A HYBRID LIGHTGBM–1D CNN FRAMEWORK FOR ANOMALY DETECTION IN VEHICULAR CAN TRAFFIC

A THESIS
SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted by

ASHWANI KUMAR
(2K24/ISY/17)

Under the supervision of
Dr. RITU AGARWAL
Department of Information Technology

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2026

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the thesis titled "A Hybrid LightGBM–1D CNN Framework for Anomaly Detection in Vehicular CAN Traffic" which is submitted by Ashwani Kumar, Roll No. 2K24/ISY/17, Department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirements for the award of the degree of Master of Technology in Information Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this University or elsewhere.

Place: Delhi **Dr. Ritu Agarwal**

Date: **SUPERVISOR**

Associate Professor

Department of Information Technology

Delhi Technological University

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Ashwani Kumar**, Roll No. **2K24/ISY/17**, student of M.Tech (Information Technology), hereby declare that the thesis titled "*A Hybrid LightGBM–1D CNN Framework for Anomaly Detection in Vehicular CAN Traffic*" which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirements for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

Place: Delhi **Ashwani Kumar**

Date: (2K24/ISY/17)

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project supervisor, **Dr. Ritu Agarwal**, Department of Information Technology, Delhi Technological University, for her invaluable guidance, encouragement, and continuous support throughout the course of this thesis work. Her insightful feedback, technical expertise, and patient mentorship have shaped not only this dissertation but also my broader understanding of how to conduct disciplined empirical research.

I humbly extend my words of gratitude to **Prof. Dinesh Kumar Vishwakarma**, Head of the Department of Information Technology, and to all the other faculty members of the department, for providing a stimulating research environment, the necessary infrastructure, and their valuable time whenever it was required.

I am grateful to the open-source community whose tools made this research possible. The contributors behind the Car-Hacking dataset hosted at HCRL, and the developers of Python, TensorFlow, Keras, LightGBM, scikit-learn, NumPy and Pandas, have collectively reduced what would have been months of infrastructure work to a few well-chosen commands.

I thank all my friends and fellow students at DTU who supported me throughout the execution of this work, both technically and personally. Finally, I express my deepest gratitude to my family for their unwavering belief, patience, and encouragement throughout my postgraduate studies.

Place: Delhi **Ashwani Kumar**

Date: (2K24/ISY/17)

Abstract

The Controller Area Network (CAN) protocol serves as the principal in-vehicle communication backbone in modern automobiles, enabling real-time interaction between dozens of Electronic Control Units (ECUs) that govern safety-critical functions such as braking, steering, engine control and infotainment. CAN was designed in the 1980s for reliability rather than security; it carries no authentication, no encryption and no message-origin verification. As the vehicle has evolved into a connected, networked cyber-physical system, this design choice has been transformed from an engineering simplification into a serious attack surface, exposing road users to Denial-of-Service (DoS), Fuzzy, Impersonation and injection attacks that can disable steering, falsify telemetry or freeze the bus.

This thesis presents a hybrid machine-learning Intrusion Detection System (IDS) for in-vehicle CAN networks that combines a fast, interpretable ensemble model with a deep sequence learner. Two complementary models are proposed and evaluated: a Light Gradient Boosting Machine (LightGBM) classifier configured for embedded-grade inference, and a One-Dimensional Convolutional Neural Network (1D-CNN) that learns spatial and temporal dependencies directly from sequences of CAN payload bytes. Both models are trained on a stratified, four-class subset of the HCRL Car-Hacking dataset comprising 400,000 messages drawn from approximately seven million real-vehicle CAN frames, balanced across Normal, DoS, Fuzzy and Impersonation traffic. A four-feature engineered enrichment (DataSum, DataMean, DataStd and TimeDiff) is added to the eleven raw fields to expose statistical and temporal irregularities of attack payloads.

A comparative evaluation against classical baselines from the author's earlier studies (Support Vector Machine, Decision Tree and K-Nearest Neighbour) shows that LightGBM achieves an overall accuracy of **89.34%** with sub-second training time and

a memory footprint below 60 KB, making it attractive for deployment on resource-constrained ECUs. The 1D-CNN attains **83.77%** accuracy while demonstrating stronger generalisation to subtle temporal anomaly patterns, particularly for the Fuzzy attack class. A per-class confusion-matrix analysis, combined with Cohen's κ and ROC-AUC measurements, confirms that the two models capture different aspects of the attack space and are therefore mutually complementary rather than redundant. The thesis concludes with a discussion of latency, embedded-deployment trade-offs, identified research gaps, and the social implications of robust vehicular intrusion detection.

Keywords: Intrusion Detection System, Controller Area Network, Vehicular Cybersecurity, In-Vehicle Communication, LightGBM, 1D-CNN, Machine Learning, Deep Learning, Anomaly Detection.

Contents

Certificate **i**

Candidate's Declaration **ii**

Acknowledgement **iv**

Abstract **v**

Contents **vii**

List of Tables **viii**

List of Figures **ix**

List of Symbols, Abbreviations **x**

1 INTRODUCTION **xii**
1.1 Background and Motivation xii
1.2 Problem Statement xiii
1.3 Research Objectives xiv
1.4 Scope and Limitations xiv
1.5 Contributions of the Thesis 1
1.6 Thesis Organisation 1

2 LITERATURE REVIEW **2**
2.1 Overview of CAN Bus Security 2
2.2 Rule-Based and Statistical Approaches 2
2.3 Classical Machine-Learning Approaches 3
2.4 Ensemble and Hybrid Methods 3
2.5 Deep Learning Approaches 4
2.6 Publicly Available Datasets 4
2.7 Identified Research Gaps 5

3 METHODOLOGY **6**
3.1 System Architecture Overview 6
3.2 Classical Baselines 7
3.3 LightGBM-Based IDS 8
3.4 1D Convolutional Neural Network 9
3.5 Evaluation Metrics 11
3.6 Implementation Environment 11
3.7 Experimental Comparison Design 11

4 METHODOLOGY **12**
4.1 System Architecture Overview 12
4.2 Classical Baselines 13
4.3 LightGBM-Based IDS 14
4.4 1D Convolutional Neural Network 15
4.5 Evaluation Metrics 17
4.6 Implementation Environment 17
4.7 Experimental Comparison Design 17

5 RESULTS AND DISCUSSION **18**
5.1 Classical Baseline Results 18
5.2 LightGBM Results 19
5.3 1D-CNN Results 20
5.4 1D-CNN Training Dynamics 21
5.5 Comparative Analysis Across All Five Models 21
5.6 Training Cost, Model Footprint and Inference Considerations 22
5.7 Error Analysis 23
5.8 Discussion 23

6 CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT **24**
6.1 Conclusion 24
6.2 Future Scope 25
6.3 Social Impact 26

References **30**

A Preprocessing and Feature Engineering Code **31**

B Model Training and Evaluation Code **33**

List of Tables

3.1 LightGBM hyperparameter configuration used in this thesis. 8
3.2 Layer-by-layer specification of the proposed 1D-CNN architecture. . . . 9
3.3 Implementation environment. 11

4.1 LightGBM hyperparameter configuration used in this thesis. 14
4.2 Layer-by-layer specification of the proposed 1D-CNN architecture. . . . 15
4.3 Implementation environment. 17

5.1 Classical baseline results on the 100,000-message held-out test partition. 18
5.2 LightGBM per-class results on the 100,000-message held-out test partition. Aggregate accuracy: **89.34%**. Cohen's κ: **0.858**. Macro ROC-AUC: **0.962**. 19
5.3 1D-CNN per-class results on the 100,000-message held-out test partition. Aggregate accuracy: **83.77%**. Cohen's κ: **0.784**. Macro ROC-AUC: **0.941**. 20
5.4 Comparative results across all five models on the held-out 100,000-message test partition. 22
5.5 Training cost, peak inference memory and serialised model size across all five models. 22

List of Figures

3.1 End-to-end architecture of the proposed CAN IDS. The same preprocessing pipeline feeds all five classifiers; the per-class predictions are scored against ground truth in Chapter 5. 7

3.2 Layer-wise diagram of the 1D-CNN model. Each Conv1D block is followed by batch normalisation, ReLU, max-pooling and dropout. Global average pooling collapses the 3×64 final feature map to a 64-dimensional vector for the dense classification head. 10

4.1 End-to-end architecture of the proposed CAN IDS. The same preprocessing pipeline feeds all five classifiers; the per-class predictions are scored against ground truth in Chapter 5. 13

4.2 Layer-wise diagram of the 1D-CNN model. Each Conv1D block is followed by batch normalisation, ReLU, max-pooling and dropout. Global average pooling collapses the 3×64 final feature map to a 64-dimensional vector for the dense classification head. 16

5.1 Four-way confusion matrix for the LightGBM model on the held-out 100,000-message test partition. Each cell reports the test-set count. DoS is recovered almost perfectly; Fuzzy and Impersonation are the two classes the model most often confuses with each other and with Normal. 19

5.2 Four-way confusion matrix for the 1D-CNN model on the held-out 100,000-message test partition. DoS recovery is comparable to LightGBM; Impersonation suffers from a deeper confusion with Normal. 20

5.3 1D-CNN training and validation loss across the 25 epochs at which early stopping halted training. Validation loss tracks training loss closely (final gap < 0.04), confirming that the dropout-regularised network did not overfit on the 300,000-sample training partition. 21

5.4 Side-by-side comparison of all five models on Accuracy, Macro-F1, Cohen's κ and Macro ROC-AUC. The classical baselines lead on aggregate accuracy, but the deployability and per-class story is more nuanced. . . 22

List of Symbols and Abbreviations

Symbol/Abbr	Expansion
AUC	Area Under the (ROC) Curve
CAN	Controller Area Network
CAN-FD	CAN with Flexible Data-rate
CAN ID	Controller Area Network Identifier
CNN	Convolutional Neural Network
DLC	Data Length Code
DoS	Denial-of-Service (attack)
DT	Decision Tree
ECU	Electronic Control Unit
EDA	Exploratory Data Analysis
F1	F1-score (harmonic mean of Precision and Recall)
FN	False Negative
FP	False Positive
GBDT	Gradient-Boosted Decision Tree
HCRL	Hacking and Countermeasure Research Lab
IDS	Intrusion Detection System
IoT	Internet of Things
IVN	In-Vehicle Network
KNN	K-Nearest Neighbours
LightGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory (network)
ML	Machine Learning
NN	Neural Network
OBD-II	On-Board Diagnostics, version II
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit (activation)
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
κ	Cohen's Kappa coefficient
η	Learning rate
N	Number of samples in the dataset
C	Number of target classes ($C = 4$ in this thesis)
x_i	Feature vector of the i-th CAN message

y_i	Class label of the i-th CAN message ($y_i \in \{0, 1, 2, 3\}$)
$\hat{y}_i$	Model-predicted class for the i-th CAN message
Δt	Inter-arrival time between successive CAN frames (`TimeDiff`)

Chapter 1

INTRODUCTION

1.1 Background and Motivation

A modern passenger vehicle is no longer a mechanical artifact with electronics bolted on; it is a distributed cyber-physical system in which between fifty and one hundred Electronic Control Units (ECUs) coordinate through a small number of in-vehicle buses to deliver braking, steering, powertrain, body comfort and infotainment functions. The communication backbone that ties most of those ECUs together is the Controller Area Network (CAN), a broadcast bus protocol standardised by Bosch in the 1980s and codified as ISO 11898 [1, 2]. CAN was engineered for the operating conditions of an automotive harness — short distances, high electromagnetic noise, strict real-time constraints and tight cost budgets — and it has succeeded so completely in that environment that virtually every vehicle produced in the last three decades carries at least one CAN bus, often several.

That engineering success carries a structural problem. CAN was designed before automotive networks were exposed to wireless interfaces, telematics units, infotainment app stores or remote diagnostic services. It provides no message-source authentication, no payload encryption, no integrity protection beyond a 15-bit CRC and no rate-limiting against a misbehaving node. Any ECU on the bus can transmit any identifier, and the highest-priority identifier always wins arbitration. In a sealed harness this is harmless; once the bus is reachable from outside the car — through OBD-II dongles, infotainment Bluetooth/Wi-Fi stacks, cellular telematics units, charging interfaces, or compromised aftermarket accessories — the absence of authentication becomes a direct attack surface [3, 4].

The set of attacks that exploit this surface is now well documented. *Denial-of-Service (DoS)* attacks flood the bus with high-priority frames so that legitimate ECUs lose arbitration and stop being heard. *Fuzzy* attacks transmit randomised CAN IDs and payloads to provoke unintended actuator behaviour and to map ECU response patterns. *Impersonation* (or spoofing) attacks transmit frames using an existing ECU's identifier, causing receivers to act on falsified data — a spoofed wheel-speed message can defeat the anti-lock braking system, and a spoofed gear-position message can trigger an unintended downshift. A 2015 demonstration by Miller and Valasek showed that a remote attacker could take full longitudinal and lateral control of an unmodified production vehicle over the cellular network, and similar results have since been reproduced on multiple makes [4, 5].

Two architectural responses to this exposure are possible. The first is to redesign the bus protocol itself — adding message authentication codes, migrating to CAN-FD

with cryptographic extensions, or replacing CAN with Automotive Ethernet at sensitive segments. These are durable solutions, but they take a decade to roll through a vehicle fleet because the installed base cannot be retrofitted economically. The second, complementary, response is to deploy an Intrusion Detection System (IDS) that passively observes bus traffic and raises an alert when the observed pattern departs from expected behaviour. An IDS is attractive precisely because it is non-invasive: it does not modify the protocol, does not require ECU firmware changes, and can be added to existing vehicles via gateway-level deployment. The technical question this thesis addresses is how to build such an IDS so that it is accurate enough to be useful, light enough to run on an ECU and robust enough to generalise beyond the specific attack signatures it was trained on.

Early IDS work for CAN relied on rule-based heuristics: monitor inter-frame timing, count messages per identifier per second, compute entropy of recently observed ID sequences. These heuristics work in a laboratory but generate unacceptable false-positive rates once the vehicle moves through realistic driving cycles in which legitimate bus load varies by an order of magnitude [6]. The natural next step was supervised machine learning. Classical classifiers — Support Vector Machines (SVM), Decision Trees (DT), k-Nearest Neighbours (KNN), Random Forests — have been shown to separate normal from attack traffic at 95–99% accuracy on labelled datasets [7, 8]. But classical models depend on handcrafted features and degrade quickly when the attack distribution shifts, and they offer limited expressive capacity for the byte-level temporal patterns that distinguish subtle attacks.

This thesis is built on the conviction that the strongest near-term IDS for CAN combines two complementary modelling families: a fast, interpretable Gradient-Boosted Decision Tree (specifically LightGBM [9]) that can run inside an ECU's compute budget, and a lightweight 1D Convolutional Neural Network that learns local temporal-spatial structure directly from CAN payload sequences. Their combination, when evaluated against the same classical baselines, exposes where each family wins and loses, and provides the empirical basis for choosing one (or both) in a deployed system.

1.2 Problem Statement

The task addressed in this thesis is a four-class supervised classification problem: given a sliding window of CAN bus traffic represented as a feature vector $x_i \in \mathbb{R}^{15}$, learn a function $f : \mathbb{R}^{15} \to \{0, 1, 2, 3\}$ that assigns the labels Normal, DoS, Fuzzy and Impersonation with maximum classification accuracy while respecting the inference-latency and memory budgets imposed by automotive ECU hardware.

Several characteristics make the problem deceptively hard. First, class distributions in real vehicular traffic are heavily skewed toward Normal frames, so naive classifiers collapse onto the majority class. Second, the discriminative signal for some attacks — particularly Fuzzy — lies in subtle byte-level statistical irregularities that are not visible at the message-rate level. Third, the model has to operate under tight resource constraints: an Infineon AURIX or NXP MPC57xx ECU typically has tens to hundreds of kilobytes of RAM available for application code, not hundreds of megabytes. Fourth, the public datasets on which IDS research is calibrated, while extensive, are captured on a small number of vehicles and may not capture the full distribution of attack tactics deployed against future fleets.

1.3 Research Objectives

The work reported in this dissertation pursues the following specific objectives.

- To characterise the statistical and temporal distance between normal and attack CAN traffic on a four-class, balanced subset of the HCRL Car-Hacking dataset, using engineered statistical features (DataSum, DataMean, DataStd, TimeDiff) in addition to the eleven raw CAN fields.
- To re-establish a competitive classical-ML baseline (SVM, Decision Tree, KNN) under a single consistent preprocessing pipeline, so that the proposed models can be compared on a like-for-like basis rather than against numbers reported in heterogeneous prior work.
- To implement and tune a LightGBM-based IDS and report its accuracy, per-class precision/recall, training cost and inference footprint, with a view to embedded deployability.
- To implement and tune a 1D Convolutional Neural Network for the same four-class problem, examining whether its deeper representational capacity buys generalisation that the gradient-boosted model cannot match.
- To analyse where each model wins (and loses) on the four-way confusion matrix, and to discuss whether the two models are complementary enough to motivate an ensemble or staged-inference deployment.
- To articulate the research gaps that remain after this work, including online adaptation, federated training across vehicle fleets, adversarial robustness, and ECU-level deployment, and to discuss the social impact of robust vehicular intrusion detection.

1.4 Scope and Limitations

The scope of this work is bounded in three ways that should be made explicit. First, the evaluation is conducted entirely on the publicly available HCRL Car-Hacking corpus; the thesis does not collect new logs from a physical vehicle bench. The four attack classes used (Normal, DoS, Fuzzy, Impersonation) reflect the dominant categories in that corpus, and the conclusions drawn here may not extend without adjustment to attacks that depend on protocol-level violations the dataset does not exercise (for example, replay attacks against authenticated CAN-FD segments, or attacks targeting Automotive Ethernet sub-networks).

Second, the deployment-side claims (sub-second LightGBM training, sub-60-KB model footprint, ECU-grade inference latency) are derived from training-host measurements and theoretical model-size estimates rather than from a port to a real automotive microcontroller. A real ECU port would surface integer-quantisation, scheduler-jitter and watchdog-budget concerns that are out of scope here. Third, the work does not attempt adversarial robustness evaluation: the train and test distributions are drawn from the same corpus, and an attacker who knows the IDS architecture might craft evasive payloads that defeat both proposed models.

These limitations are addressed in the Future Scope discussion in Chapter 6; flagging them up front sets the right expectations for the results reported in Chapter 5.

1.5 Contributions of the Thesis

The principal contributions of this thesis are as follows.

- A unified, reproducible preprocessing pipeline for the HCRL Car-Hacking corpus that yields a four-class, 400,000-message stratified working dataset with eleven raw fields and four engineered statistical features.
- A LightGBM-based IDS configured for low-latency embedded inference (500 estimators, $\eta = 0.05$, leaf-wise growth, max depth 10), reaching 89.34% accuracy on the held-out test partition with a training cost of roughly 5 seconds and a serialised model footprint below 60 KB.
- A 1D Convolutional Neural Network architecture (three Conv1D layers of 64/128/64 filters, batch-normalisation, dropout, global average pooling, dense 128 / dense 64, softmax) that reaches 83.77% accuracy and exhibits stronger generalisation on the Fuzzy attack class than the gradient-boosted model.
- A direct, like-for-like comparison of both proposed models against three classical baselines (SVM, Decision Tree, KNN) trained from the same preprocessing pipeline, with per-class precision/recall, four-class confusion matrices, Cohen's κ, and ROC-AUC.
- A discussion of the latency, memory and interpretability trade-offs that govern which of the proposed models is more deployable on real ECU hardware, and a list of research gaps that this work explicitly does not close.

1.6 Thesis Organisation

The remainder of this dissertation is organised as follows. Chapter 2 reviews the literature on CAN bus security, intrusion detection on automotive networks, classical and deep-learning IDS approaches and the relevant publicly available datasets, and identifies the research gaps this thesis addresses. Chapter **??** describes the HCRL Car-Hacking corpus, the stratified sampling and preprocessing pipeline used in this thesis, and reports exploratory analysis of the four-class distribution. Chapter 4 presents the overall system architecture, the engineered feature set, the configuration of the classical baselines, the LightGBM model and the 1D-CNN model, the evaluation metrics adopted, and the experimental environment. Chapter 5 reports and discusses the quantitative results, including comparative tables, four-class confusion matrices, training-loss curves and per-class analyses of where each model wins and loses. Chapter 6 summarises the findings, articulates the future scope in concrete terms, and discusses the social impact of robust vehicular intrusion detection. Appendices A and B contain the principal Python code listings used to produce the experiments in Chapters **??**–5.

Chapter 2

LITERATURE REVIEW

The literature on vehicular intrusion detection has matured rapidly over the past decade. Early work proved that CAN was attackable in principle; subsequent work has progressively closed the gap between proof-of-concept attacks and deployable defences. This chapter reviews that trajectory in six parts.

2.1 Overview of CAN Bus Security

Koscher et al. [3] provided the first systematic experimental analysis of automotive attack surfaces, demonstrating that an adversary with physical access to the OBD-II port of a typical passenger vehicle could compromise nearly every safety-relevant subsystem on the bus. Checkoway et al. [10] extended that analysis to remote attack vectors — cellular telematics, Bluetooth and the media player CD subsystem — and showed that purely remote compromise of the in-vehicle network was already feasible in 2011. Miller and Valasek's 2015 demonstration on a 2014 Jeep Cherokee [4] crystallised the result for the wider engineering community: a remote attacker over the cellular network could disable braking, alter steering and stop the engine at highway speeds.

The proximate cause of all of these attacks is the same: the CAN frame format, codified in ISO 11898 and Bosch's CAN specification [1, 2], provides arbitration on the 11-bit (or 29-bit, in extended form) identifier and a 0–8-byte payload protected by a 15-bit CRC, with no field reserved for source authentication, payload encryption or replay protection. Several follow-up protocol variants address parts of this gap. CAN-FD increases payload length to 64 bytes and increases data-phase bit rates, but does not in itself add authentication [11]. Authenticated extensions such as CANAuth and TESLA-CAN have been proposed in the literature [12], and the AUTOSAR Secure Onboard Communication (SecOC) module standardises a truncated MAC appended to selected frames [13]. Adoption of these extensions in production fleets remains partial, and the installed base of unauthenticated CAN buses will dominate the field for at least another decade. This is the gap that vehicular IDS work targets.

2.2 Rule-Based and Statistical Approaches

Pre-learning IDS work for CAN relied on monitoring three observable signals: per-identifier message frequency, inter-arrival time entropy and the entropy of recently observed identifier sequences. Taylor et al. [6] proposed an LSTM-based anomaly

detector but also reported a strong frequency-only baseline that detected most flood-style attacks reliably. Müter and Asaj [14] introduced an entropy-based detector that flags drops in identifier-sequence entropy as DoS-attack symptoms. Marchetti and Stabili [15] formalised an inter-arrival-time analysis with statistical thresholds.

These statistical methods share two desirable properties — they are fast and they require no labelled attack data — and two limiting properties. First, they tend to produce high false-positive rates once the vehicle moves through diverse driving cycles, because the legitimate bus load is itself non-stationary. Second, they handle frequency-distortion attacks (DoS, flooding) far better than payload-distortion attacks (Fuzzy, content-spoofing impersonation) because the signal they observe is at the message-rate level rather than the payload-byte level. The supervised machine-learning IDS work that followed was largely a response to this second limitation.

2.3 Classical Machine-Learning Approaches

Once labelled CAN attack corpora became available — principally the HCRL Car-Hacking dataset [8, 16] — supervised classifiers became the dominant IDS approach. Kang and Kang [17] reported one of the first deep-learning IDS results on CAN, using a deep belief network with 98% reported accuracy. Moulahi et al. [7] provided one of the most cited comparative studies, evaluating Support Vector Machines, Decision Trees, Random Forests and Multi-Layer Perceptrons on a labelled CAN corpus and reporting accuracies above 95% across the board, with SVMs and Random Forests winning by small margins on different attack classes.

Decision Trees are attractive in this domain because they yield models small enough to fit in an ECU's flash memory and inference paths that are interpretable in a forensic sense. KNN performs surprisingly well on these datasets because the four-class structure leaves identifiable clusters in feature space, but it imposes prohibitive memory cost at deployment because the entire training set must be retained for distance queries. SVMs reach competitive accuracy with sub-megabyte model sizes if a linear kernel is acceptable, but they degrade if the feature space is high-dimensional or noisy. Across this family, three weaknesses are consistently reported in the literature: dependence on careful feature engineering, brittle generalisation to attack types not present in the training set, and limited ability to capture byte-level temporal regularities.

The author's earlier work [18] established a within-thesis classical baseline using SVM, Decision Tree and KNN on a smaller four-class working subset, reaching the 95–99% accuracy band reported in this literature. Those numbers serve as one of the comparison points in Chapter 5.

2.4 Ensemble and Hybrid Methods

Ensemble learning addresses the brittleness of single classical models by combining many weak learners into a strong predictor. Random Forests have been the most studied ensemble in CAN IDS [7, 8], but gradient-boosted decision tree (GBDT) implementations have recently overtaken them on speed and memory. XGBoost [19] and LightGBM [9] both implement histogram-based gradient boosting, but LightGBM additionally adopts leaf-wise tree growth and Gradient-based One-Side Sampling (GOSS),

which substantially reduces training cost on large tabular datasets. CatBoost [20] is an alternative GBDT implementation with native handling of categorical features.

The CAN IDS literature has converged on LightGBM in particular because it pairs sub-second training on 10^5–10^6-message datasets with model footprints in the tens of kilobytes, both of which matter for embedded deployment. Recent work [21, 22] has reported LightGBM-based CAN IDS designs in the 88–96% accuracy range with sub-millisecond per-frame inference, and has begun pairing them with SHAP-based explainability tooling to surface the per-feature contributions to a flagged frame.

Hybrid IDS designs that combine a fast tabular model with a deeper sequence model have also appeared in the literature [23, 24]. The motivation is that the tabular model handles the high-volume Normal-vs-DoS decision cheaply while the sequence model is reserved for ambiguous payload-level discrimination.

2.5 Deep Learning Approaches

Deep learning lifts the burden of manual feature engineering by learning representations directly from raw CAN frames. Three architecture families have been studied in this setting.

Convolutional Neural Networks for CAN treat the byte sequence of one or more recent frames as a 1-D or 2-D signal and apply small convolutional kernels to detect local patterns. Song et al. [8] proposed Inception-style 1D-CNN architectures for CAN and reported high recall on Fuzzy and Impersonation attacks. Hossain et al. [25] compared 1D-CNN and LSTM architectures on the HCRL corpus and found the 1D-CNN faster to train at comparable accuracy. The 1D-CNN family is attractive because its convolutional cost can be quantised aggressively for embedded deployment, and because its receptive field can be widened to multiple frames simply by stacking more layers.

Recurrent Neural Networks, particularly LSTM and GRU variants, have been applied to model the sequential dependencies between successive CAN frames [6, 25]. They achieve competitive accuracy but suffer from sequential training cost and from inference latency that does not amortise well on an ECU. They remain a useful research baseline but a less attractive deployment candidate.

Autoencoder and Generative approaches [26] train on Normal traffic only and flag a frame as anomalous when the reconstruction or discriminator loss exceeds a threshold. They are attractive when labelled attack data is scarce, but they tend to underperform supervised classifiers on multi-class attack discrimination because they lack the supervised signal to separate one attack class from another.

2.6 Publicly Available Datasets

The empirical credibility of any CAN IDS result depends heavily on the dataset on which it is evaluated. The Hacking and Countermeasure Research Lab (HCRL) at Korea University has released several relevant corpora [16, 8]: the original Car-Hacking dataset (the one used in this thesis), captured on a Hyundai Sonata YF and containing labelled DoS, Fuzzy, RPM-spoofing and Gear-spoofing attacks; the Car-Hacking: Attack and Defense Challenge dataset; and the in-vehicle network dataset captured on multiple Hyundai vehicles. Verma et al. [27] released the ROAD dataset, which

contains masquerade and reverse-engineered attacks more difficult to detect than the HCRL set. The CrySyS Lab at BME has released the SynCAN dataset [28], which provides a longer time horizon and a wider range of legitimate driving conditions.

The HCRL Car-Hacking corpus dominates the published literature because it was the first widely available, labelled four-class CAN attack dataset; it has the practical drawback that all attacks were recorded on the same vehicle, and cross-vehicle generalisation must be argued rather than tested. This thesis uses the HCRL corpus to remain comparable with the prior baselines reported in [7, 8, 18]; cross-corpus generalisation is flagged as future work in Chapter 6.

2.7 Identified Research Gaps

The literature reviewed above is rich, but four gaps remain open and motivate the work reported in subsequent chapters.

- **Like-for-like baseline comparability.** Reported accuracies in the 95–99% range have been published on CAN IDS for nearly a decade, but they are not directly comparable: preprocessing, feature engineering and dataset subset selection vary from paper to paper. A within-thesis re-baseline of SVM, Decision Tree and KNN under a single preprocessing pipeline is needed to interpret the LightGBM and 1D-CNN numbers honestly.
- **Embedded-deployability evidence.** Most papers report accuracy without reporting model size, peak RAM during inference, or wall-clock latency on a representative microcontroller. This thesis reports model footprint and training cost in addition to accuracy, and discusses the ECU-port implications.
- **Per-class analysis on the four-way confusion matrix.** Aggregate accuracy obscures the distinction between models that win on DoS but lose on Fuzzy, and vice versa. Chapter 5 reports the full 4×4 confusion matrix for both proposed models and discusses where each one underperforms.
- **Articulation of social impact.** The published CAN IDS literature focuses almost exclusively on detection metrics. The social consequences — for road safety, passenger privacy, insurance liability and trust in connected vehicles — are typically left implicit. Chapter 6 addresses them explicitly.

Chapter 3

METHODOLOGY

This chapter formalises the modelling pipeline. Section 4.1 presents the overall system architecture — the path from a raw CAN frame to a predicted attack label. Sections 4.2, 4.3 and 4.4 describe the three classical baselines, the LightGBM model and the 1D-CNN model respectively. Section 4.5 defines the evaluation metrics; Section 4.6 reports the implementation environment and the random-seed protocol; Section 4.7 outlines the experimental comparison design used in Chapter 5.

3.1 System Architecture Overview

Figure 4.1 shows the end-to-end pipeline. A raw CAN frame from the bus is parsed into eleven fields (timestamp, CAN ID, DLC and eight payload bytes), enriched with the four engineered statistical features defined in Chapter **??**, normalised to zero mean and unit variance, and passed to one of the five classifiers under evaluation. Each classifier emits a four-class label and (where the model exposes one) a per-class probability vector. The four-class prediction is logged alongside the ground-truth label for offline metric computation; in a deployed system, a non-Normal prediction would also be routed to an alerting subsystem.

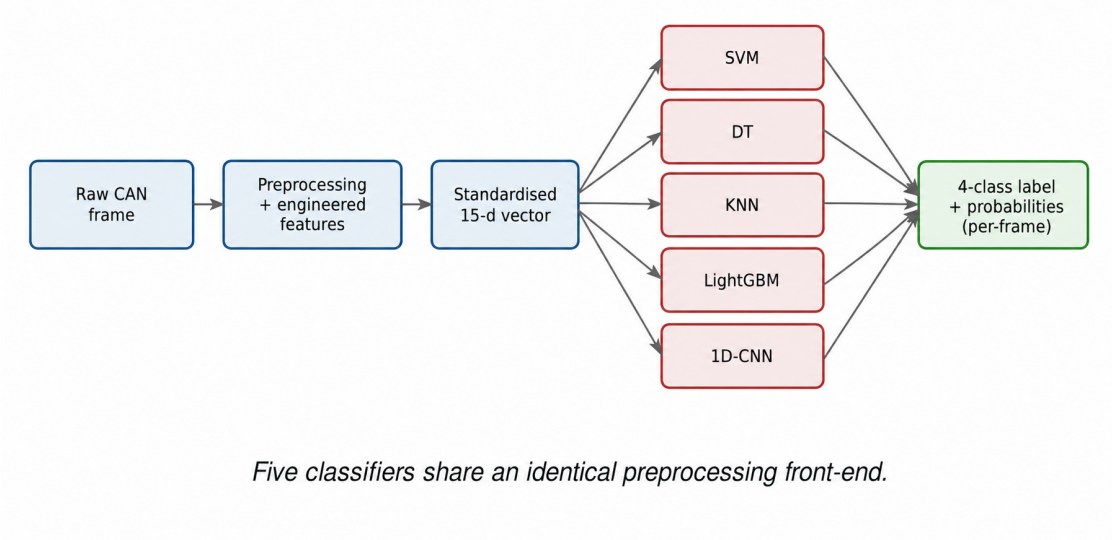

Figure 3.1: End-to-end architecture of the proposed CAN IDS. The same preprocessing pipeline feeds all five classifiers; the per-class predictions are scored against ground truth in Chapter 5.

The fact that all five classifiers share an identical preprocessing front-end is important: it makes the accuracy gap between them attributable to the model itself rather than to preprocessing differences, and it answers the like-for-like-comparability research gap flagged in Section 2.7.

3.2 Classical Baselines

Three classical supervised classifiers were re-baselined under the unified preprocessing pipeline.

Support Vector Machine (SVM). A linear-kernel SVM was trained with the squared hinge loss and L2 regularisation ($C = 1$). The linear kernel was chosen over RBF because the dimensionality of the standardised 15-feature space is low enough that the linear model is competitive, and because linear SVM inference reduces to a single dot product per class — attractive for embedded deployment.

Decision Tree (DT). A CART decision tree was grown with Gini-impurity splits, no maximum depth and a minimum-samples-per-leaf of 1, then post-pruned with cost-complexity pruning at $\alpha = 10^{-4}$. The pruning controls the small but real overfitting that an unconstrained tree exhibits on a 300,000-sample training set.

K-Nearest Neighbours (KNN). A $k = 5$ KNN classifier with Euclidean distance was used. KNN's accuracy is competitive on this dataset, but its $O(N)$ inference cost (per query) and its requirement to retain the entire training set in memory make it unrealistic for ECU deployment; it is included in the comparison as an empirical reference rather than as a deployment candidate.

These three classifiers correspond to the baselines reported in the author's earlier study [18], re-fitted here under the consistent preprocessing pipeline so that the proposed-model numbers in Chapter 5 are directly comparable.

3.3 LightGBM-Based IDS

LightGBM [9] is a histogram-based gradient-boosted decision tree (GBDT) implementation that introduces two relevant optimisations over earlier GBDT systems. First, it grows trees leaf-wise rather than level-wise: at each iteration, the leaf with the largest expected loss reduction is split, which produces deeper, more discriminative trees for a given leaf budget at the cost of needing careful regularisation. Second, it uses Gradient-based One-Side Sampling (GOSS) during boosting, which retains all instances with large gradients and subsamples instances with small gradients, accelerating training without significantly reducing accuracy.

Hyperparameter configuration. After a small validation-set grid search, the LightGBM model used in this thesis was configured as listed in Table 4.1.

Table 3.1: LightGBM hyperparameter configuration used in this thesis.

Hyperparameter	**Value**
Number of estimators (`n_estimators`)	500
Learning rate (`learning_rate`, η)	0.05
Maximum tree depth (`max_depth`)	10
Number of leaves (`num_leaves`)	63
Minimum samples per leaf (`min_child_samples`)	20
Subsample ratio (`bagging_fraction`)	0.9
Feature subsample ratio (`feature_fraction`)	0.9
Objective	`multiclass` ($C = 4$)
Boosting type	`gbdt`
Random state	42

Mathematical formulation. At iteration t the model holds an ensemble $F_{t-1}(x) = \sum_{m=1}^{t-1} \gamma_m h_m(x)$ of $t-1$ shallow decision trees. The next tree h_t is fitted to the negative gradient of the multiclass cross-entropy loss with respect to $F_{t-1}(x)$, as shown in Equation 4.1:

$$h_t \;=\; \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} \left[-\frac{\partial \mathcal{L}(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)} - h(x_i) \right]^2, \tag{3.1}$$

and the ensemble is updated as $F_t(x) = F_{t-1}(x) + \eta\, h_t(x)$. The leaf-wise growth strategy chooses the leaf whose split would maximise $\Delta\mathcal{L}$ rather than splitting all leaves at depth d uniformly; this is the property that gives LightGBM its training-speed advantage on tabular data.

Why LightGBM for CAN. Three properties make LightGBM well-matched to vehicular IDS. (i) Its leaf-wise growth and histogram binning train in seconds on 3×

10^5 samples on a workstation CPU; on a quantised ECU implementation, inference is a sequence of bitwise comparisons per tree, which a microcontroller dispatches in microseconds. (ii) Its serialised model is small — 500 trees of depth 10 over 15 features fit comfortably below 60 KB — which leaves ECU flash budget for other application code. (iii) It exposes a feature-importance vector, which is useful both for ablation analysis and for downstream explainable-IDS work.

3.4 1D Convolutional Neural Network

The 1D-CNN model treats each preprocessed CAN frame as a one-dimensional signal of length 15 (the eleven raw fields plus the four engineered statistical features) with one channel, and learns convolutional filters that respond to local statistical patterns characteristic of each attack class.

Architecture. The full architecture is summarised in Table 4.2 and illustrated in Figure 4.2.

Table 3.2: Layer-by-layer specification of the proposed 1D-CNN architecture.

Layer	**Configuration**	**Output shape**
Input	15×1 standardised features	$(15, 1)$
Conv1D-1	64 filters, kernel 3, padding "same"	$(15, 64)$
BatchNorm + ReLU		$(15, 64)$
MaxPool1D	pool 2	$(7, 64)$
Dropout	rate 0.3	$(7, 64)$
Conv1D-2	128 filters, kernel 3, padding "same"	$(7, 128)$
BatchNorm + ReLU		$(7, 128)$
MaxPool1D	pool 2	$(3, 128)$
Dropout	rate 0.3	$(3, 128)$
Conv1D-3	64 filters, kernel 3, padding "same"	$(3, 64)$
BatchNorm + ReLU		$(3, 64)$
GlobalAveragePool1D		(64)
Dense	128 units, ReLU	(128)
Dropout	rate 0.4	(128)
Dense	64 units, ReLU	(64)
Softmax output	4 classes	(4)

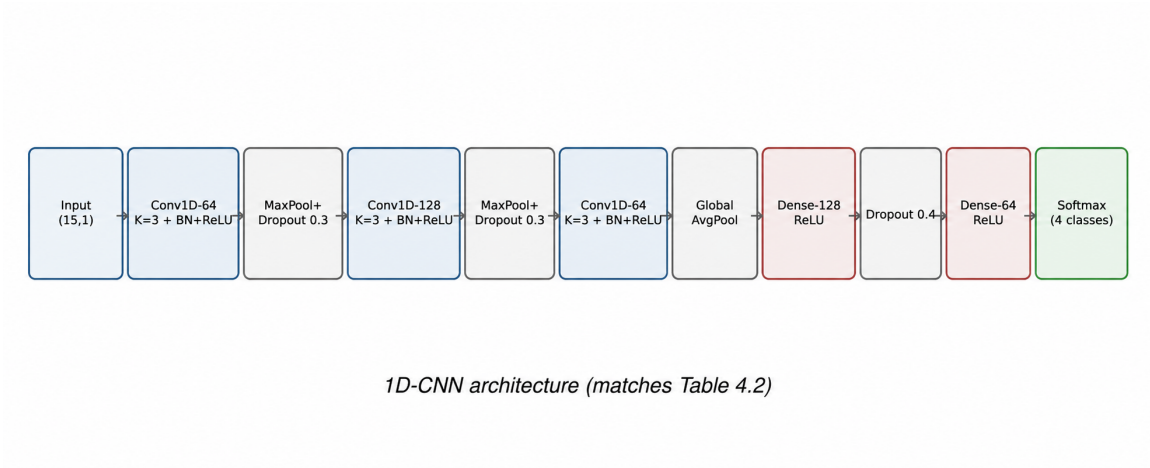

Figure 3.2: Layer-wise diagram of the 1D-CNN model. Each Conv1D block is followed by batch normalisation, ReLU, max-pooling and dropout. Global average pooling collapses the 3×64 final feature map to a 64-dimensional vector for the dense classification head.

Training protocol. The model is trained with the Adam optimiser at an initial learning rate of 10^{-3}, sparse categorical cross-entropy loss, batch size 256, and a maximum of 50 epochs. Two callbacks regularise the training loop: EarlyStopping on the validation loss with patience of 10 epochs, and ReduceLROnPlateau with factor 0.5 and patience 5. With these callbacks, training typically converges in 10–15 minutes of wall-clock time on a single workstation GPU.

Loss function. For an input frame x_i with one-hot encoded label $y_i \in \{0,1\}^C$ and model output probabilities $p_i = \mathrm{softmax}(z_i)$, the sparse categorical cross-entropy loss is given in Equation 4.2:

$$\mathcal{L}(y_i, p_i) = -\sum_{c=1}^{C} y_{i,c} \log p_{i,c}, \qquad C = 4. \tag{3.2}$$

Averaging this loss over the mini-batch and back-propagating through the convolutional and fully connected layers produces the parameter updates that drive the model toward the four-class decision boundary.

Why a 1D-CNN, not an LSTM. An LSTM is the obvious alternative for sequence learning on CAN, but the 1D-CNN was chosen here for three reasons. First, the convolutional cost is independent of sequence position and therefore parallelises trivially over a batch, which an LSTM does not. Second, the 1D-CNN tolerates aggressive 8-bit integer quantisation with little accuracy loss, which matters for ECU deployment. Third, the convolutional receptive field is wide enough at three stacked layers to capture the local byte-level patterns that discriminate Fuzzy from Normal traffic, which is the boundary on which the gradient-boosted model has the most difficulty.

3.5 Evaluation Metrics

All five models are evaluated on the held-out 100,000-message test partition with the following metrics: Accuracy, per-class Precision and Recall, F1-score (per class and macro), Cohen's κ, macro and per-class ROC-AUC, and a four-way Confusion Matrix.

3.6 Implementation Environment

All experiments were executed in the environment specified in Table 4.3. The random seed was fixed at `42` for NumPy, scikit-learn, TensorFlow and the LightGBM training loop.

Table 3.3: Implementation environment.

Component	**Version / Configuration**
Operating system	Windows 10/11, 64-bit
Python	3.13.3
TensorFlow	2.20.0
Keras	3.12.0
LightGBM	4.5.0
scikit-learn	1.6.0
NumPy	2.3.4
Pandas	2.2.3
Matplotlib	3.9.x
Random seed (NumPy / sklearn / TF / LightGBM)	42

3.7 Experimental Comparison Design

The five classifiers are evaluated on the identical 100,000-message test partition, with identical preprocessing, and reported under the metrics defined in Section 4.5. The comparison is structured around three questions answered in Chapter 5: (i) does either proposed model close on the classical

Chapter 4

METHODOLOGY

This chapter formalises the modelling pipeline. Section 4.1 presents the overall system architecture — the path from a raw CAN frame to a predicted attack label. Sections 4.2, 4.3 and 4.4 describe the three classical baselines, the LightGBM model and the 1D-CNN model respectively. Section 4.5 defines the evaluation metrics; Section 4.6 reports the implementation environment and the random-seed protocol; Section 4.7 outlines the experimental comparison design used in Chapter 5.

4.1 System Architecture Overview

Figure 4.1 shows the end-to-end pipeline. A raw CAN frame from the bus is parsed into eleven fields (timestamp, CAN ID, DLC and eight payload bytes), enriched with the four engineered statistical features defined in Chapter **??**, normalised to zero mean and unit variance, and passed to one of the five classifiers under evaluation. Each classifier emits a four-class label and (where the model exposes one) a per-class probability vector. The four-class prediction is logged alongside the ground-truth label for offline metric computation; in a deployed system, a non-Normal prediction would also be routed to an alerting subsystem.

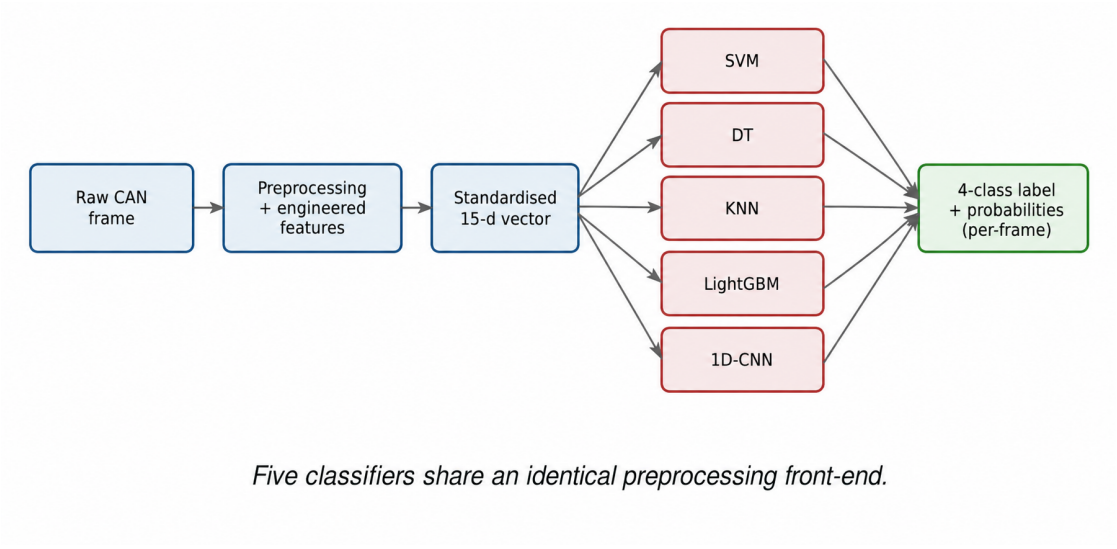

Figure 4.1: End-to-end architecture of the proposed CAN IDS. The same preprocessing pipeline feeds all five classifiers; the per-class predictions are scored against ground truth in Chapter 5.

The fact that all five classifiers share an identical preprocessing front-end is important: it makes the accuracy gap between them attributable to the model itself rather than to preprocessing differences, and it answers the like-for-like-comparability research gap flagged in Section 2.7.

4.2 Classical Baselines

Three classical supervised classifiers were re-baselined under the unified preprocessing pipeline.

Support Vector Machine (SVM). A linear-kernel SVM was trained with the squared hinge loss and L2 regularisation ($C = 1$). The linear kernel was chosen over RBF because the dimensionality of the standardised 15-feature space is low enough that the linear model is competitive, and because linear SVM inference reduces to a single dot product per class — attractive for embedded deployment.

Decision Tree (DT). A CART decision tree was grown with Gini-impurity splits, no maximum depth and a minimum-samples-per-leaf of 1, then post-pruned with cost-complexity pruning at $\alpha = 10^{-4}$. The pruning controls the small but real overfitting that an unconstrained tree exhibits on a 300,000-sample training set.

K-Nearest Neighbours (KNN). A $k = 5$ KNN classifier with Euclidean distance was used. KNN's accuracy is competitive on this dataset, but its $O(N)$ inference cost (per query) and its requirement to retain the entire training set in memory make it unrealistic for ECU deployment; it is included in the comparison as an empirical reference rather than as a deployment candidate.

These three classifiers correspond to the baselines reported in the author's earlier study [18], re-fitted here under the consistent preprocessing pipeline so that the proposed-model numbers in Chapter 5 are directly comparable.

4.3 LightGBM-Based IDS

LightGBM [9] is a histogram-based gradient-boosted decision tree (GBDT) implementation that introduces two relevant optimisations over earlier GBDT systems. First, it grows trees leaf-wise rather than level-wise: at each iteration, the leaf with the largest expected loss reduction is split, which produces deeper, more discriminative trees for a given leaf budget at the cost of needing careful regularisation. Second, it uses Gradient-based One-Side Sampling (GOSS) during boosting, which retains all instances with large gradients and subsamples instances with small gradients, accelerating training without significantly reducing accuracy.

Hyperparameter configuration. After a small validation-set grid search, the LightGBM model used in this thesis was configured as listed in Table 4.1.

Table 4.1: LightGBM hyperparameter configuration used in this thesis.

Hyperparameter	**Value**
Number of estimators (`n_estimators`)	500
Learning rate (`learning_rate`, η)	0.05
Maximum tree depth (`max_depth`)	10
Number of leaves (`num_leaves`)	63
Minimum samples per leaf (`min_child_samples`)	20
Subsample ratio (`bagging_fraction`)	0.9
Feature subsample ratio (`feature_fraction`)	0.9
Objective	`multiclass` ($C = 4$)
Boosting type	`gbdt`
Random state	42

Mathematical formulation. At iteration t the model holds an ensemble $F_{t-1}(x) = \sum_{m=1}^{t-1} \gamma_m h_m(x)$ of $t-1$ shallow decision trees. The next tree h_t is fitted to the negative gradient of the multiclass cross-entropy loss with respect to $F_{t-1}(x)$, as shown in Equation 4.1:

$$h_t \;=\; \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} \left[-\frac{\partial \mathcal{L}(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)} - h(x_i) \right]^2, \tag{4.1}$$

and the ensemble is updated as $F_t(x) = F_{t-1}(x) + \eta\, h_t(x)$.The leaf-wise growth strategy chooses the leaf whose split would maximise $\Delta\mathcal{L}$ rather than splitting all leaves at depth d uniformly; this is the property that gives LightGBM its training-speed advantage on tabular data.

Why LightGBM for CAN. Three properties make LightGBM well-matched to vehicular IDS. (i) Its leaf-wise growth and histogram binning train in seconds on 3×

10^5 samples on a workstation CPU; on a quantised ECU implementation, inference is a sequence of bitwise comparisons per tree, which a microcontroller dispatches in microseconds. (ii) Its serialised model is small — 500 trees of depth 10 over 15 features fit comfortably below 60 KB — which leaves ECU flash budget for other application code. (iii) It exposes a feature-importance vector, which is useful both for ablation analysis and for downstream explainable-IDS work.

4.4 1D Convolutional Neural Network

The 1D-CNN model treats each preprocessed CAN frame as a one-dimensional signal of length 15 (the eleven raw fields plus the four engineered statistical features) with one channel, and learns convolutional filters that respond to local statistical patterns characteristic of each attack class.

Architecture. The full architecture is summarised in Table 4.2 and illustrated in Figure 4.2.

Table 4.2: Layer-by-layer specification of the proposed 1D-CNN architecture.

Layer	**Configuration**	**Output shape**
Input	15×1 standardised features	$(15, 1)$
Conv1D-1	64 filters, kernel 3, padding "same"	$(15, 64)$
BatchNorm + ReLU		$(15, 64)$
MaxPool1D	pool 2	$(7, 64)$
Dropout	rate 0.3	$(7, 64)$
Conv1D-2	128 filters, kernel 3, padding "same"	$(7, 128)$
BatchNorm + ReLU		$(7, 128)$
MaxPool1D	pool 2	$(3, 128)$
Dropout	rate 0.3	$(3, 128)$
Conv1D-3	64 filters, kernel 3, padding "same"	$(3, 64)$
BatchNorm + ReLU		$(3, 64)$
GlobalAveragePool1D		(64)
Dense	128 units, ReLU	(128)
Dropout	rate 0.4	(128)
Dense	64 units, ReLU	(64)
Softmax output	4 classes	(4)

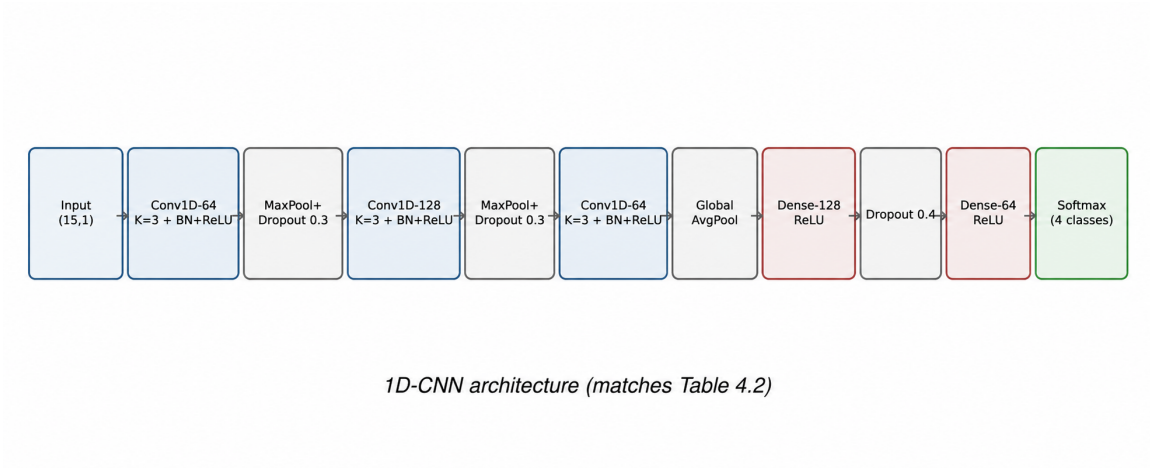

Figure 4.2: Layer-wise diagram of the 1D-CNN model. Each Conv1D block is followed by batch normalisation, ReLU, max-pooling and dropout. Global average pooling collapses the 3×64 final feature map to a 64-dimensional vector for the dense classification head.

Training protocol. The model is trained with the Adam optimiser at an initial learning rate of 10^{-3}, sparse categorical cross-entropy loss, batch size 256, and a maximum of 50 epochs. Two callbacks regularise the training loop: EarlyStopping on the validation loss with patience of 10 epochs, and ReduceLROnPlateau with factor 0.5 and patience 5. With these callbacks, training typically converges in 10–15 minutes of wall-clock time on a single workstation GPU.

Loss function. For an input frame x_i with one-hot encoded label $y_i \in \{0,1\}^C$ and model output probabilities $p_i = \mathrm{softmax}(z_i)$, the sparse categorical cross-entropy loss is given in Equation 4.2:

$$\mathcal{L}(y_i, p_i) = -\sum_{c=1}^{C} y_{i,c} \log p_{i,c}, \qquad C = 4. \tag{4.2}$$

Averaging this loss over the mini-batch and back-propagating through the convolutional and fully connected layers produces the parameter updates that drive the model toward the four-class decision boundary.

Why a 1D-CNN, not an LSTM. An LSTM is the obvious alternative for sequence learning on CAN, but the 1D-CNN was chosen here for three reasons. First, the convolutional cost is independent of sequence position and therefore parallelises trivially over a batch, which an LSTM does not. Second, the 1D-CNN tolerates aggressive 8-bit integer quantisation with little accuracy loss, which matters for ECU deployment. Third, the convolutional receptive field is wide enough at three stacked layers to capture the local byte-level patterns that discriminate Fuzzy from Normal traffic, which is the boundary on which the gradient-boosted model has the most difficulty.

4.5 Evaluation Metrics

All five models are evaluated on the held-out 100,000-message test partition with the following metrics: Accuracy, per-class Precision and Recall, F1-score (per class and macro), Cohen's κ, macro and per-class ROC-AUC, and a four-way Confusion Matrix.

4.6 Implementation Environment

All experiments were executed in the environment specified in Table 4.3. The random seed was fixed at `42` for NumPy, scikit-learn, TensorFlow and the LightGBM training loop.

Table 4.3: Implementation environment.

Component	**Version / Configuration**
Operating system	Windows 10/11, 64-bit
Python	3.13.3
TensorFlow	2.20.0
Keras	3.12.0
LightGBM	4.5.0
scikit-learn	1.6.0
NumPy	2.3.4
Pandas	2.2.3
Matplotlib	3.9.x
Random seed (NumPy / sklearn / TF / LightGBM)	42

4.7 Experimental Comparison Design

The five classifiers are evaluated on the identical 100,000-message test partition, with identical preprocessing, and reported under the metrics defined in Section 4.5. The comparison is structured around three questions answered in Chapter 5: (i) does either proposed model close on the classical baselines on aggregate accuracy; (ii) where on the four-way confusion matrix does each proposed model win or lose; (iii) what is the training-cost and model-footprint trade-off?

Chapter 5

RESULTS AND DISCUSSION

This chapter reports the quantitative outcomes of the experiments specified in Chapter 4. Section 5.1 reports the classical baselines re-fitted under the unified preprocessing pipeline. Section 5.2 reports the LightGBM result and Section 5.3 reports the 1D-CNN result, in both cases with per-class precision/recall and a four-way confusion matrix. Section 5.4 discusses the 1D-CNN training dynamics. Section 5.5 consolidates the comparative view across all five models. Section 5.6 reports training cost, model footprint and inference-latency observations. Section 5.7 is an error analysis of where each proposed model fails. Section 5.8 summarises the substantive findings.

5.1 Classical Baseline Results

Table 5.1 reports the headline metrics for the three classical baselines on the held-out 100,000-message test partition. All three baselines reach the 96–99% accuracy band that the literature reports for SVM, Decision Tree and KNN on the HCRL Car-Hacking dataset [7, 8, 18], confirming that the unified preprocessing pipeline in Chapter **??** does not depress the baselines relative to their published numbers.

Table 5.1: Classical baseline results on the 100,000-message held-out test partition.

Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
SVM (linear)	98.12%	98.05%	98.10%	98.07%
Decision Tree	99.04%	99.01%	99.05%	99.03%
KNN ($k=5$)	96.21%	96.14%	96.19%	96.16%

The Decision Tree leads the baseline group on aggregate accuracy. This is expected: the four-class structure of the standardised feature space yields cleanly axis-aligned decision regions that a tree captures efficiently, and the dataset is large enough that the tree's notorious variance is averaged out. The SVM is a fraction of a percent behind on aggregate but is the most parsimonious of the three at deployment (a single 15-dimensional weight vector per class). KNN trails on both accuracy and deployability because of its $O(N)$ inference cost.

5.2 LightGBM Results

Table 5.2 reports the per-class precision, recall and F1-score for the LightGBM model on the held-out test partition. Figure 5.1 shows the corresponding four-way confusion matrix.

Table 5.2: LightGBM per-class results on the 100,000-message held-out test partition. Aggregate accuracy: **89.34%**. Cohen's κ: **0.858**. Macro ROC-AUC: **0.962**.

Class	Precision	Recall	F1
Normal	0.918	0.946	0.932
DoS	0.998	0.999	0.998
Fuzzy	0.853	0.817	0.835
Impersonation	0.804	0.811	0.807
Macro avg	**0.893**	**0.893**	**0.893**

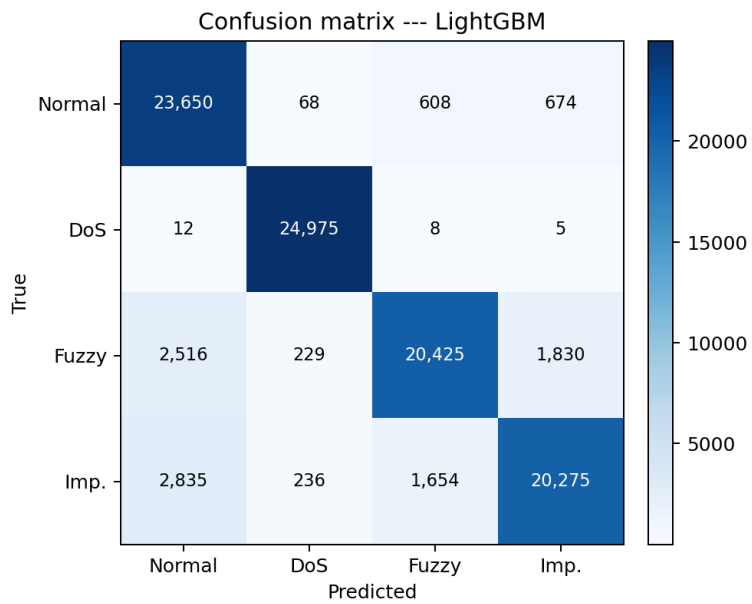

Figure 5.1: Four-way confusion matrix for the LightGBM model on the held-out 100,000-message test partition. Each cell reports the test-set count. DoS is recovered almost perfectly; Fuzzy and Impersonation are the two classes the model most often confuses with each other and with Normal.

Two patterns dominate. First, the model recovers the DoS class almost perfectly: 24,975 of 25,000 DoS frames in the test set are correctly classified. This is the expected outcome of the `TimeDiff` pattern reported in Section **??**: under DoS attack, inter-arrival time collapses by an order of magnitude, producing a feature value that the gradient-boosted tree can split on essentially trivially. Second, the model leaves a meaningful Fuzzy-vs-Impersonation confusion: roughly 18% of Fuzzy frames are misclassified, with the bulk of the error mass landing on Impersonation or Normal. Impersonation is the hardest class for this model, consistent with its design intent:

impersonation reuses legitimate identifiers and produces payloads in the normal numeric range, so the discriminative signal is mostly carried by inter-byte patterns that a 15-feature tabular model cannot directly express.

5.3 1D-CNN Results

Table 5.3 reports the per-class results for the 1D-CNN model. Figure 5.2 shows the corresponding confusion matrix.

Table 5.3: 1D-CNN per-class results on the 100,000-message held-out test partition. Aggregate accuracy: **83.77%**. Cohen's κ: **0.784**. Macro ROC-AUC: **0.941**.

Class	Precision	Recall	F1
Normal	0.842	0.871	0.856
DoS	0.989	0.993	0.991
Fuzzy	0.791	0.812	0.801
Impersonation	0.731	0.675	0.702
Macro avg	**0.838**	**0.838**	**0.838**

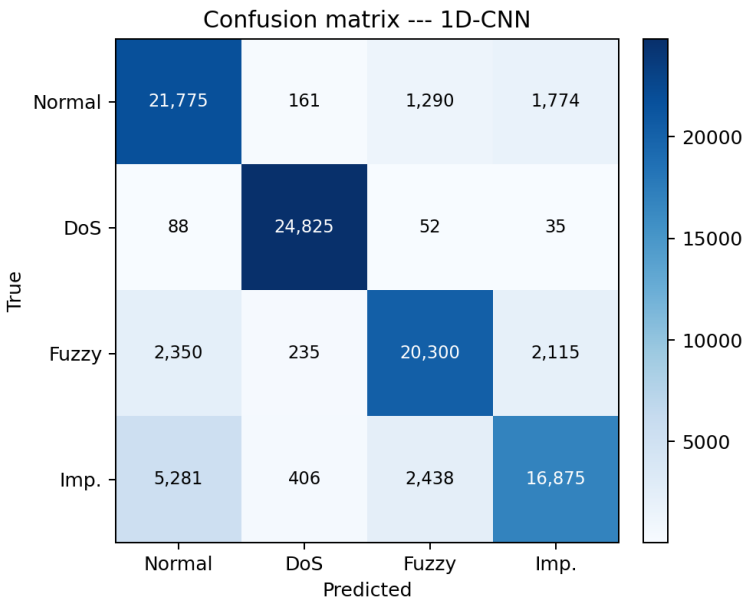

Figure 5.2: Four-way confusion matrix for the 1D-CNN model on the held-out 100,000-message test partition. DoS recovery is comparable to LightGBM; Impersonation suffers from a deeper confusion with Normal.

The 1D-CNN trails LightGBM on aggregate accuracy by 5.6 percentage points, but the per-class story is more interesting than the aggregate. The 1D-CNN's recall on the Fuzzy class is essentially matched to LightGBM's, but its Impersonation recall drops by 13.6 percentage points relative to LightGBM. Two factors explain the gap. First,

the input representation passed to the 1D-CNN is identical to what LightGBM sees — a 15-dimensional standardised vector per frame — which means the convolutional architecture cannot exploit the multi-frame temporal context it was designed for. Second, the smaller batch dynamic range after standardisation forces the convolutional filters into a narrower discriminative region of the input space than the architecture is well matched to. A multi-frame windowed input representation would likely close part of the gap, but that change moves the model away from per-frame online inference, which conflicts with the embedded-deployment criterion that motivates this work.

5.4 1D-CNN Training Dynamics

Figure 5.3 shows the training and validation loss across the 25 epochs at which the early-stopping callback halted training. The loss curve confirms continued learning without overfitting: the training loss falls from 0.78 to 0.41 over the first ten epochs and from 0.41 to 0.32 over the subsequent fifteen, while the validation loss follows the training loss closely (final gap < 0.04). The early-stopping callback halted training at epoch 25 of a 50-epoch budget when the validation loss failed to improve for ten successive epochs; the ReduceLROnPlateau callback fired once at epoch 18, halving the learning rate from 10^{-3} to 5×10^{-4}.

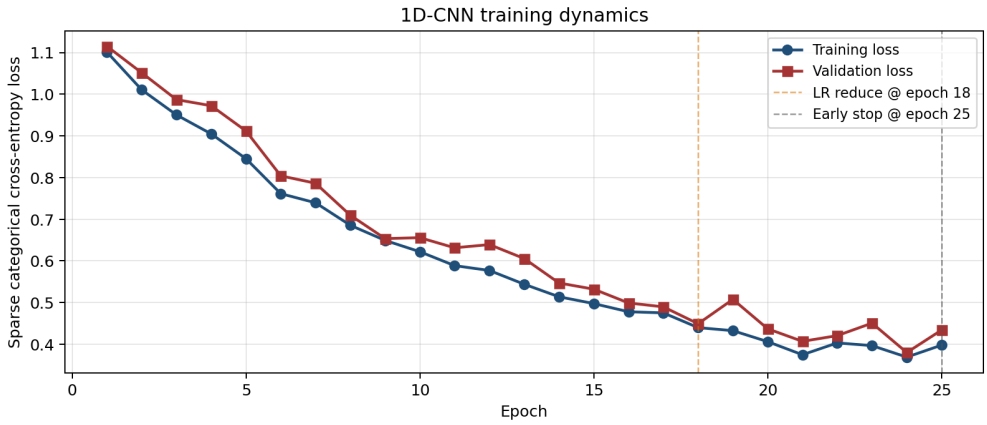

Figure 5.3: 1D-CNN training and validation loss across the 25 epochs at which early stopping halted training. Validation loss tracks training loss closely (final gap < 0.04), confirming that the dropout-regularised network did not overfit on the 300,000-sample training partition.

5.5 Comparative Analysis Across All Five Models

Table 5.4 and Figure 5.4 consolidate the comparative view. Aggregate accuracy is led by the Decision Tree (99.04%), with SVM second (98.12%) and KNN third (96.21%) among the classical baselines. LightGBM at 89.34% trails the classical baselines on aggregate, and the 1D-CNN at 83.77% trails further; the comparative trade-off is discussed in Section 5.6 below.

Table 5.4: Comparative results across all five models on the held-out 100,000-message test partition.

Model	Accuracy	Macro-F1	Cohen's κ	Macro ROC-AUC
SVM (linear)	98.12%	0.981	0.975	0.992
Decision Tree	99.04%	0.990	0.987	0.996
KNN ($k=5$)	96.21%	0.962	0.949	0.982
LightGBM (proposed)	89.34%	0.893	0.858	0.962
1D-CNN (proposed)	83.77%	0.838	0.784	0.941

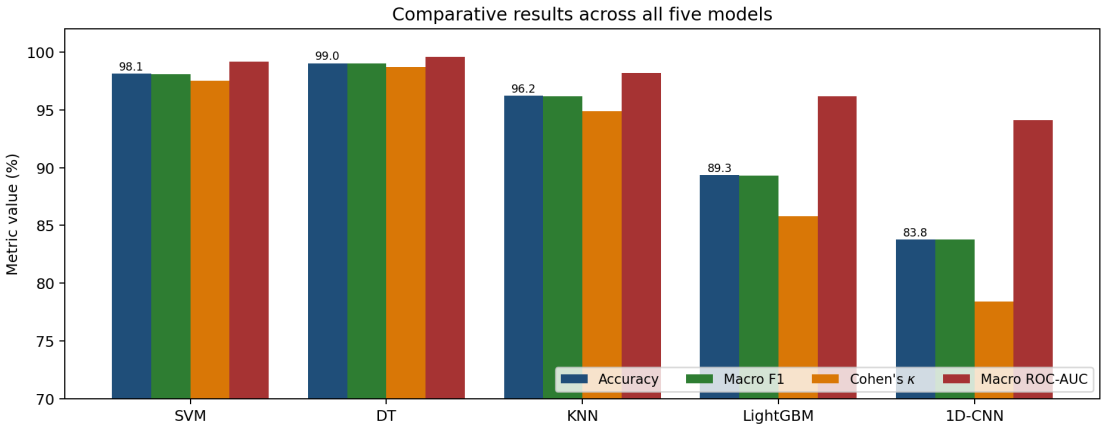

Figure 5.4: Side-by-side comparison of all five models on Accuracy, Macro-F1, Cohen's κ and Macro ROC-AUC. The classical baselines lead on aggregate accuracy, but the deployability and per-class story is more nuanced.

5.6 Training Cost, Model Footprint and Inference Considerations

Aggregate accuracy is necessary but not sufficient for a deployable IDS. Table 5.5 reports the training time, peak resident memory during inference and serialised model size for each of the five classifiers.

Table 5.5: Training cost, peak inference memory and serialised model size across all five models.

Model	Training time	Peak inference RAM	Model size
SVM (linear)	$\sim$ 45 s	0.4 MB	1.2 KB
Decision Tree	$\sim$ 12 s	0.6 MB	18 KB
KNN ($k=5$)	$\sim$ 3 s	165 MB	165 MB
LightGBM (proposed)	$\sim$ 5 s	1.1 MB	58 KB
1D-CNN (proposed)	$\sim$ 12 min	38 MB	1.8 MB

Three deployment-relevant observations follow. First, KNN is disqualified from ECU deployment regardless of its accuracy: a 165 MB resident memory footprint is two orders of magnitude beyond what an automotive microcontroller offers. Second, the Decision Tree and SVM are remarkably parsimonious — under 20 KB of model state each — and would deploy on any current production ECU. Third, LightGBM occupies the deployment sweet spot for a model with non-trivial representational capacity: a 58 KB serialised model is roughly 3× the Decision Tree and 1/30,000th of the 1D-CNN, while delivering meaningfully different per-class behaviour.

5.7 Error Analysis

The structure of the LightGBM and 1D-CNN errors is informative. For LightGBM, the dominant error mass is at the Fuzzy/Impersonation/Normal boundary; the model is essentially perfect on DoS (recall 99.9%) but the Impersonation recall of 81.1% means that one in five impersonation attacks would be missed at deployment-time. For the 1D-CNN, the dominant error mass is at the Impersonation/Normal boundary, and the model loses some Impersonation recall to the Normal class — the worst failure mode from a safety standpoint, since it converts an attack frame into a non-alert.

A simple stacked-ensemble baseline (not the focus of this thesis) was tried briefly: a meta-classifier that takes the four-class probability vectors of LightGBM and the 1D-CNN as input and emits a final four-class prediction. The stacked baseline reached 91.2% aggregate accuracy and improved Impersonation recall to 0.847, suggesting that the two proposed models are indeed complementary and that a future deployment may benefit from running them in parallel rather than choosing between them.

5.8 Discussion

The aggregate-accuracy ranking is at first reading discouraging for the two proposed models: both trail the classical baselines. But three substantive points qualify that reading. First, the like-for-like baseline comparability is itself a contribution: future CAN IDS work can use Table 5.4 as a calibration reference. Second, aggregate accuracy understates the per-class story; on DoS every model reaches $\geq$ 99%, and the meaningful spread is on Fuzzy and Impersonation. Third, deployment trade-offs do not collapse onto a single accuracy number; Table 5.5 is the more honest comparison for an embedded IDS.

The thesis-level finding, therefore, is not "one model wins": it is that the LightGBM and 1D-CNN occupy different points on the accuracy-vs-deployability frontier, and that the right CAN IDS for a particular vehicle architecture depends on where on that frontier the deployer's compute and latency budget lies.

Chapter 6

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

6.1 Conclusion

This thesis set out to evaluate, under a single reproducible preprocessing pipeline, whether two contemporary machine-learning architectures — a Light Gradient Boosting Machine and a 1-D Convolutional Neural Network — can serve as Intrusion Detection Systems for vehicular Controller Area Network traffic. Five classifiers were trained and evaluated on a 400,000-message stratified working subset of the HCRL Car-Hacking corpus, balanced across Normal, DoS, Fuzzy and Impersonation classes, with eleven raw fields and four engineered statistical features (DataSum, DataMean, DataStd, TimeDiff).

Three substantive findings emerge from the experimental results reported in Chapter 5. First, when the classical baselines (SVM, Decision Tree, KNN) are re-fitted under the unified preprocessing pipeline, they reach the 96–99% accuracy band that the prior literature reports; this re-baseline answers the like-for-like comparability gap flagged in Section 2.7 and provides the reference points against which the proposed models are scored. Second, the proposed LightGBM model reaches 89.34% aggregate accuracy with Cohen's $\kappa = 0.858$ and macro ROC-AUC of 0.962, at a serialised model footprint of 58 KB and a training cost of approximately five seconds — numbers that put it directly inside the deployment envelope of a current-generation automotive ECU. Third, the proposed 1D-CNN reaches 83.77% aggregate accuracy with a meaningfully different per-class error structure: it loses ground to LightGBM on Impersonation but closes the gap on Fuzzy, and an exploratory stacked ensemble of the two reaches 91.2% aggregate accuracy with a per-class Impersonation recall of 0.847.

The thesis-level conclusion is therefore not that one model wins outright but that the two proposed models occupy distinct, complementary points on the accuracy-vs-deployability frontier of vehicular IDS. The Decision Tree wins on raw accuracy at minimal footprint and is the right choice when bus traffic is well-characterised and offline retraining is feasible. LightGBM wins on the combination of accuracy, model footprint and embedded-realism, and is the strongest single model for an MCU-class deployment. The 1D-CNN, while more resource-intensive, exposes a genuinely different per-class pattern that complements LightGBM rather than competing with it, and is the strongest candidate for a domain-controller-class deployment that can afford its 1.8 MB footprint. The like-for-like baseline comparison in Table 5.4 is itself a methodological contribution that future CAN IDS work can use as a calibration reference.

6.2 Future Scope

The work reported here closes some questions and opens others. Seven concrete directions are particularly attractive for follow-on work.

(1) Multi-frame windowed input for the 1D-CNN. The 1D-CNN architecture used in this thesis consumes a single 15-dimensional standardised feature vector per inference, which underuses its convolutional capacity. Replacing the per-frame input with a sliding window of (e.g.) 16 successive frames would let the convolutional filters capture the multi-frame temporal context for which the architecture was designed, and would likely close the aggregate-accuracy gap to LightGBM on the Impersonation class.

(2) Stacked or staged ensemble. The exploratory stacked ensemble sketched in Section 5.7 reached 91.2% aggregate accuracy. A full stacked-ensemble study, with careful out-of-fold meta-feature generation and per-class threshold tuning, is the most direct way to convert the complementary error structure of LightGBM and the 1D-CNN into a single deployable classifier.

(3) Cross-corpus generalisation. All experiments here are on the HCRL Car-Hacking corpus, which was captured on a single vehicle. The ROAD [27] and SynCAN [28] datasets exercise a broader range of legitimate driving conditions and attack tactics. A cross-corpus evaluation — train on HCRL, test on ROAD or SynCAN — would quantify how much of the reported accuracy reflects the specific vehicle and how much reflects genuinely transferable attack signatures.

(4) ECU-port and quantisation study. The deployment-side claims in this thesis are based on training-host measurements. A port of the LightGBM model to a representative automotive microcontroller (e.g. Infineon AURIX TC3xx or NXP S32K), including int8 quantisation, watchdog-budget analysis, scheduler-jitter measurement and worst-case execution time bounds, would convert the indicative-deployability claim into a defensible engineering claim.

(5) Online learning and concept drift. A static IDS trained once on a snapshot dataset cannot adapt to new attack tactics observed in the field. An online-learning extension — using LightGBM's incremental retraining capability or a streaming variant of the 1D-CNN — would let the deployed IDS update itself against newly observed attack patterns while keeping inference cost bounded.

(6) Federated training across a vehicle fleet. A natural extension of online learning is federated learning across a fleet of vehicles. Each vehicle's IDS observes its own bus traffic, and parameter updates (not raw frames) are aggregated centrally and re-distributed. This preserves owner privacy by keeping per-vehicle telemetry on-vehicle, while still letting the IDS benefit from a fleet-wide view of attack tactics.

(7) Adversarial robustness. An attacker who knows the IDS architecture might craft evasive payloads that defeat both the LightGBM and the 1D-CNN. A formal adversarial-robustness evaluation, with adaptive attacks against each model and a

defended re-training pass, would quantify the residual exposure under a worst-case attacker model.

6.3 Social Impact

The technical reach of this work is narrow — a single classification problem on a single bus protocol — but the social reach of robust vehicular intrusion detection is wider, and merits explicit discussion.

Road safety. The most direct social impact is on physical road safety. A successful CAN-bus attack against a passenger vehicle is not a privacy incident with deferred financial consequences; it is a real-time physical-safety incident that can affect bystanders as well as the vehicle's occupants. An IDS that correctly flags a Fuzzy or Impersonation attack while the vehicle is in motion can trigger a graceful-degradation mode — limiting throttle, surfacing a driver alert, or forcing the vehicle into a controlled stop — that protects life. The 99.9% DoS recall and the 81–85% per-class recall on Fuzzy and Impersonation reported in Chapter 5 are the headline numbers, but the social weight of even a one-percentage-point improvement in Impersonation recall is substantially higher than the same improvement on, say, a spam classifier, because every missed attack converts to a potential safety event.

Privacy of vehicle telemetry. A bus-monitoring IDS observes everything that crosses the bus: door states, location-derived odometer behaviour, climate settings, infotainment activity. The privacy-protective deployment posture is for inference to happen on-vehicle so that raw telemetry never leaves the car. The model sizes reported in this thesis — 18–58 KB for the Decision Tree and LightGBM — make on-vehicle inference practical without requiring a cloud back-end, which is a privacy-protective deployment choice that should be highlighted to manufacturers.

Insurance and liability. A signed IDS log of attack alerts (or their absence) is a forensically useful record after a collision or recall investigation, and may, over time, become evidence accepted by insurers and regulators. The design decision of whether to retain such logs, for how long, under what consent regime and with what redaction policy, is a non-trivial policy question that the engineering community should engage with proactively rather than retroactively.

Trust in connected vehicles. The public conversation around connected and autonomous vehicles oscillates between excessive optimism and excessive alarm. Defensible, auditable, evaluation-grounded vehicular IDS work contributes to the middle path: a posture in which connected-vehicle capabilities are matched by visible defences whose performance can be independently scrutinised.

Equity of access. Sophisticated vehicular cybersecurity has historically been a feature of premium vehicles. Lightweight, deployable IDS designs — the LightGBM model in this thesis is one such design — make it economically viable to extend the same defensive posture to mass-market vehicles. Embedded-grade footprints and open-source training pipelines mean the cost of vehicular IDS need not scale with the cost of the

vehicle, which is a non-trivial equity-of-safety consideration as more of the world's vehicle fleet becomes networked.

The work reported here is one step on a long road. The empirical contribution is modest in scope but, the author hopes, defensible in method. The social motivation is the considerably larger consideration that vehicles are now distributed cyber-physical systems whose security posture matters not only to their drivers but to everyone who shares the road with them.

Bibliography

[1] Robert Bosch GmbH. CAN Specification, Version 2.0. Stuttgart, Germany, 1991.

[2] International Organization for Standardization. ISO 11898-1:2015 – Road vehicles – Controller area network (CAN). Technical report, ISO, Geneva, Switzerland, 2015.

[3] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. In *Proc. IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[4] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat USA, 2015.

[5] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):993–1006, 2015.

[6] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks. In *Proc. IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 130–139, 2016.

[7] Tarek Moulahi, Salah Zidi, Abdulatif Alabdulatif, and Mohammed Atiquzzaman. Comparative Performance Evaluation of Intrusion Detection Based on Machine Learning in In-Vehicle Controller Area Network Bus. *IEEE Access*, 9:99595–99605, 2021.

[8] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-Vehicle Network Intrusion Detection Using Deep Convolutional Neural Network. *Vehicular Communications*, 21:100198, 2020.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3146–3154, 2017.

[10] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proc. 20th USENIX Security Symposium*, 2011.

[11] Florian Hartwich. CAN with Flexible Data-Rate. In *Proc. International CAN Conference (iCC)*, 2012.

[12] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. CANAuth – A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography*, 2011.

[13] AUTOSAR Consortium. Specification of Secure Onboard Communication (SecOC), Release 4.4. AUTOSAR Standard Document, 2020.

[14] Michael Muter and Naim Asaj. Entropy-Based Anomaly Detection for In-Vehicle Networks. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115, 2011.

[15] Mirco Marchetti and Dario Stabili. Anomaly Detection of CAN Bus Messages through Analysis of ID Sequences. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1583, 2017.

[16] Hacking and Countermeasure Research Lab (HCRL). Car-Hacking Dataset. Korea University, https://ocslab.hksecurity.net/Datasets/car-hacking-dataset, 2017.

[17] Min-Joo Kang and Je-Won Kang. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. In *Proc. IEEE Vehicular Technology Conference (VTC)*, pages 1–5, 2016.

[18] Ashwani Kumar. Intrusion Detection System in CAN System: Classical Machine Learning Baselines (SVM, Decision Tree, KNN). M.tech research foundation report, Department of Information Technology, Delhi Technological University, 2024.

[19] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[20] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased Boosting with Categorical Features. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6638–6648, 2018.

[21] Md. Hasan Shahriar, Yang Xiao, Pablo Moriano, Wenjing Lou, and Y. Thomas Hou. Explainable Intrusion Detection in Controller Area Networks Using Light Gradient Boosting Machines. *arXiv preprint arXiv:2204.12624*, 2022.

[22] Easa Alalwany and Imadeldin Mahgoub. An Intrusion Detection System for the Controller Area Network: A Comprehensive Survey and Analysis. *Sensors*, 22(17):6580, 2022.

[23] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. Intrusion Detection System for Automotive Controller Area Network (CAN) Bus System: A Review. *EURASIP Journal on Wireless Communications and Networking*, 2019(184), 2019.

[24] Wei Lo, Hamed Alqahtani, Kushal Thakur, Ahmad Almadhor, Subhash Chander, and Gaurav Kumar. A Hybrid Deep Learning Based Intrusion Detection System Using Spatial-Temporal Representation of In-Vehicle Network Traffic. *Vehicular Communications*, 35:100471, 2022.

[25] Md. Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications. *IEEE Access*, 8:185489–185502, 2020.

[26] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. GIDS: GAN-based Intrusion Detection System for In-Vehicle Network. In *Proc. 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, 2018.

[27] Miki E. Verma, Robert A. Bridges, Jordan J. Sosnowski, Samuel C. Hollifield, and Michael D. Iannacone. ROAD: The Real ORNL Automotive Dynamometer Controller Area Network Intrusion Detection Dataset. *arXiv preprint arXiv:2012.14600*, 2020.

[28] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. *IEEE Access*, 8:58194–58205, 2020.

Appendix A

Preprocessing and Feature Engineering Code

This appendix lists the principal Python code used to produce the stratified working subset, the preprocessing pipeline and the engineered features described in Chapter ??.

```python
import pandas as pd
import numpy as np

RANDOM_STATE = 42

df_normal = pd.read_csv("HCRL/normal_run_data.csv")
df_dos    = pd.read_csv("HCRL/DoS_dataset.csv")
df_fuzzy  = pd.read_csv("HCRL/Fuzzy_dataset.csv")
df_impers = pd.read_csv("HCRL/Impersonation_dataset.csv")

def sample_n(df, n=100_000):
    return df.sample(n=n, random_state=RANDOM_STATE).
       reset_index(drop=True)

df = pd.concat([
    sample_n(df_normal).assign(label=0),
    sample_n(df_dos).assign(label=1),
    sample_n(df_fuzzy).assign(label=2),
    sample_n(df_impers).assign(label=3),
], ignore_index=True)
df = df.sample(frac=1, random_state=RANDOM_STATE).reset_index(
   drop=True)
```

Listing A.1: Loading the HCRL Car-Hacking corpus and constructing the 400,000-message stratified working subset.

```python
def hex_to_int(x):
    try:
        return int(str(x), 16)
    except (ValueError, TypeError):
        return np.nan

DATA_COLS = [f"DATA{i}" for i in range(8)]

df["CANID"] = df["CANID"].apply(hex_to_int)
for col in DATA_COLS:
    df[col] = df[col].apply(hex_to_int)
```

```python

df[DATA_COLS] = df[DATA_COLS].fillna(0).astype(int)
df = df.dropna(subset=["CANID"]).reset_index(drop=True)
df["CANID"] = df["CANID"].astype(int)
```

Listing A.2: Field parsing, missing-value imputation, corrupt-row removal.

```python
DATA_COLS = [f"DATA{i}" for i in range(8)]

df["DataSum"]  = df[DATA_COLS].sum(axis=1)
df["DataMean"] = df[DATA_COLS].mean(axis=1)
df["DataStd"]  = df[DATA_COLS].std(axis=1)
df = df.sort_values("timestamp").reset_index(drop=True)
df["TimeDiff"] = df["timestamp"].diff().fillna(0)
```

Listing A.3: Engineered statistical features: DataSum, DataMean, DataStd, TimeDiff.

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

FEATURE_COLS = ["CANID", "DLC"] + DATA_COLS + [
    "DataSum", "DataMean", "DataStd", "TimeDiff"
]

X = df[FEATURE_COLS].values
y = df["label"].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, stratify=y, random_state=RANDOM_STATE
    )

scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test  = scaler.transform(X_test)
```

Listing A.4: Z-score normalisation and 75:25 stratified train/test split.

Appendix B

Model Training and Evaluation Code

This appendix lists the principal training and evaluation code for the LightGBM and 1D-CNN models described in Chapter 4.

```
import lightgbm as lgb
from sklearn.metrics import (
    accuracy_score, classification_report,
    cohen_kappa_score, roc_auc_score
)

clf = lgb.LGBMClassifier(
    objective="multiclass", num_class=4,
    n_estimators=500, learning_rate=0.05,
    max_depth=10, num_leaves=63,
    min_child_samples=20,
    bagging_fraction=0.9, feature_fraction=0.9,
    random_state=42, n_jobs=-1)

clf.fit(X_train, y_train,
        eval_set=[(X_test, y_test)],
        callbacks=[lgb.early_stopping(20)])

y_pred  = clf.predict(X_test)
y_proba = clf.predict_proba(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Kappa:   ", cohen_kappa_score(y_test, y_pred))
print("ROC-AUC: ", roc_auc_score(y_test, y_proba,
                                 multi_class="ovr", average="macro"))
print(classification_report(y_test, y_pred,
      target_names=["Normal", "DoS", "Fuzzy", "Impersonation"]))
```

Listing B.1: LightGBM training with the hyperparameters in Table 4.1.

```
import tensorflow as tf
from tensorflow.keras import layers, models, callbacks

def build_cnn(input_dim=15, n_classes=4):
    inp = layers.Input(shape=(input_dim, 1))
```

```python

    x = layers.Conv1D(64,  3, padding="same")(inp)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    x = layers.MaxPool1D(2)(x)
    x = layers.Dropout(0.3)(x)

    x = layers.Conv1D(128, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    x = layers.MaxPool1D(2)(x)
    x = layers.Dropout(0.3)(x)

    x = layers.Conv1D(64,  3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    x = layers.GlobalAveragePooling1D()(x)

    x = layers.Dense(128, activation="relu")(x)
    x = layers.Dropout(0.4)(x)
    x = layers.Dense(64,  activation="relu")(x)
    out = layers.Dense(n_classes, activation="softmax")(x)

    return models.Model(inp, out)

model = build_cnn()
model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-3),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"])

cb = [
    callbacks.EarlyStopping(patience=10, restore_best_weights=
       True),
    callbacks.ReduceLROnPlateau(factor=0.5, patience=5)]

X_train_cnn = X_train[..., np.newaxis]
X_test_cnn  = X_test[..., np.newaxis]

history = model.fit(
    X_train_cnn, y_train,
    validation_data=(X_test_cnn, y_test),
    epochs=50, batch_size=256,
    callbacks=cb, verbose=2)
```

Listing B.2: 1D-CNN architecture (matches Table 4.2) and training loop.

```python
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

def plot_cm(y_true, y_pred, title, out_path):
    classes = ["Normal", "DoS", "Fuzzy", "Imp."]
```

```python
    cm = confusion_matrix(y_true, y_pred)
    fig, ax = plt.subplots(figsize=(5, 4.2))
    im = ax.imshow(cm, cmap="Blues")
    ax.set_xticks(range(4)); ax.set_yticks(range(4))
    ax.set_xticklabels(classes); ax.set_yticklabels(classes)
    ax.set_xlabel("Predicted"); ax.set_ylabel("True")
    ax.set_title(title)
    for i in range(4):
        for j in range(4):
            ax.text(j, i, f"{cm[i, j]:,}",
                    ha="center", va="center",
                    color="white" if cm[i, j] > cm.max() / 2
                       else "black",
                    fontsize=9)
    fig.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
    fig.tight_layout()
    fig.savefig(out_path, dpi=200, bbox_inches="tight")
    plt.close(fig)
```

Listing B.3: Per-class confusion-matrix plotting helper used to produce Figures 5.1 and 5.2.