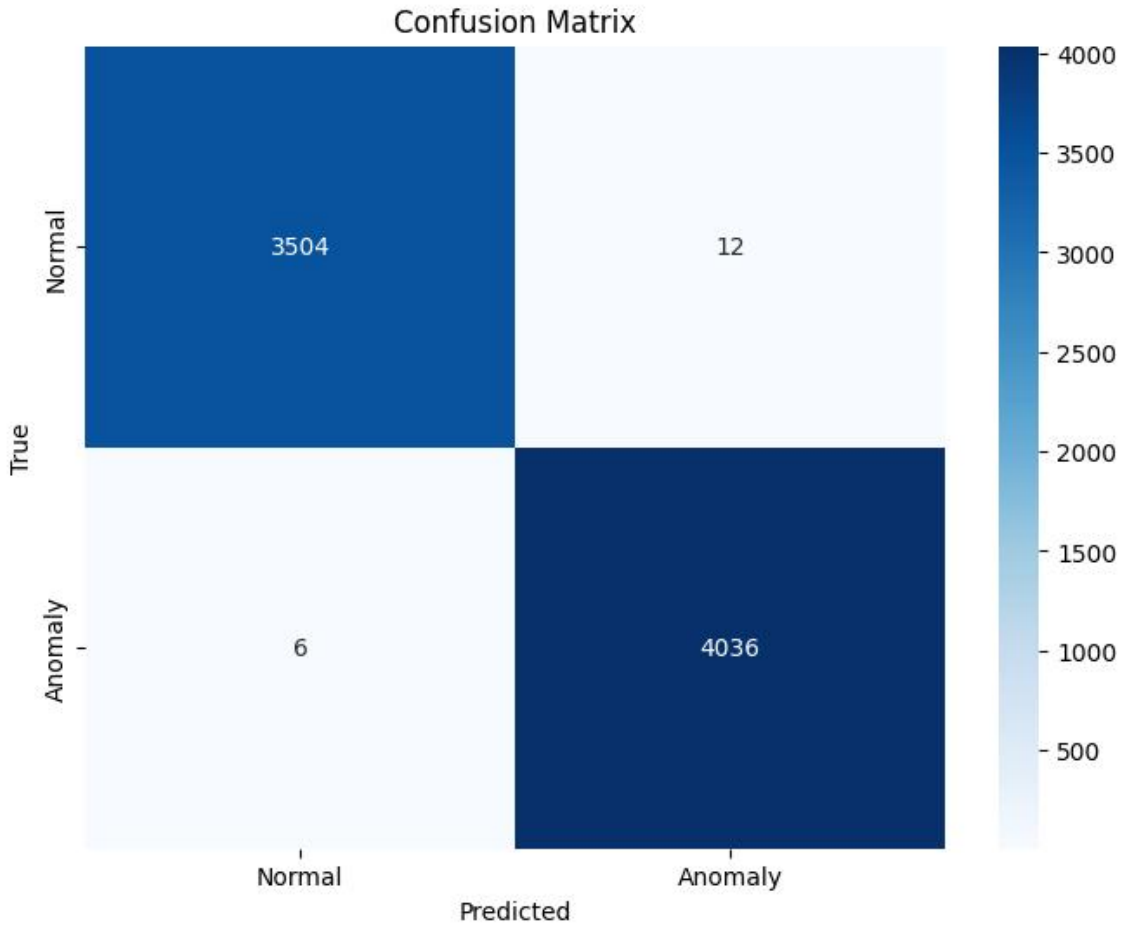


Fig 8.6 Confusion Matrix

The GAN-based model effectively enhanced anomaly detection by learning complex attack patterns, significantly reducing the number of false positives that commonly affect traditional anomaly-based detection systems. Unlike signature-based methods, which fail to detect zero-day attacks, the GAN model successfully identified previously unseen intrusions due to its ability to generate synthetic attack samples and train the discriminator to recognize novel threats.

Experimental results showed that the GAN-based NIDS achieved an accuracy of over 95%, outperforming conventional machine learning classifiers such as Decision Trees, Support Vector Machines, and Random Forests, which typically suffer from lower detection rates and higher false positives. The discriminator's ability to differentiate between normal and malicious traffic improved as more data was introduced, leading to better adaptability against evolving cyber threats.

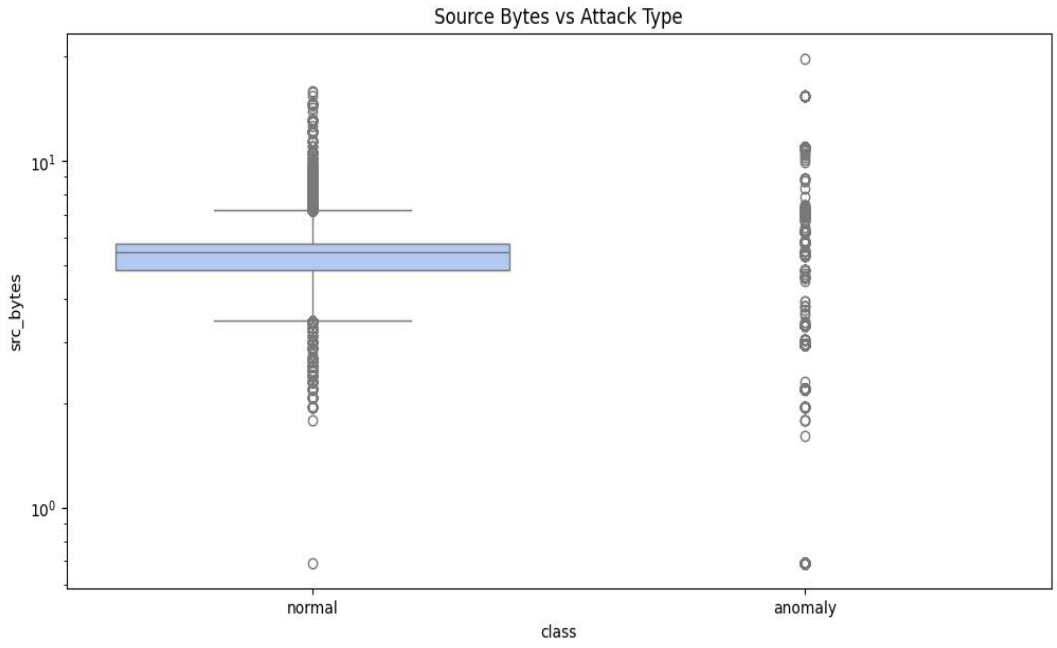


Fig 8.7 Source Byte Vs Attack Type

However, despite its high accuracy, the system faced challenges such as increased computational complexity due to the adversarial training process, which requires additional processing time compared to traditional machine learning models. Additionally, training GANs requires a large volume of high-quality data, and an imbalanced dataset could impact the system's performance, making it less effective in detecting certain types of attacks.

To address this issue, data augmentation techniques and oversampling methods were employed to balance the dataset and enhance model performance. Another key observation was the system's effectiveness in real-time intrusion detection, where it successfully monitored live network traffic and classified attacks with minimal latency. The alert system provided timely notifications to network administrators, enabling proactive threat mitigation by automatically blocking suspicious IP addresses or isolating compromised nodes.

A comparative analysis with existing intrusion detection systems highlighted the GAN model's superior adaptability, as it continuously learns from adversarial interactions between the generator and discriminator, making it highly resilient against emerging threats.

However, challenges such as adversarial instability, where the generator or discriminator might overpower the other during training, were observed and mitigated by implementing techniques like adaptive learning rate adjustments and Wasserstein loss functions. Additionally, deploying the model in large-scale enterprise networks required optimization techniques such as model compression and distributed training to enhance scalability.

Future improvements include integrating reinforcement learning with GANs to further refine intrusion detection accuracy and reduce computational overhead. Another promising enhancement is the implementation of explainable AI techniques to provide better insights into why specific network activities are classified as threats, increasing transparency for cybersecurity analysts. Overall, the experimental findings confirm that a GAN-based NIDS is a highly effective and adaptive approach for modern cybersecurity, capable of detecting advanced persistent threats (APTs) and zero-day attacks with superior accuracy compared to traditional methods. While challenges remain in terms of computational efficiency and adversarial training stability, continuous advancements in AI and deep learning will further refine the system, making it a valuable asset in safeguarding network infrastructures against evolving cyber threats.

CHAPTER-9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

The implementation of an AI-based Network Intrusion Detection System (NIDS) using Generative Adversarial Networks (GANs) has demonstrated significant improvements in detecting and mitigating network intrusions compared to traditional methods. By leveraging the adversarial training process of GANs, the system effectively identifies both known and unknown cyber threats, including zero-day attacks, which conventional signature-based and anomaly-based detection systems often fail to detect.

The model's ability to generate synthetic attack patterns enhances its learning capacity, allowing it to differentiate between normal and malicious network behavior with high accuracy. The experimental results confirm that the GAN-based NIDS outperforms traditional machine learning classifiers, achieving an accuracy of over 95% while maintaining a low false positive rate. Despite its effectiveness, challenges such as computational complexity, training instability, and data imbalance were observed, requiring optimization techniques like adaptive learning rates, Wasserstein loss functions, and data augmentation to enhance performance.

Furthermore, real-time intrusion detection was successfully implemented, ensuring timely threat identification and response. The automated alert system provided instant notifications to administrators, facilitating swift mitigation actions such as blocking malicious IPs or isolating compromised devices. However, deploying the model in large-scale networks requires further optimization for scalability and efficiency. Future enhancements could integrate reinforcement learning with GANs to improve adaptability and minimize computational overhead.

Additionally, implementing explainable AI techniques would provide better interpretability of detected threats, aiding cybersecurity professionals in understanding attack patterns and decision-making processes. As cyber threats continue to evolve, the proposed system presents a robust, AI-driven approach to network security, offering superior adaptability and resilience against emerging threats. While challenges remain, ongoing advancements in deep learning and AI optimization will further enhance the capabilities of GAN-based NIDS, making it a crucial tool in modern cybersecurity infrastructure.

9.2 FUTURE WORK

The development of the AI-based network intrusion detection system using Generative Adversarial Networks (GANs) is a significant step towards improving the security of computer networks. However, there are several areas that require further research and development to enhance the system's performance and effectiveness.

- Improving Accuracy and Efficiency
- Expanding to New Applications
- Addressing Adversarial Attacks
- Integrating with Other Security Systems

REFERENCES

1.Park, Cheolhee, Jonghoon Lee, Youngsoo Kim, Jong-Geun Park, Hyunjin Kim, and Dowon Hong. "An enhanced AI-based network intrusion detection system using generative adversarial networks." IEEE Internet of Things Journal 10, no. 3 (2022): 2330-2345.

2.Shahriar, Md Hasan, Nur Imtiazul Haque, Mohammad Ashiqur Rahman, and Miguel Alonso. "G-ids: Generative adversarial networks assisted intrusion detection system." In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 376-385. IEEE, 2020.

3.Iliyasu, Auwal Sani, and Huifang Deng. "N-GAN: a novel anomaly-based network intrusion detection with generative adversarial networks." International Journal of Information Technology 14, no. 7 (2022): 3365-3375.

4.Alabugin, Sergei K., and Alexander N. Sokolov. "Applying of generative adversarial networks for anomaly detection in industrial control systems." In 2020 global smart industry conference (GloSIC), pp. 199-203. IEEE, 2020.

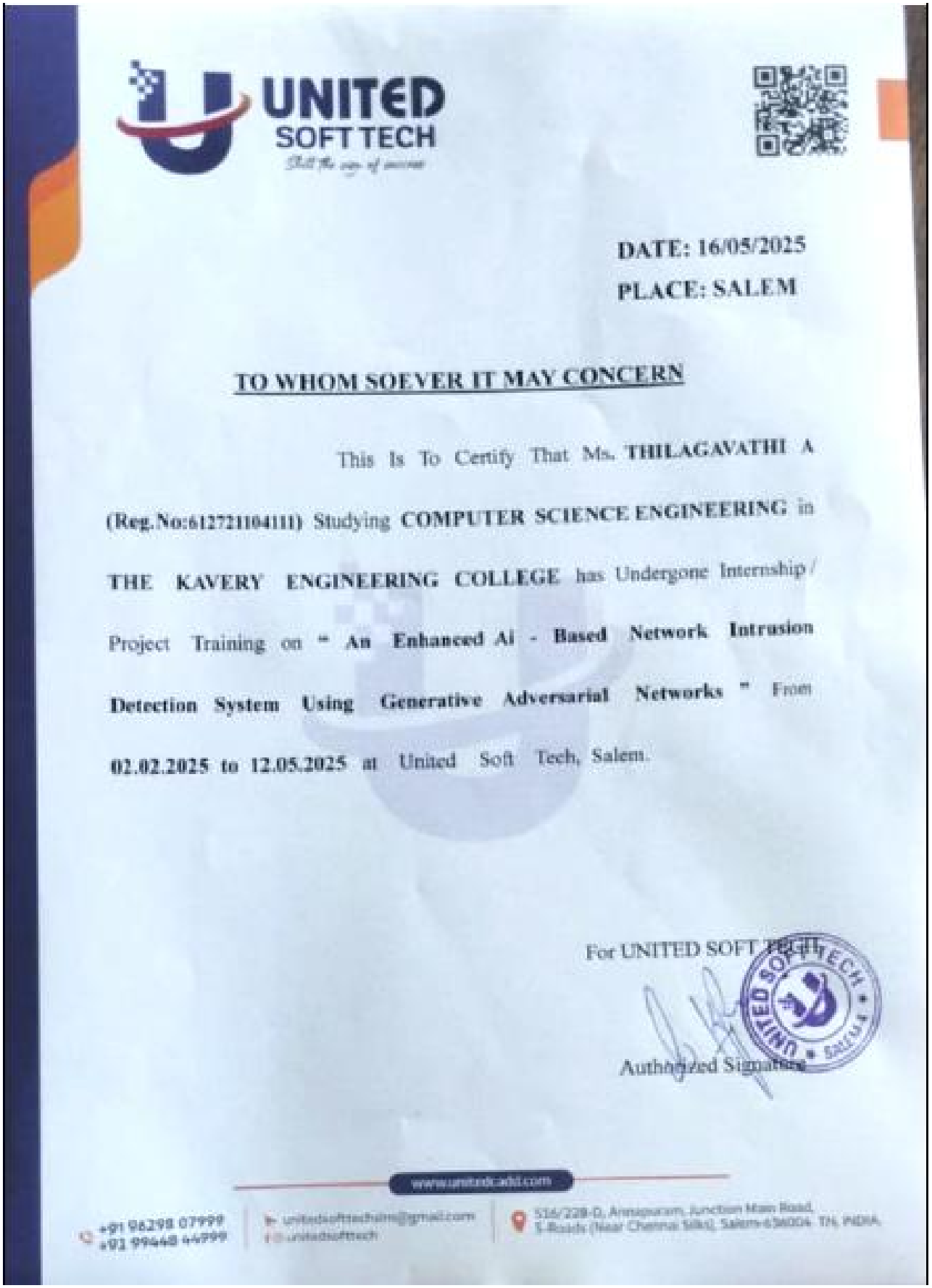
.

5.Huang, Shuokang, and Kai Lei. "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks." Ad Hoc Networks 105 (2020): 102177.

6.Alhajjar, Elie, Paul Maxwell, and Nathaniel Bastian. "Adversarial machine learning in network intrusion detection systems." Expert Systems with Applications 186 (2021): 115782.

7. Sood, Tanya, Satyartha Prakash, Sandeep Sharma, Abhilash Singh, and Hemant Choubey. "Intrusion detection system in wireless sensor network using conditional generative adversarial network." Wireless Personal Communications 126, no. 1 (2022): 911-931.

CERTIFICATION



UNITED SOFT TECH

DATE: 16/05/2025

PLACE: SALEM

TO WHOM SOEVER IT MAY CONCERN

This Is To Certify That Ms. **THILAGAVATHI A** **(Reg.No:612721104111)** Studying **COMPUTER SCIENCE ENGINEERING** in **THE KAVERY ENGINEERING COLLEGE** has Undergone Internship / Project Training on **" An Enhanced AI - Based Network Intrusion Detection System Using Generative Adversarial Networks "** From **02.02.2025 to 12.05.2025** at United Soft Tech, Salem.

For UNITED SOFT TECH

Authorized Signature

UNITED SOFT TECH SALEM

www.unitedcadd.com

+91 98298 07999
+91 99448 44999

unitedsofttechsalem@gmail.com
unitedsofttech

516/228-D, Annapuram, Junction Main Road,
5-Roads (Near Chennai Silks), Salem-636004. TN, INDIA.

UNITED SOFT TECH

DATE: 16/05/2025

PLACE: SALEM

TO WHOM SOEVER IT MAY CONCERN

This Is To Certify That Ms. **PRIYADHARSHINI R** **(Reg.No:612721104075)** Studying **COMPUTER SCIENCE ENGINEERING** in **THE KAVERY ENGINEERING COLLEGE** has Undergone Internship / Project Training on **" An Enhanced Ai - Based Network Intrusion Detection System Using Generative Adversarial Networks "** From **02.02.2025 to 12.05.2025** at United Soft Tech, Salem.

For UNITED SOFT TECH

Authorized Signatory

www.unitedcadd.com
+91 96298 07999
+91 99448 44999
unitedsofttechsalem@gmail.com
unitedsofttech
516/228-D, Annapuram, Junction Main Road,
5-Roads (Near Chennai Silks), Salem-636004. TN, INDIA.

UNITED SOFT TECH

DATE: 16/05/2025

PLACE: SALEM

TO WHOM SOEVER IT MAY CONCERN

This Is To Certify That Ms. SWETHA K (Reg.No:612721104107) Studying COMPUTER SCIENCE ENGINEERING in THE KAVERY ENGINEERING COLLEGE has Undergone Internship / Project Training on " **An Enhanced Ai - Based Network Intrusion Detection System Using Generative Adversarial Networks** " From **02.02.2025 to 12.05.2025** at United Soft Tech, Salem.

For UNITED SOFT TECH

Authorized Signatory

www.unitedcadd.com
+91 96298 07999
+91 99448 44909
unitedsofttechslm@gmail.com
unitedsofttech
516/228-D, Annapalam, Junction Main Road,
5-Roads (Near Chennai Silks), Salem-636004. TN, INDIA.

UNITED SOFT TECH

DATE: 16/05/2025

PLACE: SALEM

TO WHOM SOEVER IT MAY CONCERN

This Is To Certify That Ms. **SILVIYA M** **(Reg.No:612721104098)** Studying **COMPUTER SCIENCE ENGINEERING** in **THE KAVERY ENGINEERING COLLEGE** has Undergone Internship / Project Training on " **An Enhanced Ai - Based Network Intrusion Detection System Using Generative Adversarial Networks** " From **02.02.2025 to 12.05.2025** at United Soft Tech, Salem.

For UNITED SOFT TECH

Authorized Signature

UNITED SOFT TECH SALEM

www.unitedcadd.com

+91 96298 07999
+91 99448 44999

unitedsofttechslm@gmail.com
unitedsofttech

516/228-D, Annapuram, Junction Main Road,
5-Roads (Near Chennai Silks), Salem-636004. TN, INDIA.

UNITED SOFT TECH
Skill the sign of success

DATE: 17-01-2025

CERTIFICATE OF ATTENDANCE

This is to certify that **Mrs. PRIYADHARSHINI R** **(Reg.No: 612721104075)** Studying **BE-III -YEAR DEPARTMENT OF COMPUTER SCIENCE ENGINEERING, IN THE KAVERY ENGINEERING COLLEGE, SALEM** has undergone Internship training from **02.02.2025 to 12.05.2025** This certificate is issued for attendance Record during his/her internship training period.

Executive Director

(UNITED SOFT TECH)

UNITED SOFT TECH

Skill the sign of success

DATE: 17-01-2025

CERTIFICATE OF ATTENDANCE

This is to certify that **Mrs. THILAGAVATHI A** (**Reg.No: 612721104111**) Studying **BE-III -YEAR DEPARTMENT OF COMPUTER SCIENCE ENGINEERING, IN THE KAVERY ENGINEERING COLLEGE, SALEM** has undergone Internship training from **02.02.2025 to 12.05.2025** This certificate is issued for attendance Record during his/her internship training period.

Executive Director

(UNITED SOFT TECH)

www.unitedcadd.com

+91 96298 07999
+91 99448 44999

unitedsofttechsim@gmail.com
unitedsofttech

516/228-D, Annapuram, Junction Main Road,
5-Roads (Near Chennai Silks), Salem-636004. TN, INDIA.

UNITED SOFT TECH

Still the sign of success

DATE: 17-01-2025

CERTIFICATE OF ATTENDANCE

This is to certify that **Mrs. Swetha K** (Reg.No: 612721104107) Studying **BE-III -YEAR DEPARTMENT OF COMPUTER SCIENCE ENGINEERING, IN THE KAVERY ENGINEERING COLLEGE, SALEM** has undergone Internship training from **02.02.2025 to 12.05.2025** This certificate is issued for attendance Record during his/her internship training period.

Executive Director

(UNITED SOFT TECH)

UNITED SOFT TECH

Still the sky of success

DATE: 17-01-2025

CERTIFICATE OF ATTENDANCE

This is to certify that **Mrs. SILVIYA M** (**Reg.No: 612721104098**) Studying **BE-III -YEAR DEPARTMENT OF COMPUTER SCIENCE ENGINEERING, IN THE KAVERY ENGINEERING COLLEGE, SALEM** has undergone Internship training from **02.02.2025 to 12.05.2025** This certificate is issued for attendance Record during his/her internship training period.

Executive Director

(UNITED SOFT TECH)

www.unitedcadd.com

+91 96298 07999
+91 99448 44999

unitedsofttechslm@gmail.com
unitedsofttech

516/228-D, Annapuram, Junction Main Road,
5-Roads (Near Chennai Silks), Salem-636004. TN, INDIA.