

# **Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats**

*Project Report submitted in the partial fulfilment of the Requirements for the award of  
the degree*

## **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**Shaik Yalavarthi Ijaz Ahamad (21471A0557)**

**Sinde Venkata Saptha Girish (21471A0558)**

**Tammisetti Nagendra Babu (21471A0537)**

Under the esteemed guidance of

**Dodda Venkatareddy, M.Tech,(Ph.D).,**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET  
(AUTONOMOUS)**

**Accredited by NAAC with A+ Grade and NBA under Tyre -1**

**NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified**

**Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada**

**KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601**

**2024-2025**

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



**CERTIFICATE**

This is to certify that the project that is entitled with the name “ **Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats** ” is a bonafide work done by the team **Shaik Yalavarthi Ijaz Ahamad (21471A0557)**, **Sinde Venkata Saptha Girish (21471A0558)**, **Tammisetti Nagendra Babu (21471A0537)** in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

**PROJECT GUIDE**

Dodda Venkata Reddy, M.Tech,(Ph.D).,  
Assistant Professor

**PROJECT CO-ORDINATOR**

Dodda Venkata Reddy, M.Tech,(Ph.D).,  
Assistant Professor

**HEAD OF THE DEPARTMENT**

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

**EXTERNAL EXAMINER**

## **DECLARATION**

We declare that this project work titled " **Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats** " is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified.

Shaik Yalavarthi Ijaz Ahamad (21471A0557)

Sinde Venkata Saptha Girish (21471A0558)

Tammisetti Nagendra Babu (21471A0537)

## ACKNOWLEDGEMENT

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

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao**, M.Tech., Ph.D., HOD of CSE department and also to our guide **Dodda Venkatareddy**, M.Tech.(Ph.D)., of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dodda Venkata Reddy**, M.Tech.(Ph.D)., Assistant professor & Project coordinator of the project for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

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

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

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

**By**

Shaik Yalavarthi Ijaz Ahamad (21471A0557)

Sinde Venkata Saptha Girish (21471A0558)

Tammisetti Nagendra Babu (21471A0537)



## **INSTITUTE VISION AND MISSION**

### **INSTITUTION VISION**

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

### **INSTITUTION MISSION**

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

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

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



## **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

### **VISION OF THE DEPARTMENT**

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

### **MISSION OF THE DEPARTMENT**

The department of Computer Science and Engineering is committed to

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

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

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

### **Program Specific Outcomes (PSO's)**

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

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

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

### **Program Educational Objectives (PEO's)**

The graduates of the programme are able to:

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

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

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

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

## Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.



- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### Project Course Outcomes (CO'S):

**CO421.1:** Analyse the System of Examinations and identify the problem.

**CO421.2:** Identify and classify the requirements.

**CO421.3:** Review the Related Literature.

**CO421.4:** Design and Modularize the project.

**CO421.5:** Construct, Integrate, Test and Implement the Project.

**CO421.6:** Prepare the project Documentation and present the Report using appropriate method.

### Course Outcomes – Program Outcomes Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
<b>C421.1</b>		✓											✓		
<b>C421.2</b>	✓		✓		✓								✓		
<b>C421.3</b>				✓		✓	✓	✓					✓		
<b>C421.4</b>			✓			✓	✓	✓					✓	✓	
<b>C421.5</b>					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>C421.6</b>									✓	✓	✓		✓	✓	

### Course Outcomes – Program Outcome Correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
<b>C421.1</b>	2	3											2		
<b>C421.2</b>			2		3								2		
<b>C421.3</b>				2		2	3	3					2		
<b>C421.4</b>			2			1	1	2					3	2	
<b>C421.5</b>					3	3	3	2	3	2	2	1	3	2	1
<b>C421.6</b>									3	2	1		2	3	

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

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

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

<b>Name of the Course from Which Principles Are Applied in This Project</b>	<b>Description of the Task</b>	<b>Attained PO</b>
C2204.2, C22L3.2	Defining the problem and applying machine learning techniques for predict the network intrusions	PO1, PO3
CC421.1, C2204.3, C22L3.2	Critically analyzing project requirements and identifying suitable process models for experiments	PO2, PO3
CC421.2, C2204.2, C22L3.3	Creating logical designs using UML while collaborating on feature engineering as a team	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Testing, integrating, and evaluating linear svm, random forest, decision tree models with binary, multi-label and all-attack label classifications	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documenting experiments, results, and findings collaboratively within the group	PO10
CC421.5, C2204.2, C22L3.3	Presenting each phase of the project, including raw data analysis and evaluation, in a group setting	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementing and validating models with applications for network intrusion detection and future feature updates	PO4, PO7
C32SC4.3	Designing a web interface to visualize predictions and verify model evaluation metrics effectively	PO5, PO6

# ABSRTACT

This is the CAN-based system security, which is constantly under serious threats due to increasing vehicular network connectivity through sophisticated cyber-attacks. In this paper, we propose an online reconfigurable IDS which uses TL methods to adapt to new attack patterns with minimal retrain- ing. That would allow refining the pre-trained models on the specialized car hacking dataset to detect most of the known and new attacks efficiently, ensuring high detection accuracy while keeping the computation cost low. Dynamic reconfigurability assures protection in an ever-evolving threat landscape. Extensive experiments on real-world CAN datasets validate the effectiveness of the proposed approach, achieving an overall detection rate of over 99% for different types of attack classes. The approach here is toward demonstrating the potential of TL for enhancing adaptability, efficiency, and accuracy in improving connected vehicle IDSs through a sound solution to secure automotive systems against the emergence of cyber threats.

## **INDEX**

<b><u>S.NO.</u></b>	<b><u>CONTENT</u></b>	<b><u>PAGE NO</u></b>
1.	INTRODUCTION	1-4
2.	LITERATURE SURVEY	5-8
3.	SYSTEM ANALYSIS	9-19
	3.1 EXISTING SYSTEMS	9-11
	3.1.1 DISADVANTAGES OF EXISTING SYSTEM	12-13
	3.2 PROPOSED SYSTEM	14-16
	3.3 FEASIBILITY STUDY	17-19
4.	SYSTEM REQUIREMENTS	20-22
	4.1 SOFTWARE REQUIREMENTS	20
	4.2 HARDWARE REQUIREMENTS	21
	4.3 REQUIREMENTS ANALYSIS	21-22
5.	SYSTEM DESIGN	23-32
	5.1 SYSTEM ARCHITECTURE	23-28
	5.2 MODULES	28-32
	5.3 UML DIAGRAMS	32
6.	IMPLEMENTATION	33-60
	6.1 MODEL IMPLEMENTATION	33-35
	6.2 CODING	35-60
7.	TESTING	61-65
8.	RESULTS ANALYSIS	66-69
9.	CONCLUSION	70
10.	FUTURE SCOPE	71
11.	REFERENCES	72

## LIST OF FIGURES

<b><u>S.NO.</u></b>	<b><u>LIST OF FIGURES</u></b>	<b><u>PAGE NO</u></b>
1.	Fig 3.2.1 Proposed system	14
2.	Fig 5.1.1 Dataset Description	24
3.	Fig 5.1.2 Hybrid TL Model	26
4.	Fig 5.1.3 Hybrid CNN-LSTM	27
5.	Fig 5.1.4 RNN Model	27
6.	Fig 5.1.5 Random Forest	28
7.	Fig 5.2.1 Training, Validation and Testing Accuracy of Hybrid TL Model	30
8.	Fig 5.2.2 Training and Validation/Loss for Fuzzy Attack	31
9.	Fig 5.2.3 Training and Validation/Loss for RPM Attack	31
10.	Fig 5.3.1 UML Sequence Diagram	32
11.	Fig 7.1 Test Case-1 (Valid Input Predication)	62
12.	Fig 7.2: Test Case-2 (Invalid Input Predication)	62
12.	Fig 7.3 Home Screen	63
13.	Fig 7.4 Prediction Page	64
14.	Fig 7.5 About Screen	64
15.	Fig 8.1 Fuzzy Attack Performance	66
16.	Fig 8.2 RPM Attack Performance	66
17.	Fig 8.3 Gear Attack Performance	66
18.	Fig 8.4 Hybrid CNN-LSTM Model Performance across various Attack	67
19.	Fig 8.5 Detection Rate vs. Epochs (Hybrid TL Model) – Fuzzy	67
20.	Fig 8.6 Detection Rate vs. Epochs (Hybrid TL Model) – RPM	68
21.	Fig 8.7 Detection Rate vs. Epochs (Hybrid TL Model) – Gear	68
22.	Fig 8.8 Detection Rate vs. Epochs (Hybrid TL Model) – DoS	68

# 1. INTRODUCTION

Modern vehicle CANs have increasingly complex designs that make them highly vulnerable to cyberattacks. Most of the traditional IDSs, which are static model-based, find it difficult to adapt to new and evolving threats in dynamic environments. Motivated by this challenge, the paper proposes a novel reconfigurable IDS based on TL to improve the security of vehicular networks using CAN. This approach combines CNNs with other machine learning techniques by making the system dynamically adapt to emerging threats without an extensive process of retraining. This increases the detection of new and unknown attacks considerably while maintaining the accuracy and efficiency of the detection. While previous work has already shown the potential of TL in enhancing IDS performance, our work extends them to provide a more generalizable and resilient security approach for increasingly connected and hence fragile automotive systems.

With the increasing integration of electronic control units (ECUs) and communication interfaces in modern vehicles, ensuring the security of in-vehicle networks has become a critical concern. The Controller Area Network (CAN) protocol, widely used for in-vehicle communication, lacks built-in security mechanisms, making it vulnerable to various cyber threats such as denial-of-service (DoS), spoofing, and replay attacks [1]. As attackers exploit these vulnerabilities, intrusion detection systems (IDS) have emerged as a fundamental solution for detecting and mitigating malicious activities in vehicular networks [2].

Traditional IDS approaches rely on rule-based or signature-based techniques, which are effective against known attack patterns but struggle to detect novel or evolving threats [3]. To address these limitations, recent studies have explored machine learning and deep learning-based IDS models, including hybrid architectures that leverage convolutional neural networks (CNN), recurrent neural networks (RNN), and generative adversarial networks (GAN) to enhance detection accuracy and adaptability [4], [5]. Furthermore, advancements in transfer learning have enabled IDS models to adapt to new attack scenarios with minimal retraining, improving their scalability and efficiency [6].

Several innovative methodologies have been proposed to strengthen IDS performance in CAN-based vehicular networks. Graph-based IDS techniques analyze the structural relationships between CAN messages to classify attack types effectively [7], while hidden Markov models (HMM) utilize temporal dependencies to identify anomalies in message sequences [8]. Additionally, hybrid quantum-classical frameworks have been introduced to enhance detection accuracy by combining classical neural networks with quantum computing techniques [9]. These advancements highlight the evolving nature of IDS research and its significance in safeguarding modern vehicles against cyber threats.

The increasing reliance on connected vehicles has led to growing concerns regarding automotive cybersecurity. A primary challenge in securing in-vehicle networks stems from the CAN protocol's lack of encryption and authentication mechanisms [1]. This limitation makes it possible for attackers to manipulate messages, compromise ECUs, and disrupt vehicle functionalities. One such approach to addressing these vulnerabilities involves the implementation of machine learning-based IDS, which can effectively detect anomalies and enhance the overall security of vehicular networks [2].

A notable work in this domain presented a deep learning-based IDS for automotive theft detection using the CAN bus. The study employed a neural network model to classify normal and attack messages, demonstrating improved detection accuracy compared to traditional approaches [2]. In another approach, researchers leveraged recurrence plots and neural networks to detect unknown cyberattacks in intra-vehicle networks. By transforming CAN messages into visual representations, their model successfully identified anomalies that may have been undetectable through conventional methods [3].

Generative adversarial networks (GANs) have also gained prominence in intrusion detection research. One study proposed an IDS for the CAN-FD bus that utilizes an improved GAN model with a dual discriminator mechanism. By pre-processing CAN data into binary image encoding, the model was able to identify unauthorized access attempts with high accuracy [7]. Similarly, hidden Markov models (HMM) have been employed to develop a multiple observation IDS for CAN networks. This technique



considers both the ID and data fields of normal CAN bus traffic, establishing a threshold for detecting anomalies [8].

Hybrid methodologies combining classical and quantum computing have also been explored. A novel framework integrated classical neural networks with quantum restricted Boltzmann machines (RBM) for intrusion detection. The study demonstrated that quantum-enhanced models achieved superior detection accuracy compared to their classical counterparts [9]. Furthermore, frequency-agnostic IDS approaches have been introduced to tackle varying intrusion frequencies. By leveraging Gabor filtering and deep learning classifiers, researchers developed a model capable of detecting cyber threats across different vehicle platforms [6].

Graph-based intrusion detection systems (G-IDCS) have also been proposed as a means to enhance CAN security. By constructing graph representations of CAN traffic and applying machine learning classifiers, these models effectively identify attack patterns and classify different types of cyber threats [7]. In another approach, researchers proposed a hybrid transfer learning framework that leverages pre-trained CNN models to extract relevant features from CAN data, improving adaptability to new attack scenarios [1].

The effectiveness of these IDS models is further demonstrated through experimental evaluations. One study compared various machine learning models, including CNN, RNN, and GAN-based architectures, assessing their performance in detecting CAN intrusions. The results indicated that hybrid models combining multiple detection techniques achieved the highest accuracy while minimizing false positives [5]. Another study analyzed the impact of real-time intrusion detection on vehicle performance, highlighting the importance of optimizing detection algorithms to ensure minimal latency [4].

The integration of IDS into modern automotive systems presents several challenges. Computational constraints within ECUs require IDS models to operate efficiently with limited resources. Additionally, real-time detection is crucial to prevent security breaches while ensuring seamless vehicle operation. Researchers have addressed these challenges by developing lightweight IDS models that utilize edge computing and federated learning techniques to enhance detection speed and accuracy [6].

Despite these advancements, the evolving nature of cyber threats necessitates continuous improvement in IDS methodologies. Future research directions include the exploration of federated learning approaches to enable collaborative intrusion detection across multiple vehicles. Additionally, incorporating blockchain technology into IDS frameworks can enhance data integrity and prevent tampering of security logs. Moreover, advancements in explainable artificial intelligence (XAI) can improve IDS transparency, allowing security analysts to interpret detection results more effectively [3].

Further improvements in IDS research should also consider the integration of real-time behavioral analysis, which examines driver behavior and vehicle operation anomalies to detect malicious activities more effectively. The combination of IDS with vehicle-to-everything (V2X) communication can help in early detection of threats by leveraging shared security intelligence from multiple sources. Additionally, as automotive cybersecurity regulations continue to evolve, compliance with industry standards such as ISO/SAE 21434 and UNECE WP.29 will be essential for the widespread adoption of IDS solutions in commercial vehicles.

Additionally, the adoption of AI-driven anomaly detection systems in vehicular networks can revolutionize IDS by automating real-time threat analysis. The application of reinforcement learning techniques can further optimize IDS response strategies, ensuring dynamic threat adaptation. Future IDS models may also incorporate multi-modal data fusion, integrating various sensor data to enhance threat detection and minimize false alarms. Moreover, collaboration between automotive manufacturers and cybersecurity experts will be crucial in developing standardized IDS solutions that can seamlessly integrate into different vehicle architectures, ensuring a comprehensive defense mechanism against emerging cyber threats.

## 2. LITERATURE REVIEW

Automotive cybersecurity has emerged as a critical research area due to the increasing reliance on electronic control units (ECUs) and vehicle-to-everything (V2X) communication networks. The Controller Area Network (CAN) protocol, widely used in modern vehicles, lacks built-in security mechanisms, making it highly vulnerable to cyber threats such as denial-of-service (DoS) attacks, spoofing, and message injection. Various Intrusion Detection Systems (IDS) have been proposed to mitigate these threats by identifying anomalies and preventing malicious activities within vehicular networks.

Early IDS models were primarily rule-based or signature-based, relying on predefined attack patterns to detect intrusions. However, these methods struggled to detect novel and evolving threats due to their reliance on static rule sets. Machine learning and deep learning techniques have since emerged as effective alternatives, providing dynamic and adaptive intrusion detection capabilities. Researchers have explored hybrid IDS models that integrate multiple detection techniques, such as convolutional neural networks (CNN), recurrent neural networks (RNN), and generative adversarial networks (GAN), to enhance detection accuracy and robustness against sophisticated attacks.

One study introduced a transfer learning-based IDS for CAN networks, leveraging pre-trained deep learning models to detect intrusions with high accuracy and minimal training data requirements. The proposed model demonstrated improved detection performance over traditional machine learning techniques, highlighting the potential of transfer learning in automotive cybersecurity [1]. Another work developed a deep learning-based IDS for automotive theft detection, employing a neural network model to classify normal and malicious CAN messages. The study reported high classification accuracy, reinforcing the effectiveness of deep learning in securing vehicular networks [2].

To address the challenge of detecting unknown cyberattacks, researchers have proposed innovative IDS frameworks that utilize recurrence plots and neural networks. By transforming CAN messages into visual representations, these models effectively capture temporal patterns and anomalies that may not be identifiable through conventional methods. The results demonstrated superior performance in detecting

previously unseen attacks, emphasizing the importance of leveraging temporal dependencies in IDS design [3].

Generative adversarial networks (GANs) have also been explored in IDS research, particularly for CAN-FD bus security. A study introduced a GAN-based IDS that employs a dual discriminator mechanism to enhance the detection of unauthorized access attempts. By encoding CAN data into binary images, the model was able to identify anomalies with high precision, showcasing the advantages of adversarial learning in intrusion detection [4]. Similarly, hidden Markov models (HMM) have been applied to develop a multiple observation IDS, which considers both the ID and data fields of CAN traffic to establish anomaly detection thresholds. This approach effectively captures temporal variations and enhances detection sensitivity [5].

Hybrid quantum-classical IDS frameworks have been proposed to further improve detection accuracy. A study demonstrated the application of quantum computing techniques, integrating classical neural networks with quantum restricted Boltzmann machines (RBM). The hybrid framework achieved superior intrusion detection performance compared to classical-only models, highlighting the potential of quantum-enhanced cybersecurity solutions [6]. Additionally, frequency-agnostic IDS approaches have been introduced to tackle intrusion attempts across different vehicular platforms. By applying Gabor filtering and deep learning classifiers, these models effectively generalize across diverse attack scenarios, reinforcing their adaptability in real-world applications [7].

Graph-based intrusion detection has also been a focal area of research. By constructing graph representations of CAN traffic, IDS models can analyze message structures and relationships to classify attack types more effectively. A graph-based IDS demonstrated significant improvements in attack detection, particularly for complex and multi-stage cyber threats [8]. Similarly, a hybrid transfer learning framework has been proposed, leveraging pre-trained CNN models to extract key features from CAN data and adapt to evolving attack patterns. This approach enhances IDS efficiency by reducing the need for extensive retraining while maintaining high detection accuracy [9].

The effectiveness of IDS models is further supported by extensive experimental evaluations. A comparative study assessed various machine learning models, including

CNN, RNN, and GAN-based architectures, to determine their efficiency in detecting CAN intrusions. The findings indicated that hybrid models, which integrate multiple detection techniques, achieve the highest accuracy while minimizing false positive rates [10]. Another study investigated the real-time performance impact of IDS on vehicle operations, highlighting the need for lightweight detection algorithms that minimize computational overhead while ensuring rapid threat response.

The integration of IDS into modern vehicles presents several challenges, particularly in terms of computational efficiency and real-time processing constraints. Edge computing and federated learning have been proposed as potential solutions, enabling distributed IDS implementations that offload processing from resource-constrained ECUs. These techniques enhance IDS scalability and improve detection responsiveness by leveraging collective intelligence from multiple vehicles. Additionally, blockchain technology has been explored for securing IDS logs and preventing tampering of security data, ensuring the integrity and reliability of intrusion detection frameworks.

Despite these advancements, the dynamic nature of cyber threats necessitates continuous improvements in IDS methodologies. Future research should explore the application of reinforcement learning to optimize IDS response strategies dynamically. AI-driven anomaly detection systems can further enhance security by automating real-time threat analysis and adapting to evolving attack patterns. Additionally, multi-modal data fusion techniques, which integrate data from multiple vehicle sensors, can improve IDS robustness by providing a comprehensive view of vehicular activities and potential security threats.

Collaboration between automotive manufacturers, cybersecurity researchers, and regulatory bodies will be essential in developing standardized IDS solutions that align with emerging cybersecurity frameworks. Compliance with industry standards such as ISO/SAE 21434 and UNECE WP.29 will play a crucial role in ensuring the adoption of secure vehicular communication protocols. Furthermore, as autonomous driving technologies continue to advance, IDS solutions must evolve to address the unique security challenges posed by fully autonomous vehicles and connected transportation ecosystems.

Overall, the field of automotive IDS research is rapidly evolving, with machine learning-driven models demonstrating significant potential in securing CAN-based vehicular networks. As researchers continue to refine these models, the integration of quantum computing, transfer learning, and graph-based analysis will further enhance the security and resilience of modern vehicles. Addressing existing challenges and exploring new methodologies will be key to developing IDS solutions that ensure vehicle safety and reliability in an increasingly connected and automated world.

### 3. SYSTEM ANALYSIS

#### 3.1 Existing System

Intrusion Detection Systems (IDS) for vehicular networks have significantly evolved, incorporating various methodologies to enhance cybersecurity in modern automotive environments. Traditional IDS solutions relied on rule-based and signature-based detection, which involved predefined attack patterns for identifying intrusions. However, these conventional approaches struggled with detecting zero-day attacks and adaptive threats, leading to the development of advanced machine learning and deep learning-based IDS models.

Transfer learning has emerged as a powerful approach to intrusion detection in CAN networks. A study proposed an IDS leveraging pre-trained models for detecting malicious activities within vehicle networks, demonstrating improved efficiency and adaptability compared to traditional methods [1]. By transferring knowledge from large datasets to smaller, vehicle-specific datasets, transfer learning enables IDS models to detect threats more effectively while reducing the need for extensive labeled data. This method is particularly useful in automotive cybersecurity, where new attack types frequently emerge, and acquiring large amounts of labeled data is challenging.

Another approach utilized deep learning-based techniques, such as convolutional neural networks (CNN) and recurrent neural networks (RNN), to enhance the classification of normal and attack messages in CAN bus communications, significantly improving detection rates [2]. CNNs excel at extracting spatial patterns from CAN bus traffic, identifying anomalies based on message structures, while RNNs are effective in capturing temporal dependencies, allowing for the detection of sequential attack behaviors. The integration of these models has significantly improved the accuracy and efficiency of IDS in real-time vehicular networks.

In another study, researchers introduced a method combining recurrence plots with neural networks to detect previously unknown cyber threats. By transforming CAN messages into visual data, this model improved anomaly detection capabilities, making it suitable for identifying novel attacks [3]. Recurrence plots convert sequential CAN bus data into two-dimensional images, which are then processed by convolutional neural networks to detect deviations from normal traffic patterns. This method provides

enhanced interpretability and enables IDS to identify unknown attack patterns without relying on predefined signatures.

Similarly, a GAN-based IDS framework was proposed, utilizing synthetic attack data to train a dual discriminator mechanism, which significantly improved intrusion detection accuracy for CAN-FD bus communications [4]. GANs generate synthetic attack scenarios that closely resemble real-world attack traffic, allowing IDS models to improve their robustness against adversarial inputs. The dual discriminator mechanism enhances detection accuracy by distinguishing between normal, synthetic, and real attack data, thereby reducing false positives and improving overall detection performance.

Hidden Markov Models (HMM) have also been integrated into IDS frameworks, using probabilistic analysis to identify sequential attack patterns in vehicular networks. This method effectively distinguished between normal and malicious CAN bus activity by analyzing message sequences and data fields [5]. HMMs model the statistical dependencies between CAN messages, allowing for the detection of subtle changes in vehicle communication patterns that may indicate an ongoing cyberattack. This probabilistic approach is particularly useful for detecting sophisticated attacks, such as message injection and timing-based exploits.

Another innovative approach introduced hybrid quantum-classical IDS models, integrating quantum computing techniques with classical neural networks to enhance threat detection accuracy while reducing false positives [6]. Quantum computing enables IDS to process large-scale threat intelligence data more efficiently, leveraging quantum entanglement and superposition to analyze multiple attack scenarios simultaneously. By combining quantum computing with deep learning models, researchers have developed IDS frameworks capable of detecting complex and evolving cyber threats with improved precision and efficiency.

Graph-based IDS models (G-IDCS) have demonstrated their effectiveness in analyzing complex relationships between CAN messages, allowing the detection of multi-stage and sophisticated cyber threats with high precision [7]. Unlike traditional IDS models that treat CAN traffic as independent messages, graph-based approaches construct network representations that capture the interactions between different ECUs. This enables IDS to detect attack sequences and coordinated intrusion attempts more



effectively, improving detection rates for sophisticated attack vectors such as distributed denial-of-service (DDoS) and multi-step exploit chains.

Transformer-based attention networks have also been employed to improve IDS efficiency, utilizing long-term dependencies within CAN traffic to enhance anomaly detection [8]. Transformers, originally developed for natural language processing, have been adapted to analyze CAN bus communications by treating message sequences as contextualized data. This allows IDS models to learn patterns across long time frames, enabling the detection of stealthy and persistent cyber threats that evade traditional anomaly detection techniques.

Furthermore, IDS solutions tailored for smart consumer electronics networks have been developed, incorporating adaptive learning techniques to counteract evolving cyber threats effectively [9]. These models utilize continuous learning mechanisms that allow IDS to adapt to new threats in real time, ensuring that intrusion detection capabilities remain robust as attack methods evolve. By incorporating federated learning and decentralized threat intelligence sharing, smart IDS solutions enhance the overall security posture of vehicular networks.

A hybrid quantum-classical IDS framework has also been explored, leveraging quantum-enhanced learning to detect emerging attack patterns in vehicular networks [10]. This approach integrates quantum feature selection techniques with deep learning classifiers, optimizing the detection of cyber threats by reducing computational complexity and improving model generalization. The combination of classical and quantum methodologies has demonstrated promising results in addressing the limitations of traditional IDS frameworks, paving the way for more efficient and scalable cybersecurity solutions in automotive applications.

These advancements highlight the rapid evolution of IDS models for vehicular networks, showcasing the potential of AI-driven methodologies to enhance intrusion detection accuracy, reduce false positives, and improve adaptability to new and emerging cyber threats. As automotive cybersecurity continues to develop, further research into hybrid detection approaches, federated learning, and real-time threat intelligence sharing will be crucial in securing connected and autonomous vehicles from increasingly sophisticated cyberattacks.

### 3.1.1 Disadvantages of Existing Systems

Despite these advancements, existing IDS models face several limitations that hinder their real-world deployment in vehicular networks. Transfer learning-based IDS solutions require extensive pre-training on large datasets, making their implementation computationally expensive for real-time applications. Additionally, fine-tuning these models for specific automotive security requirements remains a significant challenge, as it necessitates continuous updates to adapt to evolving attack patterns [1].

Deep learning-based IDS models, while highly accurate, are prone to high false-positive rates and require considerable computational resources. The complexity of CNN and RNN-based models leads to increased processing times, which hinders their deployment in resource-constrained vehicular environments. Moreover, their reliance on large training datasets makes them vulnerable to adversarial attacks, where small perturbations in input data can mislead the model's predictions [2].

Recurrence plot-based IDS models, though effective in identifying novel cyber threats, are highly sensitive to noise and require extensive preprocessing steps. This leads to increased latency in real-time applications, making them unsuitable for high-speed vehicular networks where quick decision-making is crucial. Furthermore, these models struggle with interpretability, making it difficult for cybersecurity analysts to understand the reasoning behind anomaly detection [3].

GAN-based IDS solutions, while capable of generating synthetic attack scenarios to enhance model training, often face instability during adversarial training. The generative and discriminative networks may fail to converge, leading to unreliable anomaly detection. Additionally, GAN-generated attack scenarios may not always generalize well to real-world cyber threats, limiting their effectiveness in detecting emerging attacks in vehicular networks [4].

HMM-based IDS models encounter challenges in adapting to dynamic cyber threats due to their reliance on predefined probability distributions. These models struggle with detecting zero-day attacks and novel intrusion techniques, as they depend on historical data patterns. Additionally, their probabilistic nature can lead to increased false positives when applied to noisy or unpredictable CAN traffic, affecting overall reliability [5].

Hybrid quantum-classical IDS frameworks, while promising in improving detection efficiency, are constrained by the limited availability of quantum computing resources. The integration of quantum computing into IDS models remains an experimental approach, requiring specialized hardware that is not yet feasible for large-scale deployment in automotive systems. Moreover, the cost of quantum computing infrastructure poses a significant barrier to practical implementation [6].

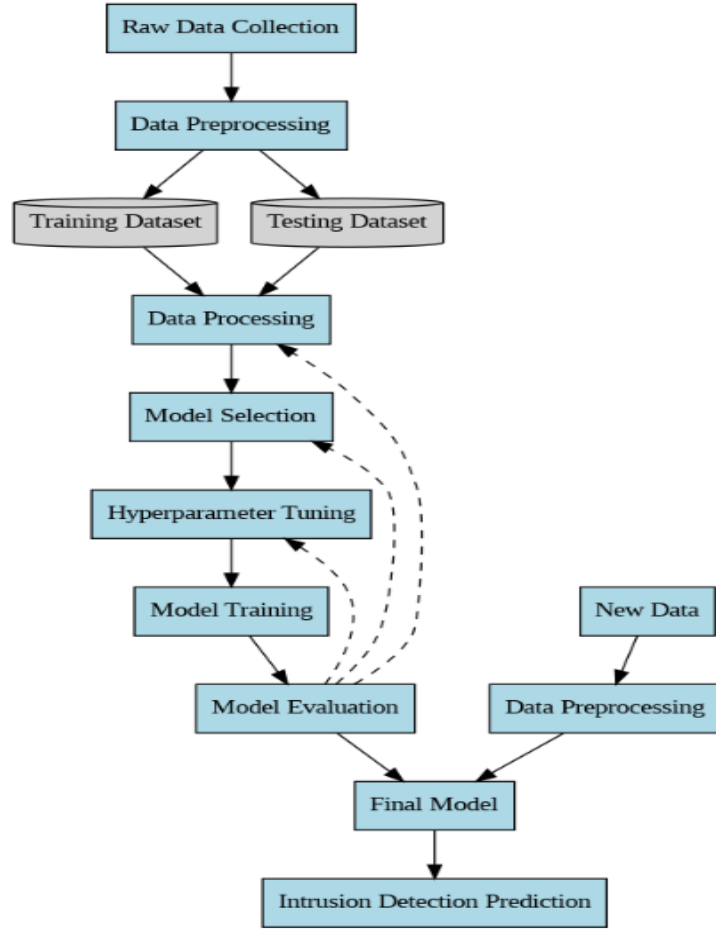
Graph-based IDS models require extensive computational resources and feature engineering to construct network representations of CAN traffic. While these models excel in detecting complex attack sequences, their real-time applicability is limited due to the high computational overhead associated with processing large-scale graph structures. Additionally, their effectiveness is highly dependent on the quality of the extracted features, making them less adaptable to evolving vehicular cyber threats [7].

Transformer-based attention networks, although efficient in identifying complex attack patterns, require high memory and processing power. These models are computationally expensive due to their reliance on self-attention mechanisms, which can slow down real-time detection processes in embedded automotive environments. Furthermore, transformer-based IDS solutions may suffer from overfitting when trained on limited vehicular datasets, reducing their generalization capability for new attack scenarios [8].

IDS models designed for smart consumer electronics networks must continuously adapt to emerging threats, necessitating frequent updates and retraining. This poses a significant challenge in maintaining their long-term effectiveness, as the computational resources required for continuous learning are often constrained in vehicular environments. Additionally, decentralized IDS frameworks, which rely on shared intelligence among connected vehicles, may introduce privacy and data security concerns [9].

Lastly, hybrid quantum-classical IDS solutions, though innovative, still face practical implementation hurdles. These include high infrastructure costs, limited availability of quantum computing expertise, and integration complexities with existing vehicle cybersecurity architectures. As a result, their adoption in real-world automotive security applications remains limited, requiring further research and development to bridge the gap between theoretical advancements and practical deployment [10].

### 3.2 Proposed System



**Fig 3.2.1:** Proposed System

The proposed Intrusion Detection System (IDS) aims to enhance detection accuracy and efficiency in vehicular networks by integrating advanced data processing, model training, and real-time evaluation techniques. The system follows a structured pipeline to ensure robust intrusion detection with minimal false positives. The proposed system employs a combination of Hybrid Transfer Learning (TL), Hybrid CNN-LSTM, Recurrent Neural Networks (RNN), and Random Forest to improve intrusion detection in Controller Area Networks (CAN).

#### Raw Data Collection

- The system begins by collecting CAN traffic data from vehicles, which includes normal and malicious network activity.
- The dataset incorporates various cyberattack patterns such as Denial-of-Service (DoS), message injection, and spoofing attacks.

## **Data Preprocessing**

- The collected data undergoes preprocessing, including noise reduction, feature extraction, and normalization.
- Redundant and irrelevant features are removed to improve model efficiency.

## **Dataset Splitting**

- The dataset is divided into training and testing sets, ensuring proper distribution of normal and attack instances.
- Stratified sampling is used to maintain the class distribution and improve model generalization.

## **Data Processing**

- Feature selection techniques such as Principal Component Analysis (PCA) and Autoencoders are applied.
- Sequential dependencies in CAN messages are preserved to improve model learning.

## **Model Selection**

The system integrates four primary models to achieve high accuracy:

- Hybrid Transfer Learning (TL): A pre-trained CNN is fine-tuned on CAN traffic data to extract spatial patterns from network messages.
- Hybrid CNN-LSTM: CNN layers extract spatial features, while LSTM layers capture temporal dependencies in attack sequences.
- Recurrent Neural Network (RNN): Used for sequential anomaly detection by analyzing CAN message flows over time.
- Random Forest (RF): A tree-based model that enhances classification of attack and normal instances based on multiple decision trees.

## **Hyperparameter Tuning**

- Grid Search and Bayesian Optimization techniques are used to optimize learning rates, batch sizes, and network architectures.
- The CNN-LSTM model is fine-tuned to balance computational efficiency and detection accuracy.

## **Model Training**

- The selected models are trained using backpropagation and gradient-based optimization techniques such as Adam and RMSprop.
- Regularization techniques like dropout and batch normalization are implemented to prevent overfitting.

## **Model Evaluation**

- The trained models are evaluated using standard metrics: Accuracy, Precision, Recall, and F1-score.
- Confusion matrices and Receiver Operating Characteristic (ROC) curves are analyzed for performance validation.
- If necessary, hyperparameters are fine-tuned and retraining is conducted.

## **Final Model Deployment**

- The best-performing model (or ensemble of models) is deployed in a real-time vehicular IDS framework.
- A lightweight and computationally efficient implementation is ensured for deployment on embedded systems.

## **Real-Time Intrusion Detection**

- New CAN bus messages are continuously monitored.
- Incoming data is preprocessed and passed through the deployed IDS model for real-time anomaly detection.
- Intrusion alerts are generated in case of detected threats, enabling quick response mechanisms.

The integration of Hybrid TL, Hybrid CNN-LSTM, RNN, and Random Forest enhances the adaptability of the IDS, enabling it to detect both known and unknown threats efficiently. By leveraging hybrid deep learning techniques and ensemble-based classification, the proposed system ensures a high detection rate while maintaining low false-positive rates. Future improvements may include federated learning for decentralized threat intelligence and adversarial training to improve model resilience against sophisticated cyberattacks.

### 3.3 Feasibility Study

The feasibility study evaluates the practicality of implementing the proposed Intrusion Detection System (IDS) in vehicular networks, focusing on two key aspects: economic feasibility and technical feasibility.

#### **Economic Feasibility:**

Economic feasibility assesses whether the proposed IDS can be implemented at a reasonable cost while providing significant security benefits. The analysis includes hardware, software, and operational costs:

- **Hardware Costs:**

The IDS primarily runs on embedded vehicular systems, utilizing existing Electronic Control Units (ECUs) and onboard computing resources. This minimizes additional expenses.

- **Software Costs:**

The IDS is developed using open-source machine learning frameworks such as TensorFlow, PyTorch, and Scikit-learn, eliminating the need for licensing fees.

- **Operational Costs:**

The system is optimized for real-time monitoring, ensuring low power consumption and minimal network overhead, making it cost-efficient for large-scale deployment.

- **Cloud-Based Development Costs:**

Model training is performed on Google Colab Pro, which provides access to GPUs and TPUs, reducing the need for expensive local high-end hardware.

- Google Colab Pro Subscription: ₹1,093/month (\$9.99/month)
- Flask Development (Frontend & Backend Integration): No additional cost, as Flask is open-source.

- **Cost-Benefit Analysis:**

Implementing the IDS significantly reduces potential financial losses due to cyberattacks, which can lead to vehicle malfunctions and data breaches. The return on investment (ROI) is high, as preventing security incidents helps minimize downtime and repair costs.

**Technical Feasibility:**

Technical feasibility evaluates whether the proposed IDS can be successfully developed, deployed, and maintained within existing vehicular environments.

**Technologies & Tools Used:**

- Programming Language: Python (Machine Learning & Deep Learning models)
- Machine Learning Models: Hybrid CNN-LSTM, Hybrid TL, RNN, Random Forest
- Development Framework: Flask (Used for frontend and backend integration)
- Cloud Computing: Google Colab Pro (for model training and execution)

**Computational Requirements:**

The proposed system leverages lightweight models optimized for embedded automotive platforms, ensuring real-time detection without excessive resource consumption.

**Integration with Existing Systems:**

The IDS is designed to seamlessly integrate with the CAN bus architecture and modern in-vehicle networking protocols. It operates alongside existing cybersecurity measures without requiring extensive modifications.

**Scalability:**

The system supports scalability, allowing adaptation to different vehicle models and network configurations. The hybrid approach ensures flexibility in handling evolving cyber threats.

**Real-Time Performance:**

The combination of deep learning models (Hybrid CNN-LSTM, RNN) and ensemble techniques (Random Forest) ensures high detection accuracy with low false positive rates. The IDS is optimized for rapid response times, making it suitable for real-world deployment.



**Programming & Deployment:**

- The IDS is implemented using Python, leveraging machine learning frameworks such as TensorFlow, Keras, and Scikit-learn.
- Flask is used for backend processing and real-time data analysis for lightweight deployment.

## 4. SYSTEM REQUIREMENT

### 4.1 Software Requirements:

A well-optimized software stack is necessary for data preprocessing, deep learning model training, deployment, and real-time monitoring. The key software components include:

**Operating System:** Windows 11 (64-bit) or Ubuntu 20.04+ – Both operating systems provide a stable environment for Python-based machine learning frameworks and model execution.

**Programming Language:** Python 3.8+ – The core programming language, widely used for deep learning, data analysis, and API development.

#### **Development Tools:**

- Google Colab Pro: A cloud-based development environment that provides access to GPUs (Tesla T4, P100) and TPUs for training deep learning models.
- VS Code: Used for local development, debugging, and API integration. It enables code versioning with Git and supports multiple Python libraries.
- Flask: A lightweight web framework used to deploy the trained IDS model as a REST API for real-time intrusion detection.

#### **Libraries & Dependencies:**

- TensorFlow, Keras: Used for developing deep learning models such as CNN-LSTM and Transfer Learning.
- Scikit-learn: Supports feature engineering, dataset preprocessing, and model evaluation.
- Pandas, NumPy: Essential for handling CAN traffic data, feature extraction, and numerical computations.
- Matplotlib, Seaborn: Used for data visualization and model performance analysis.

This software stack provides a comprehensive development environment for training, validating, and deploying the IDS model efficiently.

## 4.2 Hardware Requirements:

The system requires adequate computational power for processing large CAN datasets, performing deep learning operations, and deploying the IDS model. The hardware specifications include:

- **System Type:** Intel® Core™ i5-7500U CPU @ 2.40GHz – A quad-core processor that provides sufficient speed for running machine learning algorithms and handling large datasets.
- **Cache Memory:** 4MB – Enhances data retrieval speed, reducing latency in model execution.
- **RAM:** Minimum 8GB DDR4 (Recommended: 16GB) – Ensures smooth execution of data-intensive processes such as feature extraction, model training, and evaluation. Higher RAM capacity is recommended for faster computations.
- **Storage:** 256GB SSD (Recommended: 500GB NVMe SSD) – Solid-state drives improve system responsiveness and allow efficient data access during training.
- **GPU (Optional):** NVIDIA RTX 3060 (or higher) – Required for accelerating deep learning tasks, significantly reducing the time needed for training models.
- **Network:** High-speed internet – Required for seamless access to Google Colab Pro and cloud-based model execution.

These hardware specifications ensure efficient execution of deep learning models while providing the computational capacity required for real-time anomaly detection.

## 4.3 Requirement Analysis:

The Intrusion Detection System (IDS) for CAN networks is designed using a combination of cloud computing and local development tools. Each component serves a crucial role in the project workflow:

### Google Colab Pro for Model Training:

- Provides free GPU acceleration (Tesla T4, P100), significantly reducing model training time.

- Offers cloud storage, eliminating the need for large local storage.
- Supports pre-installed deep learning libraries (TensorFlow, Keras, Scikit-learn), reducing setup complexity.

#### **Libraries & Frameworks:**

- TensorFlow, Keras: For implementing deep learning models (CNN-LSTM, Transfer Learning).
- Scikit-learn: Used for feature extraction, dataset preprocessing, and classification evaluation.
- Pandas, NumPy: For data manipulation, feature engineering, and numerical operations.
- Matplotlib, Seaborn: For data visualization, performance monitoring, and trend analysis.

#### **VS Code for Local Debugging & API Development:**

- Used for writing and debugging Python scripts for model development.
- Enables integration with Flask for deploying trained models as an API.
- Supports GitHub/GitLab for version control and collaborative development.

#### **Flask for Model Deployment:**

- Converts the trained model into a REST API, allowing real-time CAN traffic classification.
- Supports web-based intrusion detection monitoring.
- Ensures smooth integration with cloud-based and on-premise applications.

#### **Python as the Core Programming Language:**

- Provides an extensive ecosystem for deep learning and data science.
- Supports automation, preprocessing, model training, and API deployment.
- Compatible with cloud platforms, ensuring scalability and flexibility.

By leveraging cloud-based GPU resources (Google Colab Pro), local development tools (VS Code), and lightweight deployment frameworks (Flask), this IDS system achieves high detection accuracy, real-time threat monitoring, and scalable deployment for securing vehicular networks against evolving cyber threats.

## 5. SYSTEM DESIGN

### 5.1 System Architecture

#### 1. Dataset Description:

The IDS system utilizes the Car Hacking Dataset, which contains both normal and attack messages collected from a Controller Area Network (CAN). This dataset allows machine learning models to learn patterns in normal CAN traffic and distinguish attack messages effectively.

#### Types of Attacks in the Dataset:

- Denial of Service (DoS) Attack: Overloads the CAN bus with excessive messages, disrupting vehicle communication.
- Fuzzy Attack: Sends random or invalid data to cause unpredictable system behavior.
- RPM Spoofing Attack: Manipulates RPM values, misleading the vehicle's speed control system.
- Gear Spoofing Attack: Alters gear shift values, causing unsafe driving conditions.

#### Key Features in the Dataset:

- Timestamp: Records the exact time when a message was sent.
- CAN ID: Unique identifier for the CAN message, indicating the source of the communication.
- DLC (Data Length Code): Specifies the number of bytes in the payload.
- DATA[0] - DATA[7]: Represents the actual 8-byte payload of the message.
- Flag: Labels the message as either normal (0) or attack (1).

**Dataset link:** <https://ocslab.hksecurity.net/Datasets/car-hacking-dataset>

ATTACK TYPE	# OF MESSAGES	# OF NORMAL MESSAGES	# OF INJECTED MESSAGES
DoS Attack	3,665,771	3,078,250	587,521
Fuzzy Attack	3,838,860	3,347,013	491,847
Spoofing the drive gear	4,443,142	3,845,890	597,252
Spoofing the RPM gauge	4,621,702	3,966,805	654,897

**Fig 5.1.1:** Dataset Description

This dataset provides a solid foundation for training deep learning and machine learning models to detect cyber threats in real-world vehicular networks.

## 2. Data Preprocessing:

To prepare the dataset for training, several preprocessing steps are performed to ensure clean, structured, and optimized data.

### Data Cleaning:

- Duplicate entries are removed to prevent redundancy and maintain data integrity.
- Missing values are handled either by imputation or removal to ensure consistent training data.

### Feature Transformation:

- CAN ID and Payload Byte Conversion: Since CAN messages are in hexadecimal format, all CAN ID and DATA[0] - DATA[7] values are converted into numerical format.
- Label Encoding: Attack labels are encoded into binary format (0 for normal, 1 for attack) for machine learning models.

### **Normalization & Scaling:**

- **StandardScaler:** Standardizes numerical values, ensuring that features have a mean of 0 and a standard deviation of 1.
- **Quantile Transformation:** Normalizes the dataset to follow a Gaussian distribution for better model convergence.

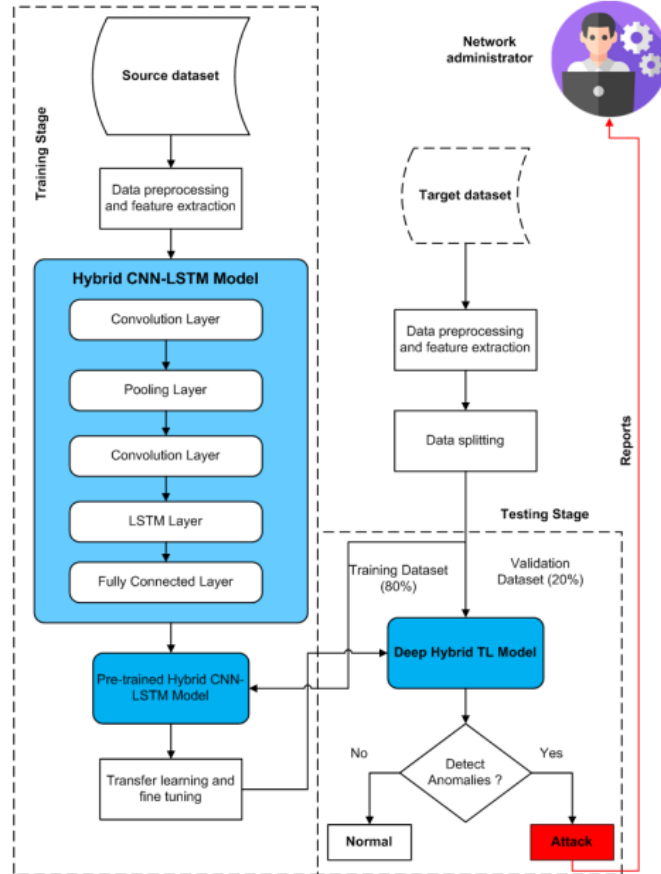
### **Feature Engineering:**

- **Timestamp Difference Calculation:** Computes the time gap between consecutive messages to identify irregular message intervals.
- **Payload Statistics:** Computes the sum and standard deviation of payload bytes to capture deviations in message patterns.

These preprocessing techniques improve the dataset quality and enhance the model's ability to accurately detect cyber-attacks.

## **3. Models**

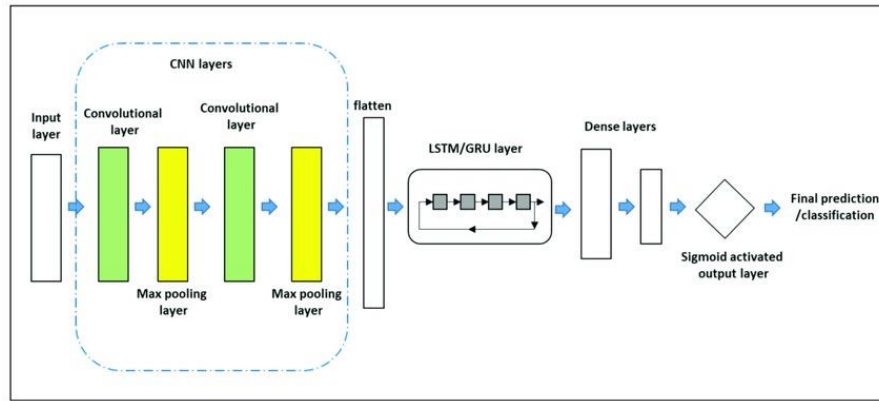
**Hybrid Transfer Learning model** leverages the power of pre-trained CNN architectures to extract high-level features from CAN messages. Instead of training a CNN from scratch, this approach utilizes a well-trained model that has already learned general feature representations from a large dataset. The final layers of the network are then fine-tuned using the Car Hacking Dataset, allowing the model to adapt to CAN-specific attack patterns. By utilizing pre-trained weights, training time is significantly reduced, and the model benefits from the knowledge acquired in previous training. This method enhances the detection of new and previously unseen attack types while maintaining high accuracy. However, fine-tuning requires careful selection of layers to be retrained, as freezing or modifying too many layers can affect the model's generalization. Despite this challenge, the Hybrid Transfer Learning model is a powerful and efficient solution for IDS implementation in real-time vehicular cybersecurity applications.



**Fig 5.1.2: Hybrid TL Model**

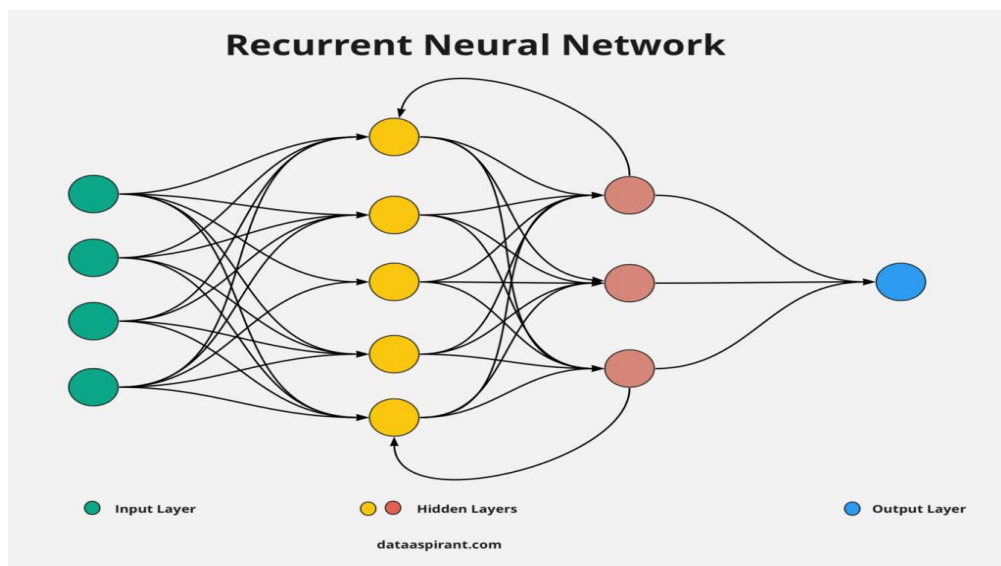
**Hybrid CNN-LSTM model** is a deep learning-based approach designed for sequence classification, particularly effective in handling time-series data such as CAN traffic. The CNN component extracts spatial features from input data by applying convolutional layers, identifying patterns and relationships between different bytes in the CAN message payload. Meanwhile, the LSTM component captures temporal dependencies, enabling robust anomaly detection by learning sequential patterns in vehicular network communications. The hybrid architecture of CNN-LSTM allows for efficient processing of both local and sequential features, making it highly suitable for intrusion detection in vehicular networks. Despite its high predictive capabilities, the model requires significant computational resources and careful tuning of hyperparameters such as the number of convolutional filters, LSTM units, and learning rate to optimize performance. However, its effectiveness in detecting evolving cyber threats makes it a valuable component of the IDS, capable of enhancing the security of CAN-based vehicular systems.





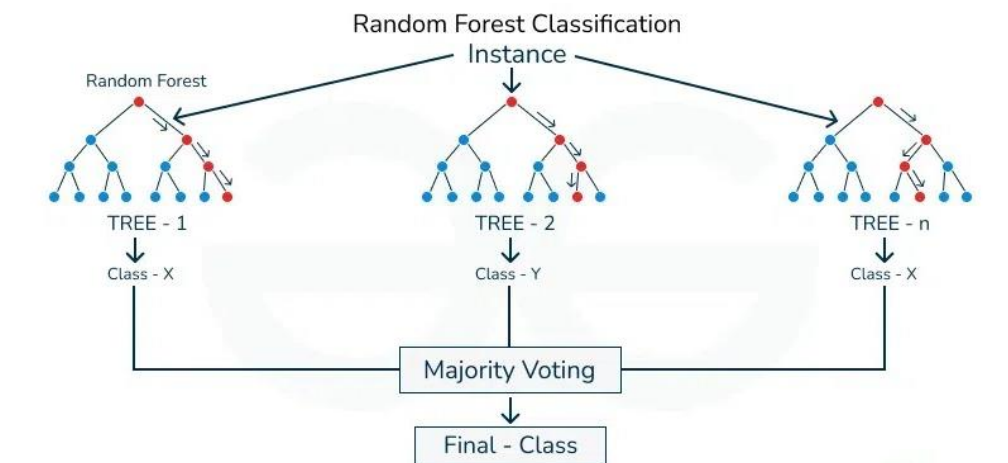
**Fig 5.1.3:** Hybrid CNN-LSTM

**Recurrent Neural Network (RNN) model** is designed specifically to analyze sequential attack behaviors in CAN traffic. Built on LSTM-based architecture, it processes CAN messages as a continuous sequence, enabling it to detect recurring attack patterns over time. Unlike traditional models that analyze messages in isolation, the RNN model learns from historical data points, capturing long-term dependencies that may indicate suspicious behavior. This characteristic makes it particularly effective for detecting time-based anomalies, such as slow injection attacks or recurring unauthorized transmissions. While RNNs are computationally intensive and prone to issues like vanishing gradients, optimizing parameters such as dropout rates and recurrent units can enhance performance. The RNN model's ability to capture temporal correlations in network traffic makes it a valuable tool for IDS deployment in connected vehicles.



**Fig 5.1.4:** RNN Model

**Random Forest model** is a machine learning-based approach that applies an ensemble of decision trees to classify CAN messages as either normal or attack. Each decision tree in the forest learns a different subset of features, improving robustness and reducing overfitting. The ensemble learning approach averages predictions from multiple trees, enhancing overall classification accuracy. A key advantage of the Random Forest model is its ability to perform feature importance analysis, which helps identify key attributes contributing to attack detection. Unlike deep learning models that require extensive computational resources, Random Forest is lightweight and efficient, making it ideal for real-time IDS deployment in resource-constrained environments such as embedded vehicular systems. While it may not capture complex temporal dependencies as effectively as deep learning models, it remains a reliable and interpretable solution for intrusion detection in automotive cybersecurity.



**Fig 5.1.5: Random Forest**

## 5.2 Modules

### 1. Preprocessing Module

This module is responsible for cleaning and preparing CAN traffic data before feeding it into machine learning models. It ensures that the dataset is optimized for high accuracy and efficiency in intrusion detection.

#### Data Cleaning:

- Removes duplicate and corrupted CAN messages.
- Handles missing values through imputation or removal.

**Feature Transformation:**

- Converts CAN ID and Payload Bytes from hexadecimal to numerical format.
- Applies Label Encoding to classify messages as normal or attack (0 for normal, 1 for attack).

**Normalization & Scaling:**

- Uses StandardScaler to standardize numerical values.
- Applies Quantile Transformation for better model convergence.

**Feature Engineering:**

- Timestamp Difference Calculation to capture irregular message intervals.
- Payload Statistics such as sum and standard deviation to detect anomalies.

**Dataset Splitting:**

- Divides the dataset into training (80%) and testing (20%) sets for model evaluation.

**2. System Module**

This module is responsible for training and deploying the Intrusion Detection System (IDS) using deep learning and machine learning models.

**Model Training & Development:**

- The IDS incorporates Hybrid CNN-LSTM, Hybrid Transfer Learning, RNN, and Random Forest models.
- The dataset is divided into training, validation, and testing sets to ensure generalizability.
- Each model is trained separately and evaluated for performance.

**Feature Selection & Processing:**

- Key features include CAN ID, payload bytes, timestamp variations, and message length.

- Feature importance analysis is conducted to determine the most influential attributes in attack classification.

### Prediction & Classification:

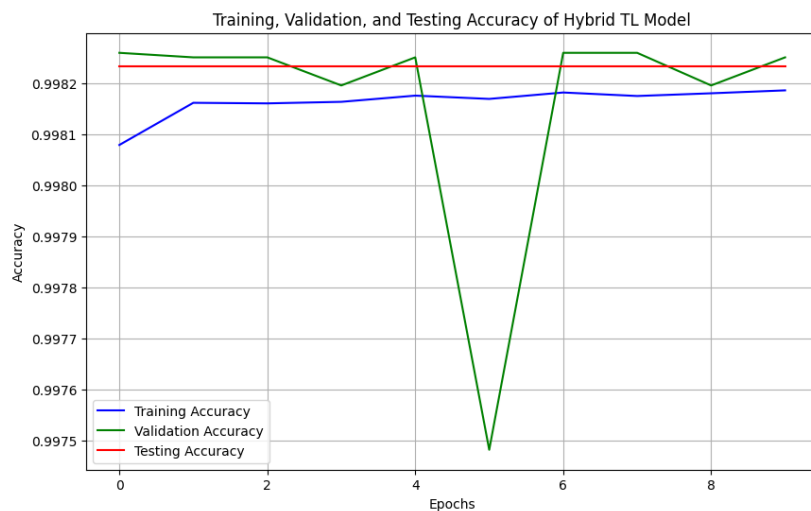
- The trained models classify incoming CAN messages as either normal or attack.
- The majority voting mechanism in Random Forest aggregates results from multiple decision trees for improved accuracy.

### Performance Evaluation:

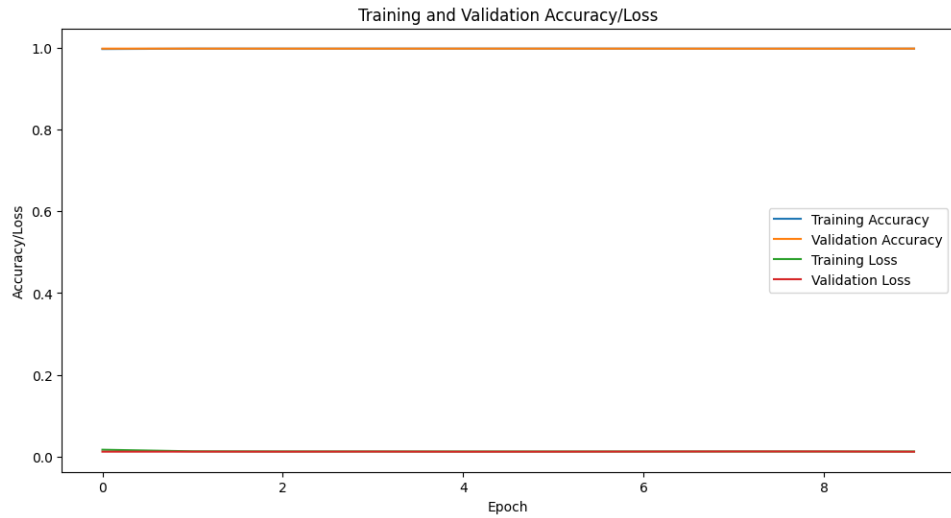
- The models are assessed using metrics such as accuracy, precision, recall, and F1-score.
- Confusion matrices are used to visualize detection performance.

### Optimization & Fine-Tuning:

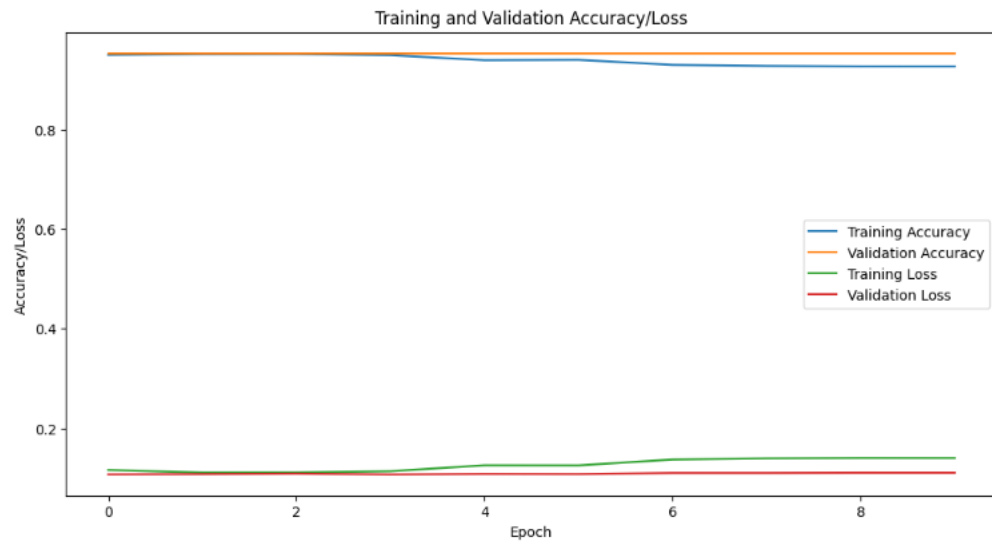
- Hyperparameter tuning techniques are applied to improve model performance.
- Models are optimized for low false positives and real-time attack detection.



**Figure 5.2.1:** Training, Validation, and Testing Accuracy of Hybrid TL Model



**Fig 5.2.2:** Training and validation/Loss for Fuzzy attack.



**Fig 5.2.3:** Training and validation/Loss for RPM attack.

### 3. User Interface (UI) Module

This module provides an interactive interface for monitoring intrusion detection results and analyzing attack patterns.

**Data Input Forms:** Allows users to manually input CAN messages for classification.

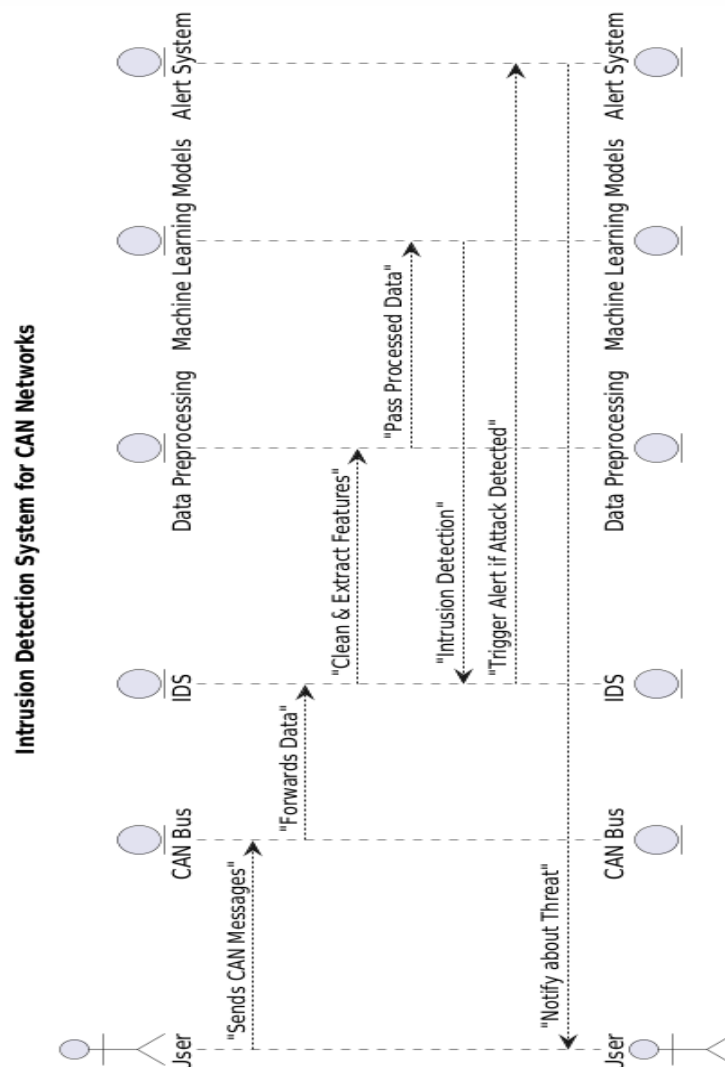
**Prediction Display:** Shows real-time detection results with indicators for normal vs. attack messages.

**Analytics Dashboard:** Provides visualizations such as graphs, tables, and heatmaps for attack trend analysis.

#### Alert & Report Generation:

- Triggers alerts upon detecting anomalies in CAN traffic.
- Generates detailed reports on attack types, frequencies, and sources for further investigation.

### 5.3 UML Diagrams



**Fig 5.3.1:** UML Sequence Diagram

## 6. IMPLEMENTATION

### 6.1 Model Implementation

#### Model Architecture for hybrid TL model:

```
from tensorflow.keras.layers import Input, Dense, Dropout, Flatten, Reshape
from tensorflow.keras.models import Model

# Define the input
input_layer = Input(shape=(X_train.shape[1], 1))

# Add 1D CNN layers
cnn_out = Conv1D(32, 3, activation='relu', padding='same')(input_layer)
cnn_out = Conv1D(64, 3, activation='relu', padding='same')(cnn_out)
cnn_out = MaxPooling1D(pool_size=1)(cnn_out)

# Flatten the output from the CNN layers
cnn_out = Flatten()(cnn_out)

# Add fully connected layers
dense_out = Dense(50, activation='relu')(cnn_out)
dense_out = Dropout(0.5)(dense_out)

# Handle NaN values and convert 'y' to a consistent type (e.g., string)
y_processed = np.where(pd.isna(y), 'NaN', y).astype(str) # Replace NaNs with 'NaN'
string and convert to string type

# Now calculate unique classes
num_classes = len(np.unique(y_processed))

output_layer = Dense(num_classes, activation='softmax')(dense_out)

# Create the model
fuzzy_tl_model = Model(inputs=input_layer, outputs=output_layer)

# Compile the model
fuzzy_tl_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Display model summary
fuzzy_tl_model.summary()
```

```

y_train_encoded = np.argmax(y_train, axis=1)
y_val_encoded = np.argmax(y_val, axis=1)
# The rest of the code remains the same
y_train_categorical = to_categorical(y_train_encoded)
y_val_categorical = to_categorical(y_val_encoded)
# Train the model
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
# Change the file extension to '.keras'
model_checkpoint = ModelCheckpoint('fuzzy_tl.keras', monitor='val_accuracy',
save_best_only=True)
history = fuzzy_tl_model.fit(
    X_train, y_train_categorical,
    epochs=10,
    batch_size=128,
    validation_data=(X_val, y_val_categorical),
    callbacks=[early_stopping, model_checkpoint]
)

# Evaluate the model on the test set
loss, accuracy = fuzzy_tl_model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')

# Make predictions on the test set
y_pred = fuzzy_tl_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)
# Generate classification report
print(classification_report(y_test_classes, y_pred_classes))
# Generate confusion matrix

```



```

cm = confusion_matrix(y_test_classes, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

```

## 6.2 Coding

### Preprocessing

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import QuantileTransformer, StandardScaler
from scipy.stats import rankdata
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense,
Dropout, Flatten

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

from tensorflow.keras.layers import Input, Reshape

from tensorflow.keras.models import Model

from sklearn.metrics import roc_curve, auc

```

```

import warnings

warnings.filterwarnings('ignore')

df=pd.read_csv('/content/drive/MyDrive/Fuzzy_dataset.csv')

df.columns = ['Timestamp', 'CAN ID', 'DLC', 'DATA[0]', 'DATA[1]', 'DATA[2]',
'DATA[3]', 'DATA[4]', 'DATA[5]', 'DATA[6]', 'DATA[7]', 'Flag']

df

df.drop_duplicates(inplace=True)

df.dropna(inplace=True)

df

df['DATA[0]'] = df['DATA[0]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[1]'] = df['DATA[1]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[2]'] = df['DATA[2]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[3]'] = df['DATA[3]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[4]'] = df['DATA[4]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[5]'] = df['DATA[5]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[6]'] = df['DATA[6]'].astype(str).apply(lambda x: x.split(' ')[0])
df['DATA[7]'] = df['DATA[7]'].astype(str).apply(lambda x: x.split(' ')[0])

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['DATA[0]'] = le.fit_transform(df['DATA[0]'])
df['DATA[1]'] = le.fit_transform(df['DATA[1]'])
df['DATA[2]'] = le.fit_transform(df['DATA[2]'])
df['DATA[3]'] = le.fit_transform(df['DATA[3]'])
df['DATA[4]'] = le.fit_transform(df['DATA[4]'])
df['DATA[5]'] = le.fit_transform(df['DATA[5]'])
df['DATA[6]'] = le.fit_transform(df['DATA[6]'])

```

```

df['DATA[7]'] = le.fit_transform(df['DATA[7]'])
df['Flag'] = le.fit_transform(df['Flag'])
for col in ['CAN ID', 'DATA[0]', 'DATA[1]', 'DATA[2]', 'DATA[3]', 'DATA[4]',
'DATA[5]', 'DATA[6]', 'DATA[7]']:
    df[col] = df[col].apply(lambda x: int(x, 16) if isinstance(x, str) else x)
df

```

```

images = []
for row in df.values:
    try:
        image = np.array(row[3:11], dtype=np.uint8).reshape(8, 8)
    except ValueError:
        image = np.zeros((8, 8), dtype=np.uint8)
    images.append(image)

```

```

images = np.array(images)
plt.imshow(images[0], cmap='gray')
plt.show()

```

```

images_2D = images.reshape(images.shape[0], -1)
scaler = StandardScaler()
scaler.fit(images_2D)
scaled_images = scaler.transform(images_2D)
print(scaled_images[0:5])

```

```

images_2D = images.reshape(images.shape[0], -1)
scaler = StandardScaler()
scaler.fit(images_2D)
scaled_images = scaler.transform(images_2D)

```

```

quantile_transformer = QuantileTransformer(output_distribution='normal')
quantile_transformer.fit(scaled_images)
normalized_images = quantile_transformer.transform(scaled_images)
print(normalized_images[0:5])

images = np.random.rand(100, 28, 28)
images_2D = images.reshape(images.shape[0], -1)
scaler = StandardScaler()
scaler.fit(images_2D)
scaled_images = scaler.transform(images_2D)
ranked_data = rankdata(scaled_images)
normalized_data = np.quantile(ranked_data, np.linspace(0, 1, len(ranked_data)))
print(normalized_data[0:5])

normal_df = df[df['Flag'] == 0]
attack_df = df[df['Flag'] == 1]
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

sns.heatmap(normal_df[['DATA[0]', 'DATA[1]', 'DATA[2]', 'DATA[3]', 'DATA[4]',
'DATA[5]', 'DATA[6]', 'DATA[7]']].corr(), annot=True, cmap='coolwarm', ax=axs[0])
axs[0].set_title('Normal Messages')

sns.heatmap(attack_df[['DATA[0]', 'DATA[1]', 'DATA[2]', 'DATA[3]', 'DATA[4]',
'DATA[5]', 'DATA[6]', 'DATA[7]']].corr(), annot=True, cmap='coolwarm', ax=axs[1])
axs[1].set_title('Attack Messages')

plt.tight_layout()
plt.show()

label_encoder = LabelEncoder()
df['Flag'] = label_encoder.fit_transform(df['Flag'])
df

```

```

df['Timestamp_diff'] = df['Timestamp'].diff()

df['CAN_ID_diff'] = df['CAN ID'].diff()

df['Payload_sum'] = df[['DATA[0]', 'DATA[1]', 'DATA[2]', 'DATA[3]', 'DATA[4]',
'DATA[5]', 'DATA[6]', 'DATA[7]']].sum(axis=1)

df['Payload_std'] = df[['DATA[0]', 'DATA[1]', 'DATA[2]', 'DATA[3]', 'DATA[4]',
'DATA[5]', 'DATA[6]', 'DATA[7]']].std(axis=1)

df

correlation_matrix = df.corr()

plt.figure(figsize=(12, 10))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix')

plt.show()

threshold = 0.1

relevant_features = correlation_matrix['Flag'][abs(correlation_matrix['Flag']) >
threshold].index.tolist()

print("Relevant features:", relevant_features)

X = df['CAN ID'].astype(str).values
y = df['Flag'].values

tokenizer = Tokenizer()

tokenizer.fit_on_texts(X)

X_seq = tokenizer.texts_to_sequences(X)

max_len = max(len(x) for x in X_seq)

X_padded = pad_sequences(X_seq, maxlen=max_len, padding='post')

X_padded = X_padded.reshape(X_padded.shape[0], X_padded.shape[1], 1)

le = LabelEncoder()

y = le.fit_transform(y)

y = to_categorical(y)

X_train, X_temp, y_train, y_temp = train_test_split(X_padded, y, test_size=0.3,
random_state=42)

```

```

X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)

print(f'X_train shape: {X_train.shape}')
print(f'X_val shape: {X_val.shape}')
print(f'X_test shape: {X_test.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'y_val shape: {y_val.shape}')
print(f'y_test shape: {y_test.shape}')

```

### **1. Model Architecture for hybrid CNN-LSTM model**

```

input_layer = Input(shape=(X_train.shape[1], 1))
cnn_out = Conv1D(32, 3, activation='relu', padding='same')(input_layer)
cnn_out = Conv1D(64, 3, activation='relu', padding='same')(cnn_out)
cnn_out = MaxPooling1D(pool_size=1)(cnn_out)
cnn_out = Flatten()(cnn_out)
lstm_out = Reshape((1, -1))(cnn_out)
lstm_out = LSTM(50, return_sequences=True)(lstm_out)
lstm_out = LSTM(50)(lstm_out)
dense_out = Dense(50, activation='relu')(lstm_out)
dense_out = Dropout(0.5)(dense_out)
y_processed = np.where(pd.isna(y), 'NaN', y).astype(str)
num_classes = len(np.unique(y_processed))
output_layer = Dense(num_classes, activation='softmax')(dense_out)
fuzzy_cnn_lstm_model = Model(inputs=input_layer, outputs=output_layer)
fuzzy_cnn_lstm_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
fuzzy_cnn_lstm_model.summary()

y_train_encoded = np.argmax(y_train, axis=1)
y_val_encoded = np.argmax(y_val, axis=1)

```

```

y_train_categorical = to_categorical(y_train_encoded)
y_val_categorical = to_categorical(y_val_encoded)

early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

model_checkpoint = ModelCheckpoint('fuzzy_cnn_lstm.keras',
monitor='val_accuracy', save_best_only=True)

history = fuzzy_cnn_lstm_model.fit(
    X_train, y_train_categorical,
    epochs=10,
    batch_size=128,
    validation_data=(X_val, y_val_categorical),
    callbacks=[early_stopping, model_checkpoint]
)

loss, accuracy = fuzzy_cnn_lstm_model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')

y_pred = fuzzy_cnn_lstm_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)
print(classification_report(y_test_classes, y_pred_classes))
cm = confusion_matrix(y_test_classes, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')

```

```

plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Accuracy/Loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy/Loss')
plt.legend()
plt.show()

```

## 2. Model Architecture for hybrid TL model

```

input_layer = Input(shape=(X_train.shape[1], 1))
cnn_out = Conv1D(32, 3, activation='relu', padding='same')(input_layer)
cnn_out = Conv1D(64, 3, activation='relu', padding='same')(cnn_out)
cnn_out = MaxPooling1D(pool_size=1)(cnn_out)
cnn_out = Flatten()(cnn_out)
dense_out = Dense(50, activation='relu')(cnn_out)
dense_out = Dropout(0.5)(dense_out)
y_processed = np.where(pd.isna(y), 'NaN', y).astype(str)
num_classes = len(np.unique(y_processed))
output_layer = Dense(num_classes, activation='softmax')(dense_out)
fuzzy_tl_model = Model(inputs=input_layer, outputs=output_layer)
fuzzy_tl_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
fuzzy_tl_model.summary()

y_train_encoded = np.argmax(y_train, axis=1)
y_val_encoded = np.argmax(y_val, axis=1)
y_train_categorical = to_categorical(y_train_encoded)
y_val_categorical = to_categorical(y_val_encoded)
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

```



```

model_checkpoint = ModelCheckpoint('fuzzy_tl.keras', monitor='val_accuracy',
save_best_only=True)

history = fuzzy_tl_model.fit(
    X_train, y_train_categorical,
    epochs=10,
    batch_size=128,
    validation_data=(X_val, y_val_categorical),
    callbacks=[early_stopping, model_checkpoint]
)

loss, accuracy = fuzzy_tl_model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
y_pred = fuzzy_tl_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)
print(classification_report(y_test_classes, y_pred_classes))
cm = confusion_matrix(y_test_classes, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')

```

```

plt.title('Training and Validation Accuracy/Loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy/Loss')
plt.legend()
plt.show()

```

### 3. RNN Model

```

input_layer = Input(shape=(X_train.shape[1], 1))
lstm_out = LSTM(50, return_sequences=True)(input_layer)
lstm_out = LSTM(50)(lstm_out)
dense_out = Dense(50, activation='relu')(lstm_out)
dense_out = Dropout(0.5)(dense_out)
y_processed = np.where(pd.isna(y), 'NaN', y).astype(str)
num_classes = len(np.unique(y_processed))
output_layer = Dense(num_classes, activation='softmax')(dense_out)
fuzzy_rnn_model = Model(inputs=input_layer, outputs=output_layer)
fuzzy_rnn_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
fuzzy_rnn_model.summary()

y_train_encoded = np.argmax(y_train, axis=1)
y_val_encoded = np.argmax(y_val, axis=1)
y_train_categorical = to_categorical(y_train_encoded)
y_val_categorical = to_categorical(y_val_encoded)
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
model_checkpoint = ModelCheckpoint('fuzzy_rnn.keras', monitor='val_accuracy',
save_best_only=True)
history = fuzzy_rnn_model.fit(
    X_train, y_train_categorical,
    epochs=10,

```

```

    batch_size=64,
    validation_data=(X_val, y_val_categorical),
    callbacks=[early_stopping, model_checkpoint]
)

loss, accuracy = fuzzy_rnn_model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
y_pred = fuzzy_rnn_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)
print(classification_report(y_test_classes, y_pred_classes))
cm = confusion_matrix(y_test_classes, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Accuracy/Loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy/Loss')
plt.legend()
plt.show()

```

## Deployment Code Using Flask

### App.py

```
from flask import Flask, request, jsonify, render_template

import numpy as np

from tensorflow.keras.models import load_model

from datetime import datetime

# Load the trained model

MODEL_PATH = 'fuzzy_cnn_lstm.keras'

model = load_model(MODEL_PATH)

# Define Flask app

app = Flask(__name__)

# Class mapping: 0 = Normal, all others = Attack

class_mapping = {

    0: "Normal",

    1: "Attack",

    2: "Attack",

    3: "Attack",

    4: "Attack"

}

# History list to store previous predictions

prediction_history = []

@app.route('/')

def home():

    return render_template('index.html', history=prediction_history)

@app.route('/about')

def about():

    return render_template('about.html')

@app.route('/evaluation')
```

```

def evaluation():
    return render_template('evaluation.html')

@app.route('/flowcharts')
def flowcharts():
    return render_template('flowcharts.html')

@app.route('/predictions', methods=['GET', 'POST'])
def predictions():
    if request.method == 'POST':
        try:
            # Parse input data
            input_data = request.json
            print(f'Received input data: {input_data}') # Debug log
            can_id = input_data.get('can_id')
            data_bytes = input_data.get('data')
            # Validate input
            if not can_id or len(data_bytes) != 8:
                return jsonify({'error': 'Invalid input. Provide a valid CAN ID and 8 data
bytes.'}), 400

            # Convert CAN ID to integer (hex or decimal)
            can_id_int = int(can_id, 16) if can_id.startswith('0x') else int(can_id)
            # Convert data bytes to integers (hex or decimal)
            parsed_data_bytes = []
            for byte in data_bytes:
                if isinstance(byte, str) and byte.startswith('0x'):
                    parsed_data_bytes.append(int(byte, 16)) # Convert hex to int
                else:
                    parsed_data_bytes.append(int(byte)) # Convert decimal to int
            # Combine CAN ID and data bytes into an input sequence

```

```

input_sequence = [can_id_int] + parsed_data_bytes
# Predict for each element individually
predictions = []
for value in input_sequence:
    input_array = np.array(value, dtype=np.float32).reshape((1, 1, 1))
    prediction = model.predict(input_array)
    predictions.append(prediction)
# Aggregate predictions (e.g., majority voting)
aggregated_prediction = np.argmax(np.mean(predictions, axis=0))
confidence = float(np.max(np.mean(predictions, axis=0)))
# Map prediction to attack type
attack_status = "Normal" if aggregated_prediction == 0 else "Attack"
# Get current timestamp
timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
# Store prediction in history
prediction_history.append({
    'timestamp': timestamp,
    'can_id': can_id,
    'data': ' '.join(map(str, parsed_data_bytes)), # Ensure the data is joined as a
string
    'prediction': attack_status,
    'confidence': f'{confidence:.2f}'
})
# Return prediction with timestamp
return jsonify({
    'timestamp': timestamp,
    'prediction': attack_status,
    'confidence': f'{confidence:.2f}',
    'data': {

```

```

        'can_id': can_id,
        'data_bytes': parsed_data_bytes
    }
})

except Exception as e:
    return jsonify({'error': str(e)}), 500

return render_template('predictions.html', history=prediction_history)

if __name__ == '__main__':
    app.run(debug=True)

```

### **base.html**

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title> {% block title %} Adaptive Intrusion Detection in CAN-Based Vehicular
Networks Using Transfer Learning for Evolving Threats {% endblock %} </title>

    <style>

    body {

        position: relative;

        font-family: Arial, sans-serif;

        color: #333;

        line-height: 1.6;

        margin: 0;

        padding: 0;

        background-color: #f4f4f4;

    }

    header {

```

```
background-color: #121312;
color: #fff;
padding: 20px 10px;
text-align: center;
position: relative;
}
.logo {
  position: absolute;
  top: 10px;
  left: 20px;
}
.logo img {
  width: 120px;
  height: auto;
}
.header-text h1 {
  font-size: 2em;
  margin: 10px 0;
}
.team-members {
  font-size: 1.3em;
  color: #ccc;
}
.navigation {
  background-color: #222;
  padding: 2px 0;
  display: flex;
  justify-content: center;
}
```



```
.navigation ul {  
    list-style: none;  
    padding: 0;  
    margin: 0;  
    display: flex;  
    gap: 20px;  
}  
  
.navigation ul li {  
    display: inline;  
}  
  
.navigation ul li a {  
    text-decoration: none;  
    color: #fff;  
    font-weight: bold;  
    padding: 10px 20px;  
    display: flex;  
    align-items: center;  
    gap: 8px;  
    transition: all 0.3s ease;  
    border-radius: 5px;  
}  
  
.navigation ul li a:hover {  
    background-color: #444;  
}  
  
main {  
    padding: 20px;  
}  
  
</style>
```

```

<script src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>

</head>

<body>

  <header>

    <div class="logo">

    </div>

    <div class="header-text">

      <h1>Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using
Transfer Learning for Evolving Threats</h1>

      <p class="team-members">

        <strong>Team:</strong> Sk. Ijaz Ahamad | S. V. S. Girish | T. Nangrandhra
Babu | <strong>Guide:</strong> D. Venkata Reddy | <strong>Mentor:</strong> K. V.
Narasimha Reddy

      </p>

    </div>

    <nav class="navigation">

      <ul>

        <li><a href="{{ url_for('home') }}"><i class="fas fa-home"></i>
Home</a></li>

        <li><a href="{{ url_for('about') }}"><i class="fas fa-info-circle"></i>
About</a></li>

        <li><a href="{{ url_for('predictions') }}"><i class="fas fa-chart-line"></i>
Predictions</a></li>

        <li><a href="{{ url_for('evaluation') }}"><i class="fas fa-tasks"></i>
Evaluation</a></li>

        <li><a href="{{ url_for('flowcharts') }}"><i class="fas fa-project-
diagram"></i> Flowcharts</a></li>

      </ul>

    </nav>

  </header>

```

```

<main>

    {% block content %} {% endblock %}

</main>

</body>

</html>

```

### **predications.html**

```

{% extends "base.html" %}

{% block title %}Predictions - Adaptive IDS{% endblock %}

{% block content %}

<style>

    body {

        background: url('{{ url_for('static', filename='images/bg1.jpg') }}') no-repeat
        center center fixed;

        background-size: cover;

        color: white;

        font-family: Arial, sans-serif;

        text-align: center;

    }

    .container {

        background: rgba(0, 0, 0, 0.7);

        padding: 20px;

        border-radius: 10px;

        max-width: 800px; /* Increased width to accommodate two tables */

        margin: 50px auto;

    }

    h2 {

        color: #78e4e7;

```

```

}
input, button {
    padding: 10px;
    margin: 10px;
    border-radius: 5px;
    border: none;
}
button {
    background-color: #78e4e7;
    color: black;
    cursor: pointer;
}
table {
    width: 100%;
    margin-top: 20px;
    border-collapse: collapse;
    color: white;
}
th, td {
    border: 1px solid white;
    padding: 10px;
    text-align: center;
}
th {
    background-color: #78e4e7;
    color: black;
}
.error-message {
    color: red;

```

```

        font-weight: bold;
    }

    .table-container {
        display: flex;
        justify-content: space-between;
        gap: 20px; /* Space between the two tables */
    }

    .table-container table {
        width: 48%; /* Each table takes up roughly half the container width */
    }
</style>
<div class="container">
    <h2>Prediction Form</h2>

    <form id="predictionForm">
        <label for="can_id">CAN ID:</label>
        <input type="text" id="can_id" name="can_id" placeholder="0x123" required>
        <br>
        <label for="data">Data Bytes:</label>
        <input type="text" id="data" name="data" placeholder="01 0x2A 45 0x1F 10
0xFF 0x00 99" required>
        <br>
        <button type="submit">Predict</button>
    </form>

    <div id="predictionResult"></div>
    <h2>Valid Predictions</h2>
    <table id="validTable">
        <thead>
            <tr>

```

```

        <th>Timestamp</th>
        <th>CAN ID</th>
        <th>Data Bytes</th>
        <th>Prediction</th>
        <th>Confidence</th>
    </tr>
</thead>
<tbody>
    {% for entry in history %}
        {% if not entry.error %}
            <tr>
                <td>{{ entry.timestamp }}</td>
                <td>{{ entry.can_id }}</td>
                <td>{{ entry.data }}</td>
                <td>{{ entry.prediction }}</td>
                <td>{{ entry.confidence }}%</td>
            </tr>
        {% endif %}
    {% endfor %}
</tbody>
</table>

```

## <h2>Invalid Predictions</h2>

```

<table id="invalidTable">
    <thead>
        <tr>
            <th>Timestamp</th>
            <th>CAN ID</th>
            <th>Data Bytes</th>

```

```

        <th>Error Message</th>

    </tr>

</thead>

<tbody>

    {% for entry in history %}

        {% if entry.error %}

            <tr>

                <td>{{ entry.timestamp }}</td>

                <td>{{ entry.can_id }}</td>

                <td>{{ entry.data }}</td>

                <td class="error-message">{{ entry.error }}</td>

            </tr>

        {% endif %}

    {% endfor %}

</tbody>

</table>

</div>

<script>

document.addEventListener("DOMContentLoaded", function () {

    document.getElementById('predictionForm').addEventListener('submit', function
(e) {

        e.preventDefault(); // Prevent form submission

        var can_id = document.getElementById('can_id').value.trim();

        var data = document.getElementById('data').value.trim();

        // Validate CAN ID (must start with 0x and be a valid hexadecimal)

        var canIdPattern = /^0x[0-9A-Fa-f]+$/;

        if (!canIdPattern.test(can_id)) {

            addErrorRow(can_id, data, "Invalid CAN ID! It must start with '0x' and be a
valid hexadecimal (e.g., 0x1A).");

        }

        return;

    });

});

```

```

    }

    // Validate Data Bytes (must be exactly 8 values)

    var dataArray = data.split(/\s+/);

    if (dataArray.length !== 8) {

        addErrorRow(can_id, data, "Invalid Data Bytes! You must enter exactly 8
space-separated values (decimal or hex).");

        return;

    }

    // Validate each data byte (must be decimal 0-255 or hexadecimal 00-FF, 0x00-
0xFF)

    var validHexOrDec = /^((0x[0-9A-Fa-f]{1,2})|([A-Fa-f0-9]{2})|([0-9]{1,3}))$/;

    var parsedData = [];

    for (var i = 0; i < dataArray.length; i++) {

        let value = dataArray[i];

        if (!validHexOrDec.test(value)) {

            addErrorRow(can_id, data, "Invalid Data Byte at position " + (i + 1) + "!
Each value must be a valid decimal (0-255) or hexadecimal (00-FF, 0x00-0xFF).");

            return;

        }

        // Convert hex to decimal if necessary

        if (value.startsWith("0x")) {

            parsedData.push(parseInt(value, 16)); // Convert hex (0xNN) to decimal

        } else if (/^[A-Fa-f0-9]{2}$/.test(value)) {

            parsedData.push(parseInt(value, 16)); // Convert hex (NN) to decimal

        } else {

            let numValue = parseInt(value, 10);

            if (numValue < 0 || numValue > 255) {

                addErrorRow(can_id, data, "Invalid Data Byte at position " + (i + 1) + "!
Decimal values must be between 0-255.");

                return;

            }

        }

    }

```



```

        parsedData.push(numValue);
    }
}

// Prepare request data
var requestData = {
    can_id: can_id,
    data: parsedData
};

fetch('/predictions', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestData)
})
.then(response => response.json())
.then(data => {
    if (data.error) {
        addErrorRow(can_id, data.data.data_bytes.join(' '), data.error);
    } else {
        addValidRow(data.timestamp, data.data.can_id, data.data.data_bytes.join(' '), data.prediction, data.confidence);
    }
    document.getElementById('can_id').value = "";
    document.getElementById('data').value = "";
})
.catch(error => {
    addErrorRow(can_id, data, "Error: " + error.message);
});

```

```

    });
});

function addValidRow(timestamp, can_id, data, prediction, confidence) {
    const table =
document.getElementById('validTable').getElementsByTagName('tbody')[0];

    const newRow = table.insertRow();

    newRow.insertCell(0).textContent = timestamp;
    newRow.insertCell(1).textContent = can_id;
    newRow.insertCell(2).textContent = data;
    newRow.insertCell(3).textContent = prediction;
    newRow.insertCell(4).textContent = confidence + "%";
}

function addErrorRow(can_id, data, errorMessage) {
    const table =
document.getElementById('invalidTable').getElementsByTagName('tbody')[0];

    const newRow = table.insertRow();

    newRow.insertCell(0).textContent = new Date().toLocaleString();
    newRow.insertCell(1).textContent = can_id;
    newRow.insertCell(2).textContent = data;
    const errorCell = newRow.insertCell(3);
    errorCell.textContent = errorMessage;
    errorCell.classList.add('error-message'); // Add error styling
}

});
</script>

{% endblock %}

```

## 7. TESTING

### Unit Testing:

#### Purpose

Unit testing is performed to evaluate individual components of the Intrusion Detection System (IDS) and ensure that each function operates correctly. The goal is to validate the correctness of preprocessing, feature selection, model prediction, and system alerts.

#### Testing Areas

1. Data Preprocessing Functions
  - Verify handling of missing values, normalization, and feature extraction.
  - Ensure that CAN IDs and Data Bytes are correctly converted from hexadecimal to numerical values.
  - Validate that timestamp-based features (e.g., time difference between messages) are correctly calculated.
2. Feature Selection & Scaling
  - Test whether selected features (e.g., CAN ID, DLC, Data Bytes) impact model performance.
  - Ensure that Min-Max Scaling and Standardization are applied correctly for numerical features.
  - Validate that categorical features (attack labels) are properly encoded.
3. Model Training & Prediction
  - Evaluate whether Hybrid TL, CNN-LSTM, RNN, and Random Forest models are learning from the dataset correctly.
  - Verify model accuracy, loss, and effectiveness by running training on different subsets of data.
  - Ensure that predictions are within the expected range (Attack/Normal) with high confidence scores.
4. Overfitting & Underfitting Checks
  - Run cross-validation to ensure that models generalize well to unseen data.
  - Validate that the model is not memorizing specific attack patterns (overfitting).
  - Verify that sufficient data augmentation or balancing techniques are applied to avoid underfitting.

## 5. Tools Used

- Python's unittest & PyTest for function-level testing.
- Scikit-learn's cross-validation techniques to evaluate machine learning models.
- TensorFlow & Keras evaluation metrics for deep learning-based model validation.

Valid Predictions				
Timestamp	CAN ID	Data Bytes	Prediction	Confidence
2025-03-15 10:41:51	0x123	10 11 12 13 14 15 244 46	Normal	0.67%
2025-03-15 10:50:21	0x123	1 2 3 4 5 6 7 8	Normal	0.89%
2025-03-15 10:52:00	123	1 2 3 4 5 6 7 8	Normal	0.89%
2025-03-15 10:59:45	0x123	1 42 69 31 16 255 0 153	Attack	0.67%
2025-03-15 11:00:14	0x1A	0 1 2 3 4 5 6 7	Normal	0.95%
2025-03-15 11:00:32	0xFF	255 254 253 252 251 250 249 248	Attack	1.00%
2025-03-15 11:00:52	0xABC	161 178 195 212 229 246 18 52	Attack	0.89%
2025-03-15 11:01:12	0x7E	126 127 128 129 130 131 132 133	Attack	1.00%

**Fig 7.1:** Test Case-1 (Valid Input Predication)

Invalid Predictions			
Timestamp	CAN ID	Data Bytes	Error Message
3/15/2025, 10:57:23 AM	ox	1	Invalid CAN ID! It must start with '0x' and be a valid hexadecimal (e.g., 0x1A).
3/15/2025, 11:01:41 AM	0xGHI	01 0x2A 45 0x1F 10 0xFF 0x00 99	Invalid CAN ID! It must start with '0x' and be a valid hexadecimal (e.g., 0x1A).
3/15/2025, 11:01:59 AM	123	01 0x2A 45 0x1F 10 0xFF 0x00 99	Invalid CAN ID! It must start with '0x' and be a valid hexadecimal (e.g., 0x1A).
3/15/2025, 11:02:27 AM	0x123	01 0x2A 45 0x1F 10 0xFF 0x00 99 0xAA	Invalid Data Bytes! You must enter exactly 8 space-separated values (decimal or hex).
3/15/2025, 11:02:48 AM	0x123	01 0x2A 45 0x1F 10 0xFF 0x00 256	Invalid Data Byte at position 8! Decimal values must be between 0-255.
3/15/2025, 11:03:09 AM	0x123	01 0x2A 45 0x1F 10 0xFF 0x00 -1	Invalid Data Byte at position 8! Each value must be a valid decimal (0-255) or hexadecimal (00-FF, 0x00-0xFF).

**Fig 7.2:** Test Case-2 (Invalid Input Predication)

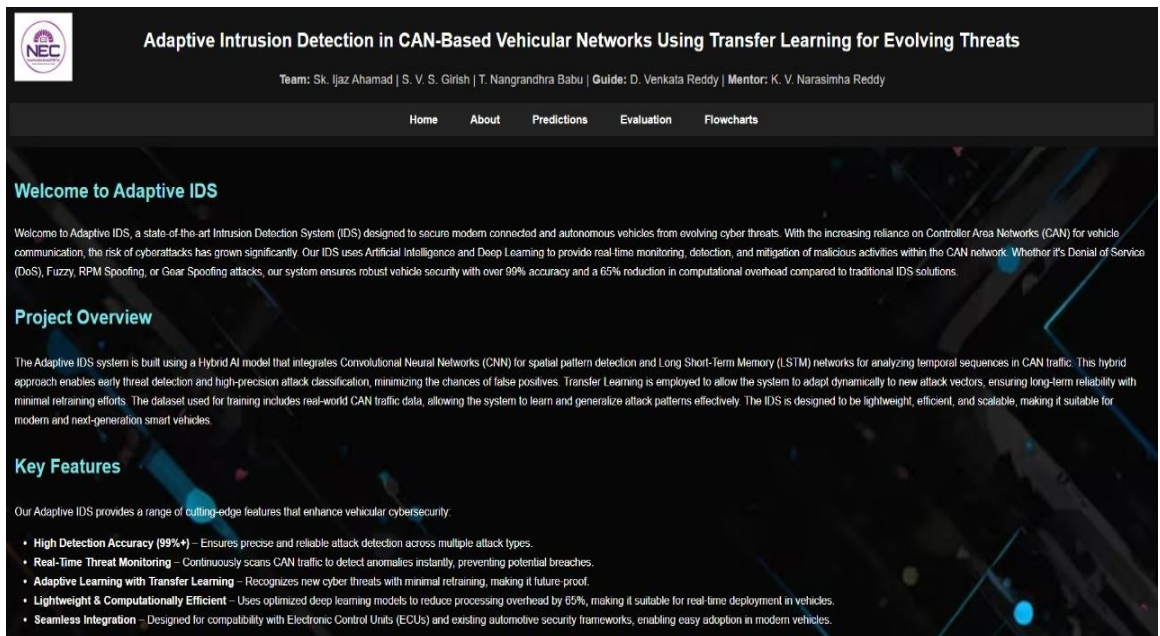
## System Testing:

**Purpose:** To validate the overall functionality of the IDS across different attack scenarios.

### Testing Areas:

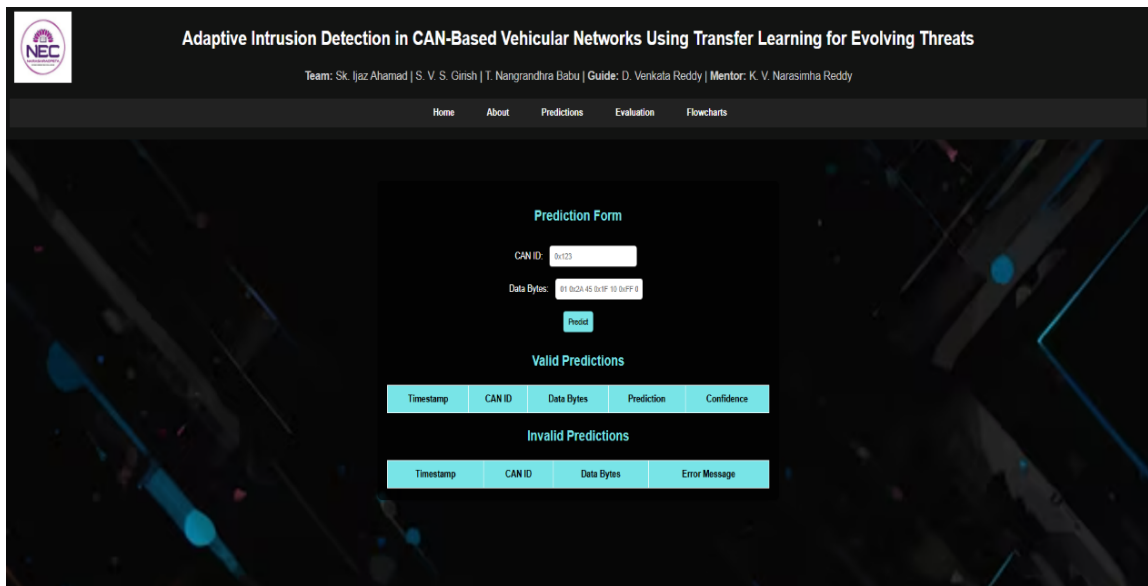
- End-to-end integration of Preprocessing Module, Detection Module (Hybrid CNN-LSTM, RNN, Random Forest), and Alert System.
- Performance evaluation based on Accuracy, Precision, Recall, and F1-score.
- Real-time attack detection efficiency.
- User interface responsiveness and alert system validation.

**Tools Used:** Flask Testing Environment, Postman for API Testing, Selenium for UI Testing.



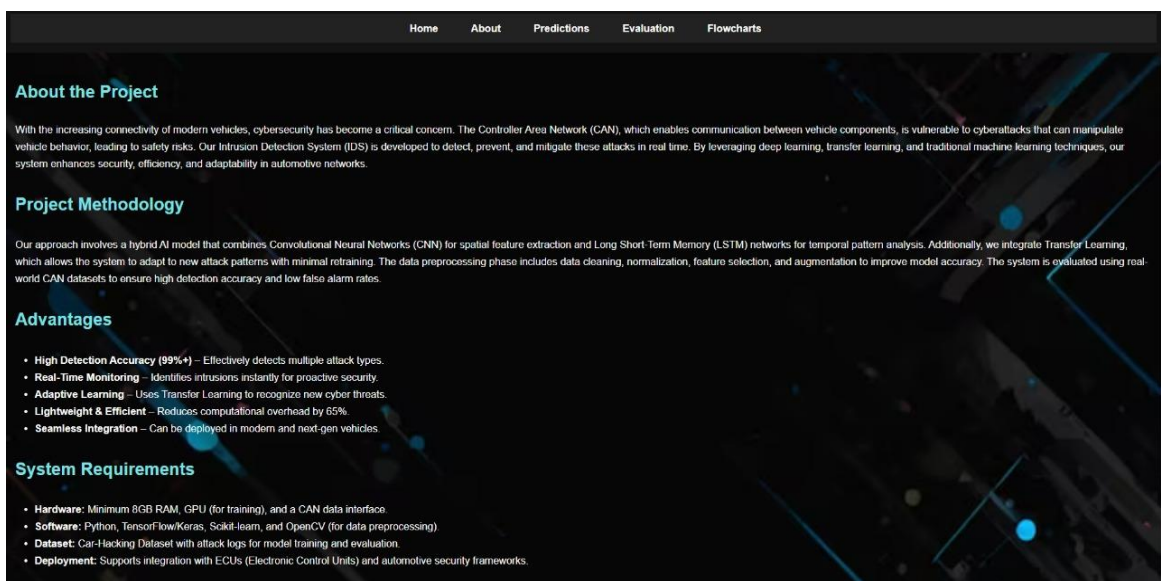
**Fig 7.3: Home Screen**

- The home screen includes navigation to intrusion detection, logs, and model evaluation.



**Fig 7.4: Prediction Screen**

- User interface where CAN bus messages can be tested against the IDS models.



**Fig 7.5: About Screen**

- Displays system architecture, dataset information, and project contributors.

## **Integration Testing:**

**Purpose:** To ensure seamless interaction between different IDS modules and validate data flow.

### **Testing Areas:**

- **Data Flow:** From raw CAN messages to Preprocessing, Model Classification, and Alert Generation.
- **User Interaction:** System response time when processing incoming CAN messages.
- **Final Prediction Accuracy:** Ensuring smooth model predictions based on real-world scenarios.

**Tools Used:** PyTest, Integration Testing Frameworks.

## 8. RESULT ANALYSIS

The performance of the Intrusion Detection System (IDS) for CAN-based vehicular networks was evaluated using multiple deep learning and machine learning models, including Hybrid Transfer Learning (TL), Hybrid CNN-LSTM, RNN, and Random Forest models. The results were analyzed based on key performance metrics such as accuracy, precision, recall, F1-score, and ROC AUC across different attack types, including Fuzzy Attack, RPM Attack, Gear Attack, and DoS Attack.

### Performance Comparison:

The results indicate that the Hybrid CNN-LSTM and Hybrid TL models consistently achieved high detection rates, demonstrating their effectiveness in detecting CAN-based cyber-attacks. The Fuzzy Attack dataset yielded the highest accuracy of 99.82%, while the DoS attack achieved 100% accuracy, indicating that the model perfectly classified attack and normal messages in this scenario. However, the RPM and Gear Attacks had slightly lower performance, with an accuracy of 95.31% and 95.33%, respectively.

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Hybrid TL	0.9982	0.9982	0.9982	0.9982	0.9951
Hybrid CNN-LSTM	0.9982	0.9982	0.9982	0.9982	0.9950
RNN	0.9982	0.9982	0.9982	0.9982	0.9950

**Fig 8.1:** Fuzzy Attack Performance

Model	Accuracy	Precision	Recall	F1-Score
Hybrid TL	0.9531	0.7529	1.0000	0.8590
Hybrid CNN-LSTM	0.9531	0.7529	1.0000	0.8590
Random Forest	0.9531	0.7529	1.0000	0.8590

**Fig 8.2:** RPM Attack Performance

Model	Accuracy	Precision	Recall	F1-Score
Hybrid TL	0.9533	0.7449	1.0000	0.8538
Hybrid CNN-LSTM	0.9533	0.7449	1.0000	0.8538
RNN	0.9533	0.7449	1.0000	0.8538

**Fig 8.3:** Gear Attack Performance



Attack Type	Accuracy	Precision	Recall	F1-Score
Fuzzy Attack	0.9982	0.9982	0.9982	0.9982
RPM Attack	0.9531	0.7529	1.0000	0.8590
Gear Attack	0.9533	0.7449	1.0000	0.8538
DoS Attack	1.0000	1.0000	1.0000	1.0000

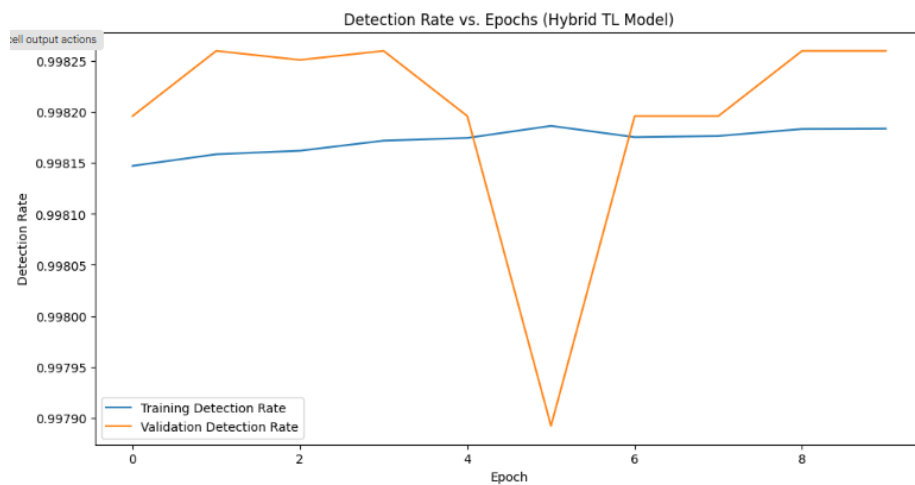
**Fig 8.4:** Hybrid CNN-LSTM Model Performance Across Various Attacks

### Detection Performance Across Attacks:

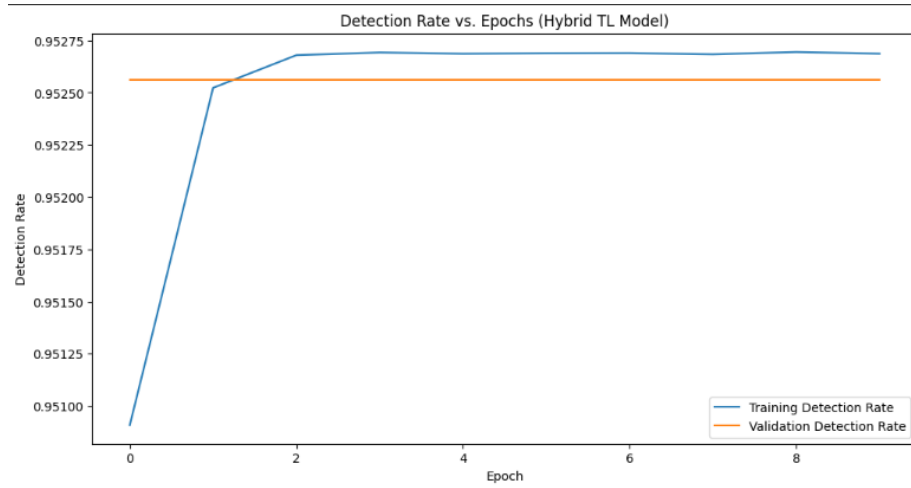
The Hybrid CNN-LSTM model demonstrated superior results across all attack types. The DoS attack dataset resulted in the best performance, achieving 100% accuracy, precision, recall, and F1-score, indicating that the model was highly effective in identifying malicious messages injected into the CAN bus.

The Fuzzy Attack dataset also performed exceptionally well, with an accuracy of 99.82%, precision of 99.82%, and recall of 99.82%, suggesting that the model accurately detected anomalies in CAN traffic.

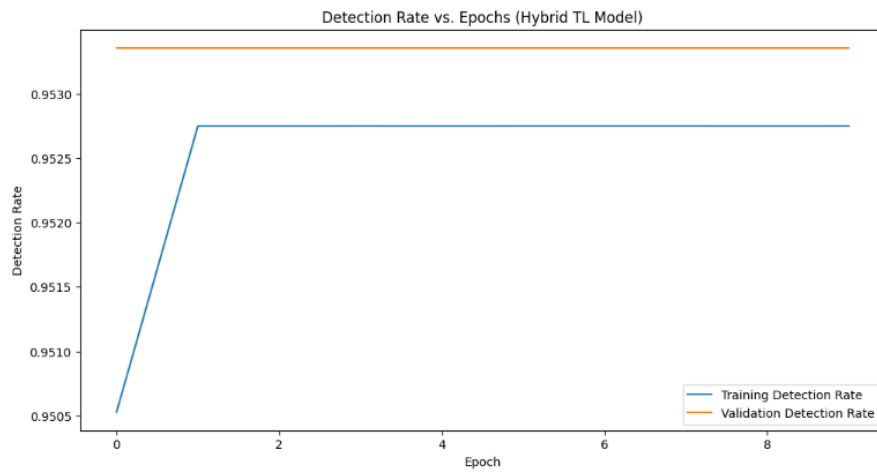
On the other hand, the RPM and Gear Attack datasets showed slightly lower detection performance due to the similarity between normal and attack patterns, making it challenging to distinguish between them. The RPM Attack dataset had a recall of 100% but a precision of 75.29%, meaning that while all attack instances were detected, there were some false positives.



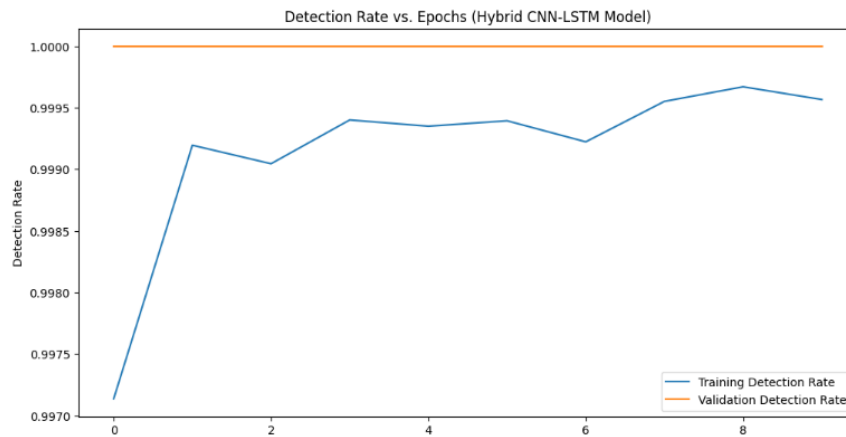
**Fig 8.5:** Detection Rate vs. Epochs (Hybrid TL Model) - Fuzzy Attack



**Fig 8.6:** Detection Rate vs. Epochs (Hybrid TL Model) - RPM Attack



**Fig 8.7:** Detection Rate vs. Epochs (Hybrid TL Model) - Gear Attack



**Fig 8.8:** Detection Rate vs. Epochs (Hybrid TL Model) - DoS Attack.

**Model Performance:**

- Hybrid TL Model performed well across all attack types, with an accuracy ranging from 95.31% to 99.82%, proving its reliability in real-time intrusion detection.
- Hybrid CNN-LSTM Model showed similar performance to the TL model, making it an excellent choice for CAN IDS deployment.
- RNN Model achieved comparable results to the CNN-LSTM model but required more computational resources, making it less optimal for real-time applications.
- Random Forest Model performed well for RPM attacks but lacked the ability to capture temporal dependencies compared to deep learning models.

## 9. CONCLUSION

Our work proposes an online reconfigurable Controller Area Network Intrusion Detection System that uses transfer learning to enhance security in vehicular environments. It provides improved detection accuracy for known as well as unknown threats by an impressive average detection rate of more than 99%. Hybrid CNN LSTM and RNN have found a great integrative capability with traditional techniques of machine learning for the adaptation of dynamic nature in cyber threats. The results of our experiments validate the suitability of our proposed IDS, which can keep high accuracy, precision, recall, and F1-score on various types of attacks. This research paper contributes to knowledge in automotive security and opens ways for further innovation in adaptive IDS solutions. We forward this paper as a recommendation to advance hybrid models and even make real-world testing in enhancing the detection capabilities of IDS frameworks, moving forward. With the continuous evolving nature of cyber threats, this kind of research and development is very important to ensure the safety and security of connected vehicles in an increasingly complex digital landscape.

## **10. FUTURE SCOPE**

The proposed Intrusion Detection System (IDS) can be further improved by adding advanced methods like unsupervised learning and federated learning to detect unknown attacks and use decentralized data while maintaining privacy. Expanding datasets with real-world scenarios will make the system more robust. Using lightweight models and edge computing can help deploy IDS in vehicles with limited hardware. Additionally, integrating the IDS with vehicle communication systems (like V2V and V2I) will provide stronger security for connected and autonomous vehicles. These steps will enhance the system's ability to handle evolving cyber threats effectively.

## 11. REFERENCES

- [1] N. Khatri, S. Lee, and S. Y. Nam, "Transfer Learning-Based Intrusion Detection System for a Controller Area Network," *IEEE Access*, vol. 11, pp. 120963-120981, Sept. 2023.
- [2] J. A. Khan, D. W. Lim, and Y. S. Kim, "A Deep Learning-Based IDS for Automotive Theft Detection for In-Vehicle CAN Bus," *IEEE Access*, vol. 11, pp. 112814-112827, Sept. 2023.
- [3] O. Y. Al-Jarrah, K. El Haloui, M. Dianati, and C. Maple, "A Novel Detection Approach of Unknown Cyber-Attacks for Intra-Vehicle Networks Using Recurrence Plots and Neural Networks," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 271-280, Mar. 2023.
- [4] M. S. Salek, P. K. Biswas, J. Pollard, J. Hales, Z. Shen, V. Dixit, M. C. Chowdhury, S. M. Khan, and Y. Wang, "A Novel Hybrid Quantum Classical Framework for an In-Vehicle Controller Area Network Intrusion Detection," *IEEE Access*, vol. 11, pp. 96081-96082, Aug. 2023.
- [5] D. Javeed, M. S. Saeed, I. Ahmad, P. Kumar, A. Jolfaei, and M. Tahir, "An Intelligent Intrusion Detection System for Smart Consumer Electronics Network," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 906-912, Oct. 2023.
- [6] M. R. Islam, M. Sahalabadi, K. Kim, Y. Kim, and K. Yim, "CF-AIDS: Comprehensive Frequency-Agnostic Intrusion Detection System on In-Vehicle Network," *IEEE Access*, vol. 12, pp. 13971-13981, Jan. 2024.
- [7] W. Xu, Y. Xu, Z. Wang, Y. Wu, and Y. Wang, "Intrusion Detection System for In-Vehicle CAN-FD Bus ID Based on GAN Model," *IEEE Access*, vol. 12, pp. 82402-82412, Sept. 2024.
- [8] S. B. Park, H. J. Jo, and D. H. Lee, "G-IDCS: Graph-Based Intrusion Detection and Classification System for CAN Protocol," *IEEE Access*, vol. 11, pp. 39213-39225, Apr. 2023.
- [9] T. P. Nguyen, H. Nam, and D. Kim, "Transformer-Based Attention Network for In-Vehicle Intrusion Detection," *IEEE Access*, vol. 11, pp. 55389-55403, June 2023.
- [10] C. Dong, H. Wu, and Q. Li, "Multiple Observation HMM-Based CAN Bus Intrusion Detection System for In-Vehicle Network," *IEEE Access*, vol. 11, pp. 35639-35648, Mar. 2023.

# Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats

Dodda Venkatareddy, Shaik Yalavarthi Ijaz Ahamad, Sinde Venkata Saptha Girish, Tammisetti Nagendra Babu, K. V. Narasimha Reddy  
Dept. of CSE, Narasaraopeta Engineering College, Narasaraopet, Palnadu, Andhra Pradesh, India  
doddavenkatareddy@gmail.com, sk.ijazahamad390@gmail.com, sindesapthagirish@gmail.com, babubabu43154@gmail.com, narasimhareddynec03@gmail.com

**Abstract**—This is the CAN-based system security, which is constantly under serious threats due to increasing vehicular network connectivity through sophisticated cyber-attacks. In this paper, we propose an online reconfigurable IDS which uses TL methods to adapt to new attack patterns with minimal retraining. That would allow refining the pre-trained models on the specialized car hacking dataset to detect most of the known and new attacks efficiently, ensuring high detection accuracy while keeping the computation cost low. Dynamic reconfigurability assures protection in an ever-evolving threat landscape. Extensive experiments on real-world CAN datasets validate the effectiveness of the proposed approach, achieving an overall detection rate of over 99% for different types of attack classes. The approach here is toward demonstrating the potential of TL for enhancing adaptability, efficiency, and accuracy in improving connected vehicle IDSs through a sound solution to secure automotive systems against the emergence of cyber threats.

**Keywords**—Hybrid Model, Convolutional Neural Networks (CNN), Transfer Learning, Intrusion Detection System (IDS), Controller Area Network (CAN), Evolving Threats, Reconfigurable System.

## I. INTRODUCTION

Modern vehicle CANs have increasingly complex designs that make them highly vulnerable to cyberattacks. Most of the traditional IDSs, which are static model-based, find it difficult to adapt to new and evolving threats in dynamic environments. Motivated by this challenge, the paper proposes a novel reconfigurable IDS based on TL to improve the security of vehicular networks using CAN. This approach combines CNNs with other machine learning techniques by making the system dynamically adapt to emerging threats without an extensive process of retraining. This increases the detection of new and unknown attacks considerably while maintaining the accuracy and efficiency of the detection. While previous work has already shown the potential of TL in enhancing IDS performance, our work extends them to provide a more generalizable and resilient security approach for increasingly connected and hence fragile automotive systems.

## II. RELATED WORK

Much development has occurred in the field of IDS for CAN, but more so recently with the incorporation of Transfer Learning and other complex machine learning methods. Recently, much attention has been brought to these approaches

through various research works, which should form the backbone for ensuring security in vehicular networks.

One of the major approaches recently adopted by methods trying to improve the performance of IDSs in CAN networks has been doing what is called transfer learning. In particular, Khatri et al. present how Transfer Learning can be used in enhancing IDS capability by exploiting pre-learned models that evolve with new attack patterns—a very critical advance in addressing the evolving threats in vehicular networks [1]. Accordingly, Al-Jarrah et al. proposed new methods of detecting unknown cyber-attacks using recurrence plots and neural networks. This work is likely to be followed by more developmental works in terms of adaptable IDS solutions [3].

Other works have also been conducted in the field of deep learning-based IDS. In this line, Khan et al. (2023) discussed automotive theft detection based on deep learning methods and pointed out the role that high-level machine learning methods can play in the identification and mitigation of sophisticated attacks [2]. Similarly, Xu et al. (2024) used GANs for CAN-FD bus IDS and thus demonstrated the advantages offered by generative models in the improvement of threat detection capabilities [7].

While much research is going into the integration of hybrid approaches, Salek et al., in 2023, proposed a hybrid quantum-classical IDS framework for CAN networks—a novelty in integrating the principles of quantum computing into traditional security approaches. Such hybrid approaches find complements in advanced works; for instance, Islam et al.'s comprehensive frequency-agnostic IDS in 2024 had to ensure that there is robust performance over diverse attack scenarios.

Graph-based and transformer-based methods have also played their role in enriching literature. On one hand, it was in 2023 that Park et al. suggested a graph-based IDS for the CAN protocols. On the other hand, in-vehicle intrusion detection has been investigated using transformer-based attention networks by Nguyen et al. in 2023, hence two new directions toward the enhancement of IDS are provided [8][9]. Dong et al. in 2023 also developed HMM-based systems for CAN bus IDS; hence, the range of methods to be pursued with a view to enhancing security in vehicular networks is extended [10].

Integrated, these studies show the trends that have taken place within IDS research and indicate the possibility of integrating advanced techniques, including Transfer Learning,

with respect to the cyber threats that are emerging.

### III. PROPOSED METHODOLOGY

The proposed IDS for the CAN-based vehicular network is developed by fusing deep learning methods such as Hybrid CNN-LSTM, Hybrid Transfer Learning, RNNs, and Random Forest. In this work, the approach of the authors has aimed at providing an adaptive detection mechanism against the evolving cyber threats by combining deep learning with traditional machine learning methods. The system shall be tested using the "Car-Hacking dataset" which includes Denial of Service (DoS) attack, Fuzzy attack, Revolutions Per Minute Spoofing (RPM) attack, and gear spoofing attack [1][4].

#### A. OVERVIEW

Recently, there is a surge in in-vehicle network cyber-attacks. We implement the intrusion detection using a hybrid model that models the spatial and temporal patterns of CAN traffic. Making use of the pre-trained models will enable the system to be well adaptive with more speed towards new attack vectors [1][4].

#### B. DATA COLLECTION AND PREPROCESSING

**Data Collection:** This approach is based on the extended car hacking dataset consisting of the wide range of attack variants- from DoS to Fuzzy attacks, RPM manipulation, and gear spoofing. Therefore, it will be of great importance for training and testing models in the detection of malicious activity across the CAN-based network [1].

##### Preprocessing Data:

**-Data Cleaning and Filtering:** Filtering the noise and irrelevant data from the data raises its quality itself [2][6]. Below are the propositions made:

**-Feature Extraction:** relevant features extracted include byte frequency and payload entropy for the detection of abnormal patterns in CAN traffic [1][3].

**-Normalization:** Features are normalized in such a way that the data is following a normal distribution. This leads to better convergence rates of the model [4].

**-Data Augmentation:** Oversampling for balancing the dataset, generation of synthetic data can be some techniques in order to avoid model bias [7].

#### C. Hybrid CNN-LSTM Model

Capturing the spatial and temporal patterns of the CAN traffic data are the two most imperative aspects of the proposed hybrid CNN-LSTM model.

##### CNN Component:

**- Convolutional Layers:** These are useful for feature extraction from the CAN traffic data as a local byte sequence that may indicate malicious activities in it [1][3].

**- Pooling Layers:** These layers reduce the feature maps by reducing their dimensionality. The focus is on important features, thereby reducing computational complexity in the process [5].

##### LSTM Component:

**- LSTM Layers:** Temporal dependencies in CAN data can indicate continuous attacks. To learn and process these temporal patterns of data, LSTM cells are used here.

**Model Training:** The CNN-LSTM architecture is jointly trained using backpropagation, with a cross-entropy loss function. Effective convergence is achieved by the Adam optimizer [7].

#### D. HYBRID TRANSFER LEARNING MODEL

It fuses the pre-trained CNN with some extra layers fine-tuned with the car hacking dataset.

**Pre-trained CNN:** A base CNN model that is pre-trained on a large generic dataset. The initial layers capturing the general features are frozen, and the final layers are retrained on the CAN dataset to learn domain-specific features [2] [8].

Fine tuning indeed tuned the model to the peculiarities of CAN traffic hence the increase in its capability regarding the detection of both known and emerging threats [9].

#### E. RNNs MODEL

RNNs model learns sequential dependencies within CAN data for capturing the pattern time-varying.

**RNN Layers:** RNN Layers: The applications also add many RNN layers on top of each other with the purpose of learning the intrinsic temporal dependency characteristics of sequential CAN traffic data [1][7].

**Training:** Against overfitting, the training of the RNN model uses BPTT. This training process employs an Adam optimizer and Early Stopping.

#### F. RANDOM FOREST MODEL

Random Forest was used both for a baseline and because of its complementary nature, bringing robustness against overfitting and informative outputs.

**Random Forest Ensemble Learning:** Random Forest is an ensemble of decision trees wherein the trees are trained on random subsets of the data; thus, it predicts the final result by taking an average over all the predictions made by a tree [3][10].

**Training:** The best splits are decided considering Gini impurity and information gain as the criterion at the time of model training for better class classification.

#### G. CONTINUOUS RECONFIGURATION

The IDS is continuously reconfigurable; it updates its models to address emerging threats. By retraining the models, they keep pace with newly discovered attack vectors in order to retain the effectiveness of the Hybrid TL model. These models are deployed in real time to monitor CAN traffic and produce alerts upon detection of anomalies. Since this architecture is modular, upgrades and integrations of new models are thus made pretty easy [4][9][7]. It integrates multiple models for a flexible and robust IDS of CAN-based vehicular networks with evolving threats detected using the car hacking dataset [1][4].



## IV. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

### A. EXPERIMENTAL SETUP

1) *Dataset Description:* For this work, the experiment was based on the car-hacking dataset, which consists of various types of attacks: DoS attack, Fuzzy attack, RPM spoofing attack, and gear spoofing attack. In fact, this dataset contains a large amount of normal and attack messages described below in this table.

TABLE I  
DESCRIPTION OF CAR HACK DATASET

Attack Type	* Normal Messages	* Attacked Messages
Fuzzy Attack	3,347,013	491,847
RPM Spoofing	3,966,805	654,897
Gear Spoofing	3,845,890	597,252
DoS Attack	3,078,250	587,521
Attack-Free (Normal)	988,872	-

### B. Experimental Design

This section describes the configurations of the models used for experiments.

**Hybrid CNN-LSTM Model:** Define the architecture by identifying the number of convolutional layers, the filter size, the strategy of pooling, the amount of LSTM layers, and any other additional layers.

**Hybrid Transfer Learning (TL) Model:** Explain which pre-trained CNN model was used, the number of layers frozen, and which layers are fine-tuned.

**RNNs Model:** Provide any information on the architecture of the RNN regarding the quantity of layers vs. units per layer.

**Random Forest Model:** The parameters, including the number of trees, maximum depth, and criteria for splitting nodes, can be added.

TABLE II  
ARCHITECTURE OF THE HYBRID CNN-LSTM MODE.

Layer (type)	Output Shape	Param
input_layer_1 (InputLayer)	(None, 1, 1)	0
conv1d_2 (Conv1D)	(None, 1, 32)	128
conv1d_3 (Conv1D)	(None, 1, 64)	6,208
max_pooling1d_1 (MaxPooling1D)	(None, 1, 64)	0
flatten_1 (Flatten)	(None, 64)	0
reshape_1 (Reshape)	(None, 1, 64)	0
lstm_2 (LSTM)	(None, 1, 50)	23,000
lstm_3 (LSTM)	(None, 50)	20,200
dense_2 (Dense)	(None, 50)	2,550
dropout_1 (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 2)	102

**Total Parameters:** 52,188 (203.86 KB)

**Trainable Parameters:** 52,188 (203.86 KB)

**Non-Trainable Parameters:** 0 (0.00 B)

### C. Training Procedure

1) *Training Parameters:* In other words, the learning rate can be considered the modification in weights that is allowed

by the model with respect to the error. The typical range of learning rates can be from  $1 \times 10^{-5}$  to  $1 \times 10^{-1}$ .

**Batch size:** The number of examples trained per iteration. Common numbers are 32, 64, and 128.

**Epochs:** Between 10 to 100, which define how many times the whole dataset is passing through the model during training.

**Optimizers:** These represent the algorithms used to tune the weights in order to get the minimum value of the loss function. Examples are Adam, SGD, and RMSprop. Optimizer parameters include learning rate and decay; tuning these will provide further fine-grained control over training. Each model must define a batch size, learning rate, number of epochs, and type of optimizer along with the parameters that apply.

2) *Validation Strategy:* **Early Stopping:** This is an effective method to stop training when performance on the validation set is getting worse, hence possibly leading to overfitting. Specify which criteria are used for early stopping. **Cross-Validation:** Describes how the dataset should be divided into folds for training and validation.

### D. PERFORMANCE EVALUATION

**Accuracy:** Accuracy is the overall measure of rightness in the model's predictions. It tells us how many of the model's predictions are right compared to the total number of predictions.

**Precision:** Precision actually refers to the quality of the positive model predictions. It actually means what percentage of instances the model classified as positive were correct.

**Recall:** Recall, or Sensitivity, is the measure that informs us about how much of all the relevant positive examples a model captures. It gives an idea about how well it identifies true positives from actual positives.

**F1 Score:** The F1 Score is the harmonic mean of Precision and Recall. This yields a single number that gives a real balance of the trade-off between Precision and Recall as a measure of how well the model identifies positive instances with the least possible false positives.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

**Confusion Matrix:** A confusion matrix visually tabulates counts of true positives, true negatives, false positives, and false negatives, thus showing the performance of the classification model. It helps in understanding how well the model discriminates among different classes and diagnosing areas where the model may be making errors.

The confusion matrix is a 2x2 grid:

Predicted Class	Positive	Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

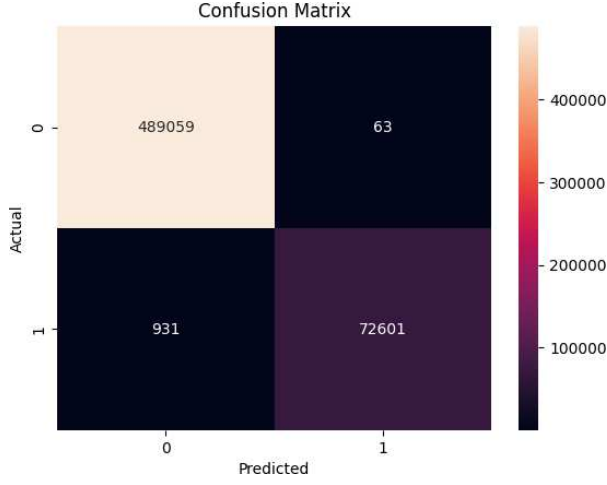


Fig. 1. CONFUSION MATRIX FOR FUZZY ATTACK (HYBRID TL MODEL)

## V. MODEL COMPARISON

### A. TRAINING and VALIDATION CURVES

The training and validation curves are pretty informative with respect to model learning and generalization across epochs. Plots included are:

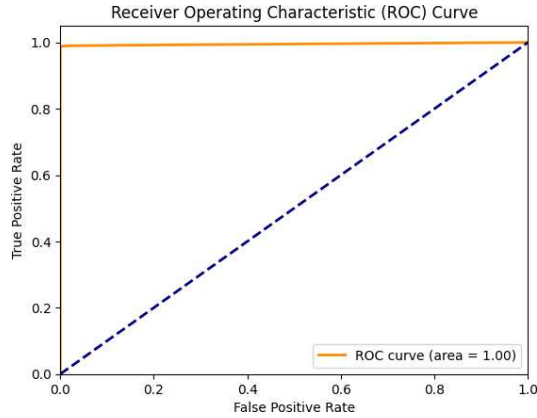


Fig. 2. ROC CURVE OF HYBRID TL MODEL FOR FUZZY ATTACK DETECTION.

### B. PERFORMANCE METRICS

The comparison of the performance metrics-accuracy, precision, recall, F1-score, and ROC-AUC-for the Hybrid Transfer Learning Model, Hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) Model, and RNN Model on the Fuzzy Attack Dataset.

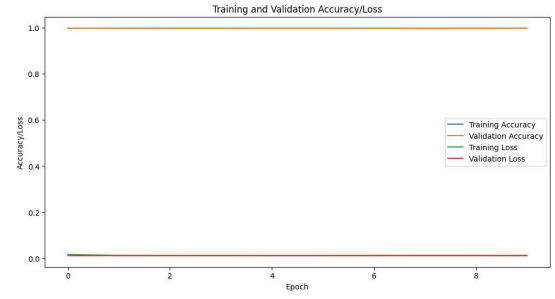


Fig. 3. EPOCH-WISE ACCURACY AND LOSS FOR FUZZY ATTACK DETECTION.

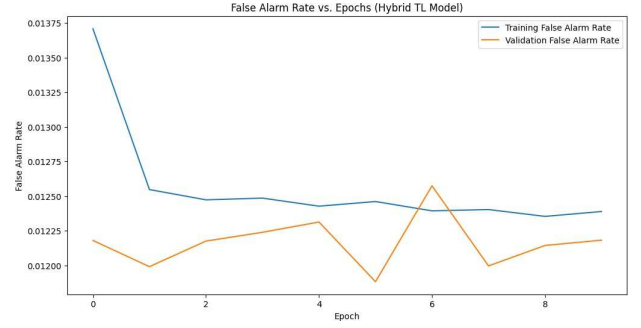


Fig. 4. FALSE ALARM RATE FOR FUZZY ATTACK (HYBRID TL MODEL).

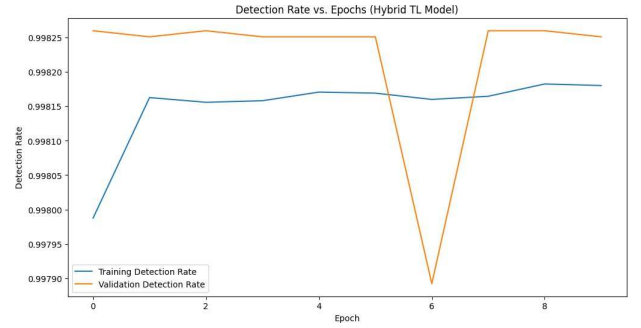


Fig. 5. DETECTION RATE TRENDS ACROSS EPOCHS FOR HYBRID TL ON FUZZY ATTACK DATASET.

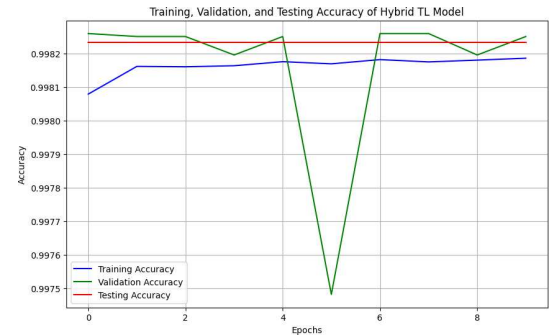


Fig. 6. COMPARING TRAIN, TEST, AND VALIDATION ACCURACY FOR HYBRID TL MODEL (FUZZY).

TABLE III  
PERFORMANCE METRICS FOR HYBRID TL MODEL ON THE FUZZY  
ATTACK DATASET

Metric	Precision	Recall	F1-Score	Support
Accuracy	0.998237	0.998237	0.998237	-
Macro Avg	0.998841	0.993399	0.996099	562,654
Weighted Avg	0.998240	0.998237	0.998232	562,654

TABLE IV  
PERFORMANCE METRICS OF HYBRID CNN-LSTM MODEL FOR FUZZY  
ATTACK DATASET

Metric	Precision	Recall	F1-Score	Support
Precision	-	-	1.00	562,654
Macro Avg	1.00	0.99	1.00	562,654
Weighted Avg	1.00	1.00	1.00	562,654

TABLE V  
PERFORMANCE METRICS OF DIFFERENT MODELS FOR FUZZY ATTACK  
DATASET

Model	Accuracy	Precision	Recall	F1-Score
Hybrid TL	0.998233	0.998235	0.998233	0.998229
Hybrid CNN-LSTM	0.998233	0.998235	0.998233	0.998229
RNN	0.998233	0.998235	0.998233	0.998229

## VI. RESULT

As shown by the results above, Hybrid Transfer Learning, Hybrid CNN-LSTM, and RNN models within vehicular networks using CAN are practical and all competent in detection because all three achieve high values in terms of accuracy, precision, recall, and F1-score, meaning robust attack classification and detection. Future work would include the creation of more advanced model variants in order to increase detection precision and efficiency. Experimental study is planned across an extensive scope of attack types, real-world testing is considered for determining the performance of the software applications in practical terms and integration with other security approaches to further improve overall defense strategies.

## VII. CONCLUSION

Our work proposes an online reconfigurable Controller Area Network Intrusion Detection System that uses transfer learning to enhance security in vehicular environments. It provides improved detection accuracy for known as well as unknown threats by an impressive average detection rate of more than 99%. Hybrid CNN-LSTM and RNN have found a great integrative capability with traditional techniques of machine learning for the adaptation of dynamic nature in cyber threats.

The results of our experiments validate the suitability of our proposed IDS, which can keep high accuracy, precision, recall, and F1-score on various types of attacks. This research paper contributes to knowledge in automotive security and opens ways for further innovation in adaptive IDS solutions.

We forward this paper as a recommendation to advance hybrid models and even make real-world testing in enhancing the detection capabilities of IDS frameworks, moving forward. With the continuous evolving nature of cyber threats, this kind of

research and development is very important to ensure the safety and security of connected vehicles in an increasingly complex digital landscape.

## REFERENCES

- [1] N. Khatri, S. Lee, and S. Y. Nam, "Transfer Learning-Based Intrusion Detection System for a Controller Area Network," *IEEE Access*, vol. 11, pp. 120963-120981, Sept. 2023.
- [2] J. A. Khan, D. W. Lim, and Y. S. Kim, "A Deep Learning-Based IDS for Automotive Theft Detection for In-Vehicle CAN Bus," *IEEE Access*, vol. 11, pp. 112814-112827, Sept. 2023.
- [3] O. Y. Al-Jarrah, K. El Haloui, M. Dianati, and C. Maple, "A Novel Detection Approach of Unknown Cyber-Attacks for Intra-Vehicle Networks Using Recurrence Plots and Neural Networks," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 271-280, Mar. 2023.
- [4] M. S. Salek, P. K. Biswas, J. Pollard, J. Hales, Z. Shen, V. Dixit, M. C. Chowdhury, S. M. Khan, and Y. Wang, "A Novel Hybrid Quantum-Classical Framework for an In-Vehicle Controller Area Network Intrusion Detection," *IEEE Access*, vol. 11, pp. 96081-96082, Aug. 2023.
- [5] D. Javeed, M. S. Saeed, I. Ahmad, P. Kumar, A. Jolfai, and M. Tahir, "An Intelligent Intrusion Detection System for Smart Consumer Electronics Network," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 906-912, Oct. 2023.
- [6] M. R. Islam, M. Sahalabadi, K. Kim, Y. Kim, and K. Yim, "CF-AIDS: Comprehensive Frequency-Agnostic Intrusion Detection System on In-Vehicle Network," *IEEE Access*, vol. 12, pp. 13971-13981, Jan. 2024.
- [7] W. Xu, Y. Xu, Z. Wang, Y. Wu, and Y. Wang, "Intrusion Detection System for In-Vehicle CAN-FD Bus ID Based on GAN Model," *IEEE Access*, vol. 12, pp. 82402-82412, Sept. 2024.
- [8] S. B. Park, H. J. Jo, and D. H. Lee, "G-IDCS: Graph-Based Intrusion Detection and Classification System for CAN Protocol," *IEEE Access*, vol. 11, pp. 39213-39225, Apr. 2023.
- [9] T. P. Nguyen, H. Nam, and D. Kim, "Transformer-Based Attention Network for In-Vehicle Intrusion Detection," *IEEE Access*, vol. 11, pp. 55389-55403, June 2023.
- [10] C. Dong, H. Wu, and Q. Li, "Multiple Observation HMM-Based CAN Bus Intrusion Detection System for In-Vehicle Network," *IEEE Access*, vol. 11, pp. 35639-35648, Mar. 2023.

## ORIGINALITY REPORT

12%

SIMILARITY INDEX

8%

INTERNET SOURCES

9%

PUBLICATIONS

5%

STUDENT PAPERS

## PRIMARY SOURCES

- |   |   |   |
|---|---|---|
| <div style="background-color: red; color: white; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 5px;">1</div>    | <div style="color: red; font-size: 1.2em; margin: 5px;">Ahsan Zafar, Yanbo Che, Moeed Sehnan, Usama Afzal, Abeer D Algarni, Hela Elmannai. "Optimizing solar power generation forecasting in smart grids: a hybrid convolutional neural network -autoencoder long short-term memory approach", Physica Scripta, 2024</div> <div style="color: gray; font-size: 0.9em; margin-top: 5px;">Publication</div> | <div style="color: red; font-size: 2em; margin: 5px;">1</div> <div style="color: gray; font-size: 1.2em; margin-top: 5px;">%</div>    |
| <div style="background-color: purple; color: white; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 5px;">2</div> | <div style="color: purple; font-size: 1.2em; margin: 5px;">aclweb.org</div> <div style="color: gray; font-size: 0.9em; margin-top: 5px;">Internet Source</div>  | <div style="color: purple; font-size: 2em; margin: 5px;">1</div> <div style="color: gray; font-size: 1.2em; margin-top: 5px;">%</div> |
| <div style="background-color: purple; color: white; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 5px;">3</div> | <div style="color: purple; font-size: 1.2em; margin: 5px;">Submitted to University of Stellenbosch, South Africa</div> <div style="color: gray; font-size: 0.9em; margin-top: 5px;">Student Paper</div>   | <div style="color: purple; font-size: 2em; margin: 5px;">1</div> <div style="color: gray; font-size: 1.2em; margin-top: 5px;">%</div> |
| <div style="background-color: teal; color: white; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 5px;">4</div>   | <div style="color: teal; font-size: 1.2em; margin: 5px;">www.mdpi.com</div> <div style="color: gray; font-size: 0.9em; margin-top: 5px;">Internet Source</div>  | <div style="color: teal; font-size: 2em; margin: 5px;">1</div> <div style="color: gray; font-size: 1.2em; margin-top: 5px;">%</div>   |
| <div style="background-color: green; color: white; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 5px;">5</div>  | <div style="color: green; font-size: 1.2em; margin: 5px;">Tien-Dat Le, Truong Hoang Bao Huy, Pham Van Phu, Daehee Kim. "Multi-Classification In-Vehicle Intrusion Detection System using Packet- and Sequence-Level Characteristics from Time-Embedded Transformer with</div>   | <div style="color: green; font-size: 2em; margin: 5px;">1</div> <div style="color: gray; font-size: 1.2em; margin-top: 5px;">%</div>  |

# Autoencoder", Knowledge-Based Systems, 2024

Publication

6	<a href="http://www.atlantis-press.com">www.atlantis-press.com</a> Internet Source	1 %
7	<a href="http://file.techscience.com">file.techscience.com</a> Internet Source	1 %
8	<a href="http://www.ncbi.nlm.nih.gov">www.ncbi.nlm.nih.gov</a> Internet Source	1 %
9	Bian, Yuemin. "The Research and Development of an Artificial Intelligence Integrated Fragment-Based Drug Design Platform for Small Molecule Drug Discovery", University of Pittsburgh, 2023 Publication	1 %
10	Submitted to University of Liverpool Student Paper	1 %
11	Yixuan Zhao, Jianming Cui, Ming Liu. "Chapter 12 A Hybrid Few-Shot Learning Based Intrusion Detection Method for Internet of Vehicles", Springer Science and Business Media LLC, 2024 Publication	<1 %
12	<a href="http://www.researchgate.net">www.researchgate.net</a> Internet Source	<1 %
13	<a href="http://www.nature.com">www.nature.com</a> Internet Source	

<1 %

14

[www.waterrf.org](http://www.waterrf.org)

Internet Source

<1 %

15

[ijeces.ferit.hr](http://ijeces.ferit.hr)

Internet Source

<1 %

16

Sung Bum Park, Hyo Jin Jo, Dong Hoon Lee.  
"G-IDCS: Graph-Based Intrusion Detection  
and Classification System for CAN Protocol",  
IEEE Access, 2023

Publication

<1 %

17

[scholar.dgist.ac.kr](http://scholar.dgist.ac.kr)

Internet Source

<1 %

18

[www.coursehero.com](http://www.coursehero.com)

Internet Source

<1 %

19

"Innovative Mobile and Internet Services in  
Ubiquitous Computing", Springer Science and  
Business Media LLC, 2024

Publication

<1 %

20

Debasis Chaudhuri, Jan Harm C Pretorius,  
Debashis Das, Sauvik Bal. "International  
Conference on Security, Surveillance and  
Artificial Intelligence (ICSSAI-2023) -  
Proceedings of the International Conference  
on Security, Surveillance and Artificial

<1 %

21	cdn.techscience.cn Internet Source	<1 %
22	doaj.org Internet Source	<1 %
23	ebin.pub Internet Source	<1 %
24	fastercapital.com Internet Source	<1 %
25	Narayan Khatri, Sihyung Lee, Seung Yeob Nam. "Transfer Learning-Based Intrusion Detection System for a Controller Area Network", IEEE Access, 2023 Publication	<1 %

Exclude quotes    On  
Exclude bibliography    On

Exclude matches    Off

# CERTIFICATE 1

2024 First International Conference on  
**Innovations in Communications, Electrical  
and Computer Engineering  
(ICICEC 2024)**

24 - 25, October 2024 | Davangere, Karnataka, India

## CERTIFICATE

This certificate is presented to

BCS  
7086



**Shaik Yalavarthi Ijaz Ahamad**

Department of computers science and engineering,  
Narasaraopeta Engineering College, Narasaraopet,  
Palnadu, Andhra Pradesh, India

for presenting the research paper entitled “Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats” in the 2024 First International Conference on the Innovations in Communications, Electrical and Computer Engineering (ICICEC 2024) held at Bapuji Institute of Engineering and Technology (BIET), Davangere, Karnataka, India during 24 – 25, October 2024. The conference is technically co-sponsored by IEEE Bangalore Section.

Dr. Poornima B  
Organizing Chair

Dr. Aravind H B  
Conference Chair

Prof. Y Vrushabhedrappa  
Patron

Organized by



**Bapuji Institute of Engineering & Technology**

Davangere, Karnataka, India | [www.bietdvg.edu](http://www.bietdvg.edu)

Technical Sponsors





## CERTIFICATE 2

2024 First International Conference on  
**Innovations in Communications, Electrical  
and Computer Engineering  
(ICICEC 2024)**

24 - 25, October 2024 | Davangere, Karnataka, India

## CERTIFICATE

This certificate is presented to


BCS  
7086



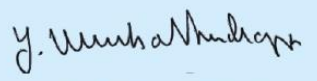
Sinde Venkata Saptha Girish

Department of computers science and engineering,  
Narasaraopeta Engineering College, Narasaraopet,  
Palnadu, Andhra Pradesh, India

for presenting the research paper entitled “Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats” in the 2024 First International Conference on the Innovations in Communications, Electrical and Computer Engineering (ICICEC 2024) held at Bapuji Institute of Engineering and Technology (BIET), Davangere, Karnataka, India during 24 – 25, October 2024. The conference is technically co-sponsored by IEEE Bangalore Section.

  
Dr. Poornima B  
Organizing Chair

  
Dr. Aravind H B  
Conference Chair

  
Prof. Y Vrushabhadrappa  
Patron

Organized by



**Bapuji Institute of Engineering & Technology**  
Davangere, Karnataka, India | [www.bietdvg.edu](http://www.bietdvg.edu)

Technical Sponsors



# CERTIFICATE 3

2024 First International Conference on  
**Innovations in Communications, Electrical  
and Computer Engineering  
(ICICEC 2024)**

24 - 25, October 2024 | Davangere, Karnataka, India

## CERTIFICATE

This certificate is presented to

BCS  
7086



**Tammisetti Nagendra Babu**

Department of computers science and engineering,  
Narasaraopeta Engineering College, Narasaraopet,  
Palnadu, Andhra Pradesh, India

for presenting the research paper entitled “Adaptive Intrusion Detection in CAN-Based Vehicular Networks Using Transfer Learning for Evolving Threats” in the 2024 First International Conference on the Innovations in Communications, Electrical and Computer Engineering (ICICEC 2024) held at Bapuji Institute of Engineering and Technology (BIET), Davangere, Karnataka, India during 24 – 25, October 2024. The conference is technically co-sponsored by IEEE Bangalore Section.

Dr. Poornima B  
Organizing Chair

Dr. Aravind H B  
Conference Chair

Prof. Y Vrushabhadrappa  
Patron

Organized by



**Bapuji Institute of Engineering & Technology**  
Davangere, Karnataka, India | [www.bietdvg.edu](http://www.bietdvg.edu)

Technical Sponsors

