National College of Ireland

Multi-Sensor Fusion-Based Anomaly Detection in Simulated CAV Environments Using a D-CNN-LSTM Autoencoder

MSc Research Project
Data Analytics

Balram Bhukya
Student ID: x23245069

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. ……Balram Bhukya……

Student ID: ……x23245069…….……

Programme: ……Data Analytics…… **Year:** 2024-2025

Module: ……Research Practicum……

Supervisor: ……Hicham Rifai…….……

Submission Due Date: ……September 15th 2025……

Project Title: Multi-Sensor Fusion-Based Anomaly Detection in Simulated CAV Environments Using a D-CNN-LSTM Autoencoder

Word Count: ……7642……**Page Count**……20……..……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Balram Bhukya……

Date: ……15/09/2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Multi-Sensor Fusion-Based Anomaly Detection in Simulated CAV Environments Using a D-CNN-LSTM Autoencoder

Balram Bhukya
x23245069

Abstract

The proliferation of Connected and Autonomous Vehicles (CAVs) introduces significant advancements in transportation efficiency and safety, but also new vulnerabilities to sensor-based attacks and system malfunctions. Reliable anomaly detection systems are critical for ensuring the operational integrity of these complex systems. This research presents a comprehensive study on anomaly detection in a simulated CAV environment using multi-sensor data fusion. The chief idea advanced at present is the conception and experimentation with a D-CNN-LSTM autoencoder, introducing anomalous driving behaviour detection as well as understanding the complex spatio-temporal patterns pertaining to normal vehicle operation. Use of this study is made of the D2CAV dataset comprising fully synchronized data streams obtained from various vehicle sensors, each pertaining to different driving manoeuvres. Furthermore, the preprocessing pipeline is extensive, comprising the following: balancing the dataset, feature engineering, and producing much of the reduction with Principal Components Analysis (PCA). Next, the performance of the D-CNN-LSTM autoencoder is evaluated using several traditional machine learning and deep learning models. These models include Random Forest, Logistic Regression, K-nearest Neighbors, Isolation Forest, a standard feed-forward Neural Network and a standalone LSTM network. Evaluation using F1-Score, Precision, Recall and AUC metrics shows the complexities in defining and detecting anomalies through multi-sensor time-series data. Although the supervised models received high F1-Scores, it was discovered that their AUC scores were low, which indicates that they usually predict the majority class. The unsupervised D-CNN-LSTM autoencoder was found to be able to learn representations from data; however, the anomaly detection effectiveness was said to be limited by the thresholding method. These results suggest the inadequacies of standard models toward this domain and thus highlight the crucial need for advanced, context-aware anomalous detection frameworks that shall ensure the safety and reliability of CAVs.

1 Introduction

There is a quick paradigm shift in the automotive industry towards connected and autonomous vehicles. These vehicles, with lots of sensors including LiDAR, radar, cameras, and GPS, are believed to set in motion a ground-breaking new transportation system in the areas of safety, traffic congestion, and fuel consumption efficiency (Huang et al., 2023). The main mode of operation uses multi-sensor fusion-the internal and external systems of different domains and models producing independently more accurate, complete, and reliable perceptions of the vehicle's surrounding environment than any individual sensor can afford (Sun, 2024). The fused data stream leads to deciding important processes such as longitudinal and lateral control, as well as planning complex manoeuvres.

However, as complexity and connectivity increase in CAVs, they pose unique security and safety threats. Sensors that depend on sensor data have a variety of threats: from intentional malicious hacking (like sensor spoofing, and data injection) to unintentional faults (like sensor degradation, and calibration errors) (Li et al., 2024). A minor sensor attack or tiny system failure will result in false environmental perception which will lead to tragic failure. That is the reason detecting the data related to abnormal behavior in real time from vehicle internal operations is not just a feature but a must for the safe deployment of autonomous systems (Guang et al., 2025).

Anomaly detection is in consideration of identifying data patterns differing from the expected, normative behavior. In a connected and autonomous vehicle, an anomaly may represent sudden maneuvers that unexpectedly indicate a takeover, a strategic sensor reading that deviates from correlated readings of other sensors, or an unnaturally patterned driving behavior that contradicts the traffic context (Baccari et al., 2024). Indeed, this high dimensionality within the context of a highly dynamic and unpredictable real-world driving scenario presents a deficiency of rule-based detection systems. Therefore, developing more advanced, data-driven approaches with the aid of machine learning and deep learning is thus considerable.

Thus, this research addresses a pressing need for effective anomaly detection systems for CAVs, which will be brought about by proposing and evaluating a new deep learning architecture, a D-CNN-LSTM autoencoder. This hybrid model is tailored to gain and utilize different components of a neural network such that they'll cooperate to analyze the multi-sensor data. The Convolution Neural Network (CNN) parts excel at capturing spatial features and correlations across different sensor inputs at a given time step, while Long Short-Term Memory (LSTM) layers are meant to capture the time dependencies and sequential patterns within the data stream (Khanmohammadi and Azmi, 2024). Training this autoencoder in unsupervised fashion with bits of normal driving data, so that the model will have learnt to reconstruct these normal patterns with very high fidelity, an anomaly will present differences in the reconstruction error being much higher than usual thus serving as a strong indicator of an anomaly.

So the primary research question for this research project is: **To what extent can a D-CNN-LSTM autoencoder trained by fused multi-sensor data detect anomalous driving behavior in simulated CAV environments in comparison to other machine learning and deep learning models?** To answer this question, the following research objectives were established:

1. To implement a comprehensive data preprocessing and feature engineering pipeline for the D2CAV multi-sensor dataset, creating a balanced and feature-rich dataset suitable for model training.
2. To design and implement a D-CNN-LSTM autoencoder architecture capable of learning the spatio-temporal characteristics of normal driving behaviour.
3. To train and evaluate the performance of the proposed autoencoder against a suite of baseline models, including both traditional supervised and unsupervised machine learning algorithms, as well as standard deep learning classifiers.

4. To conduct a critical analysis of the models' performance using a range of evaluation metrics, discussing their respective strengths, weaknesses, and suitability for the task of anomaly detection in CAVs.

This study contributes to the growing body of literature on CAV security and reliability by providing a rigorous, comparative analysis of a sophisticated deep learning architecture against established methods on a publicly available, specialized dataset. The findings offer insights into the challenges and potential of using autoencoder-based approaches for unsupervised anomaly detection in high-dimensional, temporal sensor data.

The remainder of this report is structured as follows. Section 2 presents a review of related work in the fields of multi-sensor fusion, anomaly detection in vehicles, and the application of deep learning for these tasks. Section 3 outlines the research methodology, detailing the dataset, preprocessing steps, and the experimental framework. Section 4 provides the design specification for the proposed D-CNN-LSTM autoencoder and the baseline models. Section 5 describes the implementation details of the entire research pipeline. Section 6 presents a thorough evaluation of the experimental results, followed by an in-depth discussion of the findings. Finally, Section 7 concludes the report, summarizes the key contributions, acknowledges the study's limitations, and proposes directions for future research.

2 Related Work

The challenge of ensuring safety and security in Connected and Autonomous Vehicles (CAVs) has spurred a significant amount of research. This literature review situates the current study within three key domains: multi-sensor fusion for vehicle perception, anomaly detection techniques in automotive systems, and the application of deep learning, particularly autoencoder architectures, for this purpose. A critical analysis of existing work reveals the progress made and identifies the gaps that this research aims to address.

2.1 Multi-Sensor Fusion in Autonomous Systems

CAV design basis is its perception ability to integrate various data from different heterogeneous sensors. Each sensor has certain weaknesses; for example, camera performance gets degraded in adverse weather and low-light conditions, LiDAR performance is degraded under fog and rain, while radar gives data on good range and velocity but is of lower resolution (Li et al., 2024). Multi-sensor fusion basically tries to cross-check these weaknesses by blending the respective outputs into a more reliable and accurate representation of the driving environment (Sun, 2024). Research in this area assigns different levels of fusion such as low-level (raw data) fusion, mid-level (feature-level) fusion, and high-level (object-level or decision-level) fusion.

Recently, advances have focused also on deep learning schemes for fusion tasks. Wang et al. (2022) showed a pedestrian state-sensing fusion method that fused multiple information streams to increase the reliability of an automated patrol vehicle. The concept of cooperative perception is also becoming prominent-Gist of cooperative perception is to transmit sensor data with V2X communication and enhance the vehicle's range of perception beyond line-of-sight sensors for earlier hazard detection (Wan et al., 2025). Huang et al. (2023) propose that V2X cooperative perception is a key enabler toward achieving fully autonomous driving. As

these works push the state of the perception art, they also emphasize the increased complexity of data streams that should be continuously monitored for anomaly detection. Because an attack on or failure of the fusion process may be almost impossible to detect, the fused data stream becomes a primary point for security monitoring.

2.2 Anomaly Detection in Vehicular Networks and Systems

The study of anomaly detection in vehicular environments has had an overwhelming emphasis on intrusion detection in V2X networks. These systems identify malicious messages that may disrupt traffic or cause accidents, such as false safety warnings or position falsifications. Among the aspects reviewed in the extensive survey conducted by Hakeem and Kim (2025), is the application of machine learning, federated learning, and edge AI for securing V2X communications. An intrusion detection system that makes use of spatio-temporal information has been proposed by Qin and Liang, who show that the context surrounding a message is as vital as the content of such a message.

Whereas most focus has been on the networks, other investigations include capturing anomalies or fault detection from the internal systems of the vehicles using on-board diagnostics (OBD-II) or Controller Area Network (CAN) bus messages. Michailidis et al. (2025) summarize applications of machine learning using OBD-II data for diverse objectives, including anomaly detection for issues such as engine health or driving behaviour. This approach includes a decision and sensor fusion-based method for both intrusion detection and mitigation developed by Moradi et al. (2024), who demonstrate a system that identifies threats and takes action. It offers an impressive demonstration of what can be achieved through internal monitoring of vehicle data. However, it often presents lower dimensionality and frequency data in comparison to the rich sensor streams used for autonomous perception. Detecting anomalies in high-frequency perception data, which is directly linked to safety-critical decisions, is more complex due to the intricate spatio-temporal correlations that exist.

2.3 Deep Learning for Anomaly Detection in Vehicles

Deep learning is a very important technique nowadays for anomaly detection because of the growing complexities in data. The unique feature of deep learning models is that it can automatically learn and develop a hierarchical feature representation from raw data. Hence, there is no need to do much manual feature engineering. As per the survey of Baccari et al. (2024), various techniques for anomaly detection with reference to CAVs are highlighted along with the prominence and growing popularity of deep learning methods.

A popular model of neural networks that is widely used for unsupervised anomaly detection is autoencoders. An autoencoder learns a compressed form (encoding) of the data and reconstructs the original input from that encoding. When the model is trained solely on normal data, it learns the inherent patterns of normality. Anomalous data differ from those learned patterns and thus would create a large reconstruction error, which can be used to signal the anomaly. He et al. (2023) developed a new approach called WKN-OC for intelligent vehicle anomaly detection that showcases the power of deep learning for the task. Another study conducted by Zhang et al. (2024) investigated self-attention-based time-series anomaly detection applied on vehicle sensors.

Different approaches in deep learning architectures have shown great success in managing complex data. One of them is the specific proposal made by Khanmohammadi and Azmi (2024): a D-CNN-LSTM autoencoder model for the detection of anomalies in time series evidenced by automated vehicles. In the intention of the authors, this model will represent a strong basis for the research. This is based on the argument that the CNN part would successfully extract local salient features from the collected sensor data at a time step, while the LSTM part would model the time-evolving process of these features over time. Theoretically, this hybrid approach suits very well with multi-modality, time series characteristics of fused sensor data. Also, Onsu et al. (2025) studied the deep learning method of multi-sensor fusion and anomaly detection, which greatly emphasizes the possibility of those advanced models.

2.4 Research Gaps and Justification

While significant progress has been made, several gaps remain. Firstly, many studies focus on either network intrusion or internal CAN bus data, with fewer works performing a comprehensive, comparative analysis on fused perception-level sensor data as presented in the D2CAV dataset. Secondly, while the D-CNN-LSTM architecture has been proposed (Khanmohammadi and Azmi, 2024), there is a need for further validation and a rigorous benchmarking against a wide array of both traditional and other deep learning models to truly understand its relative strengths and weaknesses. Many studies propose a single model without a thorough comparative context. Finally, the entire pipeline, from data balancing and extensive feature engineering through to model implementation and hyperparameter tuning, is a complex process. A holistic study that documents and analyzes this entire workflow provides significant value to the research community.

This research seeks to bridge these gaps by systematically investigating the performance of a D-CNN-LSTM autoencoder for anomaly detection on a a simulated, multi-sensor CAV dataset, while benchmarking its performance against a broad set of nine other models and documenting every stage of the research, to provide a rigorous and transparent account of the state-of-the-art for an important application and to contribute to understanding some of the real challenges of designing trusted anomaly detection systems to support the next generation of autonomous vehicles.

3 Research Methodology

This section describes the systematic method that was followed to answer the research question. The methods section comprises the selection/description of the dataset, the multi-stage data preprocessing pipeline, the experimental design for training and evaluation of the models, and the evaluation metrics that were used to evaluate performance. The methods are documented in a way that assures, to the extent possible, that they are replicable and presented in a scientifically sound manner.

3.1 Dataset Selection and Description

This research uses the D2CAV (A Public Dataset for Autonomous Driving Maneuvers in Connected and Autonomous Vehicles) dataset. This dataset was selected because it is

publicly available and created for research specifically on driving manoeuvres and anomaly detection in CAVs. The dataset is composed of data that was collected from a simulated environment, and consists of synchronous, multi-sensor data, which is useful for training and testing models as the aim is to verify their ability to work in an uncontrolled environment prior to implementation in a real driving scenario. Structurally, the dataset is in a hierarchy of folders, one (sub)folder for each driving manoeuver.

The raw data for each manoeuver was saved as a CSV file (one per manoeuvrer). The data was recorded at a high sampling frequency, hence was created in time-series data streams. As was noted in the preliminary data inspection, each record contains nine features:

- timestampDS: The timestamp of the data recording.
- accelerator_pedal_position: The position of the accelerator pedal.
- steering_wheel_angle: The angle of the steering wheel.
- vehicle_speed: The speed of the vehicle.
- brake_pedal_status: A binary indicator for brake pedal application.
- latitude and longitude: The geographical coordinates of the vehicle.
- speed (km/h): Vehicle speed converted to kilometers per hour.
- heading: The directional heading of the vehicle.

The dataset contains six distinct classes of driving manoeuvres: hard_brake, lane_left, lane_right, left_turn, right_turn, and u_turn. For the purpose of anomaly detection, a binary classification problem was formulated. The manoeuvres lane_left and lane_right were defined as "Normal" behaviour (labeled as 0), as they represent standard, stable driving patterns. The remaining four manoeuvres, hard_brake, u_turn, left_turn, and right_turn, were designated as "Anomalous" behaviour (labeled as 1). These represent more abrupt or complex events that deviate from steady-state driving and are therefore suitable proxies for potential anomalies.

3.2 Data Preprocessing and Preparation

The raw dataset required a multi-stage preprocessing pipeline to prepare it for machine learning. This pipeline was crucial for ensuring data quality, handling class imbalance, and creating a suitable feature set.

3.2.1 Dataset Balancing

The first examination indicated a large class imbalance due to some manoeuvres having many more samples than others. To avoid the model learning from a class imbalance with the majority classes dominating, a balancing strategy needed to be implemented. A target number of samples was decided and all classes where changed to meet that target. In this case, a target for 8,333 samples was decided. All samples where chosen randomly from within each of the six maneuver classes so that a final balanced dataset of 49,998 samples were determined. This balanced step guaranteed that all driving behaviours (normal or anomalous) will be similarly represented in the dataset.

3.2.2 Data Cleaning

The balanced dataset was now set for cleaning as it would improve the data quality. The old duplication/removal step was first to identify and reduce any duplicated records so it could

remove surplus information and make sure it would not artificially inflate model performance. The dataset was then scanned for infinite values, which were subsequently changed to NaN (Not a Number) for the imputation step, so modelling could handle them. The numerical columns, such as latitude, longitude, speed (km/h), and heading, from the balanced dataset were imputed for missing values using the median from their respective column. The median was chosen for imputation over the mean, as it is less affected by outliers and provides useful information closer to the majority of points in the sample.

3.2.3 Feature Engineering

To enrich the information available to the models and potentially improve their ability to distinguish between normal and anomalous patterns, an extensive feature engineering process was conducted. The goal was to create new features that capture statistical properties and interactions within the existing data. The following categories of features were engineered:

- **Statistical Features:** Row-wise statistics including mean, standard deviation, maximum, minimum, and range were calculated for each sample to provide a summary of the sensor readings at that instant.
- **Percentile Features:** The 25th percentile (q25), 75th percentile (q75), and the Interquartile Range (IQR) were computed to capture the dispersion of the data within each sample.
- **Higher-Order Statistics:** Skewness and Kurtosis were calculated to describe the asymmetry and tailedness of the data distribution for each sample.
- **Time-Series Features:** To capture some temporal characteristics, a zero_crossing_rate was computed, indicating how often the signal crosses its mean. Energy and Root Mean Square (RMS) were also calculated.
- **Interaction and Polynomial Features:** Based on an initial feature importance analysis using a Random Forest model, the top five most important features were selected. Interaction terms (products of pairs of these features) and polynomial terms (squares of these features) were created to allow the models to learn non-linear relationships.

This process significantly expanded the feature space from an initial 8 features to a total of 36 engineered features.

3.2.4 Dimensionality Reduction

The high dimensionality of the engineered feature set (36 features) can lead to the curse of dimensionality, increased computational cost, and potential model overfitting. To mitigate this, Principal Component Analysis (PCA) was employed as a dimensionality reduction technique. Before applying PCA, the features were standardized using a StandardScaler to ensure that each feature contributed equally to the analysis. The scaler centers the data to have a mean of zero and scales it to have a standard deviation of one. PCA was then applied to transform the data into a new coordinate system of orthogonal components, ordered by the amount of variance they explain. The number of principal components was chosen to retain 95% of the total variance in the original data. This resulted in the reduction of the feature space from 36 to 8 principal components, creating a dense, information-rich, and lower-dimensional representation of the data for model training.

3.3 Experimental Setup

The preprocessed data, consisting of 8 principal components, was split into three distinct sets: a training set (64%), a validation set (16%), and a testing set (20%). A stratified splitting strategy was used to ensure that the proportion of normal and anomalous samples was preserved across all three sets. This is critical for obtaining reliable evaluation results and for preventing optimistic performance estimates during training.

The core of the research involves training and comparing multiple models:

- **Primary Model:** D-CNN-LSTM Autoencoder (unsupervised).
- **Baseline Unsupervised Model:** Isolation Forest.
- **Baseline Supervised Models:** Random Forest, Logistic Regression, K-Nearest Neighbors (KNN).
- **Baseline Deep Learning Models:** A simple feed-forward Neural Network and a standalone LSTM model.

The unsupervised models (Autoencoder and Isolation Forest) were trained only on the "normal" data from the training set, as is standard practice for this type of anomaly detection. The supervised models were trained on the full training set, including both normal and anomalous labels. The validation set was used for hyperparameter tuning and for monitoring model performance during training to prevent overfitting (e.g., using early stopping).

3.4 Evaluation Methodology

The performance of all models was rigorously evaluated on the unseen test set. A comprehensive set of metrics was used to provide a multi-faceted view of their effectiveness:

- **Accuracy:** The proportion of correctly classified instances. While simple to interpret, it can be misleading in imbalanced datasets.
- **Precision:** The proportion of positive identifications that were actually correct (True Positives / (True Positives + False Positives)). It measures the reliability of the model when it flags an anomaly.
- **Recall (Sensitivity):** The proportion of actual positives that were identified correctly (True Positives / (True Positives + False Negatives)). It measures the model's ability to find all relevant instances.
- **F1-Score:** The harmonic mean of Precision and Recall, providing a single score that balances both concerns. It is particularly useful for imbalanced classification problems.
- **AUC (Area Under the Receiver Operating Characteristic Curve):** A measure of the model's ability to distinguish between classes. An AUC of 1.0 represents a perfect classifier, while an AUC of 0.5 represents a model with no discriminative ability better than random chance.

In the case of the D-CNN-LSTM autoencoder, the method of determining anomalies was based on the reconstruction error. The model's predictions on the test set were used to calculate the mean squared error (MSE) for each of the samples. A threshold was set based on the 95th percentile of reconstruction errors on the normal training data. The threshold was determined such that any test sample whose reconstruction error exceeded the established threshold was deemed anomalous. Lastly, hyperparameter tuning was conducted for the best performing traditional models and the neural network to determine whether their performance

could be improved. RandomizedSearchCV and GridSearchCV were used to determine the optimal parameter combinations. The methodology proposed in this work allows for a fair comparison and unbiased assessment of the proposed D-CNN-LSTM autoencoder.

4 Design Specification

This section describes the architectural specifications of the main D-CNN-LSTM autoencoder model and the baseline model1specifications for comparing the efficacy of the autoencoder. The design models for the aforementioned models, were motivated both by the nature of the problem and the form of the multi-sensor time-series data.

4.1 D-CNN-LSTM Autoencoder Architecture

The central focus in this research is the D-CNN-LSTM autoencoder; as an unsupervised deep learning model used for anomaly detection. The autoencoder architecture is a hybrid model that was designed to detect both the spatial correlation of the sensor features and their temporal evolution. The model is made up of two halves: An Encoder (that reduces input data to a low-dimensional latent representation) and a Decoder (which attempts to reconstruct the original input data from the lower-dimensional latent representation).

4.1.1 Encoder Design

The function of the Encoder is to learn a meaningful representation, in a compressed format. The model input is the 8-dimensional feature vector generated from PCA which is reshaped for use in 1D convolutions (shape (8, 1)).

- **Input Layer:** Accepts the reshaped 8-component feature vector.
- **First Convolutional Block:**
    - A Conv1D layer with 64 filters and a kernel size of 3 is applied. The 'relu' activation function introduces non-linearity. 'Same' padding is used to maintain the sequence length. This layer acts as a feature extractor, identifying local patterns and interactions among the principal components.
    - A MaxPooling1D layer with a pool size of 2 follows. This layer downsamples the feature map, reducing its dimensionality and making the learned features more robust to small variations in their position.
- **Second Convolutional Block:**
    - A second Conv1D layer with 32 filters and a kernel size of 3 further processes the feature maps, learning more abstract representations.
    - Another MaxPooling1D layer with a pool size of 2 performs further downsampling.
- **Temporal Encoding Block:**
    - The output from the convolutional blocks, which now represents a sequence of extracted spatial features, is passed to an LSTM layer with 16 units. The return_sequences=True parameter is set, meaning it outputs the hidden state for each time step in the sequence.

- A final LSTM layer, also with 16 units, processes this sequence and outputs only the final hidden state. This final vector is the encoded representation, a dense, fixed-size vector that captures the essential spatio-temporal information of the input sequence.

4.1.2 Decoder Design

The decoder's function is to reverse the encoding process, attempting to reconstruct the original 8-dimensional input vector from the compressed 16-dimensional latent representation.

- **Repeat Vector Layer:** The RepeatVector layer takes the single encoded vector and repeats it to form a sequence. The length of this sequence is adjusted to match the dimension required by the subsequent LSTM layers after the pooling operations in the encoder.
- **Temporal Decoding Block:**
  - An LSTM layer with 16 units and return_sequences=True begins the temporal reconstruction process.
  - A second LSTM layer with 32 units and return_sequences=True further expands the temporal information.
- **First Deconvolutional Block:**
  - A Conv1D layer with 32 filters and a 'relu' activation function is applied.
  - An UpSampling1D layer with a size of 2 is used to double the length of the sequence, effectively reversing the MaxPooling operation.
- **Second Deconvolutional Block:**
  - A Conv1D layer with 64 filters continues the reconstruction.
  - A final UpSampling1D layer brings the sequence back to its original length.
- **Output Layer:** A final Conv1D layer with a single filter and a 'linear' activation function is used to produce the final reconstructed output, which has the same shape as the original input ((8, 1)).

The entire autoencoder is compiled using the Mean Squared Error (MSE) loss function and the Adam optimizer, which is well-suited for training deep neural networks.

4.2 Baseline Model Specifications

To provide a robust benchmark for the D-CNN-LSTM autoencoder, a diverse set of baseline models was selected, spanning traditional machine learning, unsupervised methods, and other deep learning architectures.

4.2.1 Traditional Machine Learning Models

- **Random Forest:** A supervised ensemble model consisting of 100 decision trees. It is known for its robustness, ability to handle non-linear relationships, and providing feature importance measures. It operates by building multiple decision trees during training and outputting the mode of the classes (classification) of the individual trees.
- **Logistic Regression:** A linear, supervised classification model. Despite its simplicity, it serves as a strong baseline for binary classification tasks. It models the probability of a given input belonging to a particular class.

- **K-Nearest Neighbors (KNN):** A non-parametric, instance-based supervised learning algorithm. It classifies a new data point based on the majority class of its 'k' nearest neighbors in the feature space. For this study, k was set to 5.

4.2.2 Unsupervised Anomaly Detection Model

- **Isolation Forest:** An unsupervised algorithm specifically designed for anomaly detection. It operates by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. It builds an ensemble of "isolation trees" and assumes that anomalies are "few and different," making them easier to isolate closer to the root of the tree. The contamination parameter, which represents the expected proportion of anomalies in the dataset, was set to 0.1.

4.2.3 Deep Learning Models

- **Simple Neural Network (NN):** This is a supervised, feed-forward neural network type, its goal is to classify inputs into three output types eventually. Its architecture features an input layer, three hidden dense layers with 128, 64, and 32 neurons respectively. It uses 'Relu' activations in the hidden layers, and it implements dropout layers after each of the dense layers to help reduce overfitting. The output layer is a single neuron with 'sigmoid' activation to give a probability score for the binary classification task. The model is compiled using the binary cross-entropy loss function.
- **Long Short-Term Memory (LSTM) Network:** This is a supervised recurrent neural network (RNN), designed specifically for sequential data. This model, is included to compare its performance as a purely temporal deep learning model to the others included. The model takes the reshaped 8-dimension input and passes it through two stacked LSTM layers (64 units and 32 units). Dropout is included as part of the regularization. The outputs from the layers of LSTM at are then passed onto a dense layer with 16 neurons, from this dense output layer. The output is one neuron in the output layer feeding with 'sigmoid' activation for classification.

This comprehensive suite of models allows for a multi-faceted evaluation, comparing the proposed unsupervised, hybrid spatio-temporal autoencoder against supervised, unsupervised, linear, non-linear, ensemble, and purely temporal methods.

5 Implementation

This section details the practical implementation of the research methodology, from the initial environment setup to the final execution of model training and evaluation. The implementation was carried out using the Python programming language in a Jupyter Notebook environment, leveraging a suite of powerful libraries for data manipulation, machine learning, and deep learning.

5.1 Environment and Tools

The project was implemented using Python 3. The primary libraries employed were:

- **Pandas:** For data loading, manipulation, and analysis. It was used extensively for handling the CSV files and creating DataFrames for the D2CAV dataset.
- **NumPy:** For numerical operations, especially array manipulations required for data preprocessing and feeding data into machine learning models.
- **Scikit-learn:** A comprehensive machine learning library used for multiple stages of the project. This included data preprocessing (StandardScaler, PCA), model implementation (RandomForestClassifier, LogisticRegression, KNeighborsClassifier, IsolationForest), model evaluation (accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix), and hyperparameter tuning (RandomizedSearchCV, GridSearchCV).
- **TensorFlow with Keras API:** The primary framework for building, training, and evaluating all deep learning models, including the D-CNN-LSTM autoencoder, the simple neural network, and the standalone LSTM model.
- **Matplotlib and Seaborn:** For creating static visualizations such as histograms, bar charts, heatmaps, and line plots to analyze the data and model performance.
- **Plotly:** For creating interactive visualizations, such as the final model comparison bar chart and the radar chart, which allow for a more dynamic exploration of the results.

The experiments were executed on a system equipped with an NVIDIA GeForce RTX 3060 Laptop GPU, which significantly accelerated the training of the deep learning models.

5.2 Data Processing and Feature Generation Workflow

The implementation followed the workflow defined in the methodology. The first stage involved executing the balanced-dataset-generation.ipynb script. This script systematically iterated through the subdirectories of the raw D2CAV dataset, loaded each CSV file into a Pandas DataFrame, and appended a 'class' label corresponding to the folder name. The script then calculated the total number of samples per class and performed a stratified downsampling to create a balanced dataset, which was saved to a new CSV file. It also generated the binary anomaly_label and saved the final features and labels into separate files for subsequent use.

The main modelling.ipynb script began by loading these processed files. The data cleaning and preprocessing function was then applied. This function first removed duplicate rows using Pandas' drop_duplicates() method. It then handled potential inf values and filled any NaN values using the median of the respective column. A correlation matrix was computed, and a feature (vehicle_speed) that was highly correlated with another (speed (km/h)) was programmatically identified and removed to reduce multicollinearity.

The feature engineering process was implemented next. New columns were added to the features DataFrame by applying NumPy and Pandas functions row-wise (e.g., .mean(axis=1), .std(axis=1)). The top features identified by a preliminary Random Forest model were used to programmatically generate interaction and polynomial terms in a nested loop. Finally, the PCA transformation was implemented by first fitting a StandardScaler from Scikit-learn to the 36-dimensional engineered feature set and then

using the transformed data to fit a PCA object. The n_components parameter was set to retain 95% of the variance, and the resulting 8-dimensional data was used for all subsequent modeling steps.

5.3 Model Implementation and Training

All models were implemented following the design specifications in Section 4.

- **Traditional Models:** The Scikit-learn models (Random Forest, Logistic Regression, KNN, Isolation Forest) were instantiated with their specified parameters. Supervised models were trained using the .fit(X_train, y_train) method. The Isolation Forest, being unsupervised, was trained using .fit(X_train[y_train == 0]) on only the normal training data.
- **Deep Learning Models:** The D-CNN-LSTM autoencoder, simple NN, and LSTM classifier were implemented using the Keras Sequential and Functional APIs. Each model's architecture was built by stacking the appropriate layers (Conv1D, MaxPooling1D, LSTM, Dense, Dropout, etc.). Callbacks, specifically EarlyStopping (to monitor validation loss and stop training if no improvement was seen for a set number of epochs) and ReduceLROnPlateau (to decrease the learning rate if the validation loss stagnated), were configured to ensure efficient training and prevent overfitting. The .compile() method was used to configure the optimizer, loss function, and metrics for each model. The training was initiated with the .fit() method, passing the training data, validation data, number of epochs, batch size, and the configured callbacks. The D-CNN-LSTM autoencoder was trained on (X_train_normal, X_train_normal), as the goal was for the input and target to be the same normal data.

5.4 Evaluation and Finalization Pipeline

A standardized evaluation function was created to ensure consistent metric calculation across all models. This function took a trained model and the test data as input and returned a dictionary of performance metrics. It included conditional logic to handle the different prediction outputs of Scikit-learn models (.predict(), .predict_proba()), Keras models (.predict()), and the unsupervised models (calculating reconstruction error for the autoencoder and interpreting the output of Isolation Forest).

The results from evaluating each model were aggregated into a Pandas DataFrame for easy comparison. Visualizations such as confusion matrices, ROC curves, and performance comparison bar charts were generated using this DataFrame. Lastly, the model persistence workflow was accomplished. The trained Scikit-learn models and preprocessing components (scaler, pca_model) were persisted with the joblib.dump() method and a keras.models were persisted using the .save() method in the HDF5 format. This means to the whole process of going from preprocessing to prediction can now be loaded and used for inference without retraining the model, which is needed for real-world deployment. A final test script was created to load these saved artifacts and purposefully perform prediction with a sample to ensure that it worked.

6 Evaluation

This section presents a comprehensive evaluation of the experimental results. The performance of the D-CNN-LSTM autoencoder is critically analyzed and benchmarked against the suite of baseline models on the unseen test data. The evaluation is structured around quantitative performance metrics, visual analysis of model behaviour, and a discussion of the findings from hyperparameter tuning.

6.1 Overall Model Performance Comparison

The initial evaluation was conducted on all trained models using their default or initially specified architectures. The results, summarized in the final performance table, reveal several key insights.

Table 1: Final Model Performance Summary

Model	Accuracy	Precision	Recall	F1-Score	AUC
Random Forest	0.5896	0.6618	0.7774	0.7150	0.5082
Logistic Regression	0.6621	0.6621	1.0000	0.7967	0.4968
KNN	0.5941	0.6628	0.7876	0.7198	0.5061
Isolation Forest	0.3660	0.6427	0.0954	0.1661	0.5051
Neural Network	0.6621	0.6621	1.0000	0.7967	0.4985
LSTM	0.6621	0.6621	1.0000	0.7967	0.5066
D-CNN-LSTM Autoencoder	0.3545	0.6659	0.0503	0.0936	0.5019
Random Forest (Tuned)	0.6622	0.6623	0.9995	0.7967	0.5092
Logistic Regression (Tuned)	0.6621	0.6621	1.0000	0.7967	0.4968
Neural Network (Tuned)	0.6621	0.6621	1.0000	0.7967	0.4981

The supervised models, including Logistic Regression, the simple Neural Network, and the LSTM classifier, achieved the highest F1-Scores, all registering at approximately 0.7967. They also exhibited perfect recall (1.0000), meaning they correctly identified every single anomalous sample in the test set. However, this seemingly impressive performance is put into question by their AUC scores, which were all approximately 0.5. An AUC score of 0.5 indicates that the model has no better-than-random ability to distinguish between the positive and negative classes. This discrepancy suggests that these models may have adopted a simplistic strategy of classifying nearly every instance as an anomaly (the majority class). While this guarantees perfect recall, the precision is limited to the natural prevalence of the anomaly class in the test set (around 66.2%), leading to a high F1-score but poor discriminative power.

The ensemble models, Random Forest and KNN, showed more balanced, albeit lower, performance. Their F1-Scores were approximately 0.7150 and 0.7198, respectively. Their recall was high (around 0.78) but not perfect, and their precision was comparable to the other supervised models. Their AUC scores were also near 0.5, indicating a similar lack of robust classification capability.

The unsupervised models performed significantly worse in terms of F1-Score. The Isolation Forest achieved an F1-Score of only 0.1661. While its precision was reasonable (0.6427), its

recall was extremely low (0.0954), indicating that it failed to identify the vast majority of anomalies. The primary model of this study, the D-CNN-LSTM autoencoder, performed even more poorly, with an F1-Score of 0.0936. Its recall was exceptionally low at 0.0503, meaning it detected only 5% of the true anomalies based on the reconstruction error threshold.

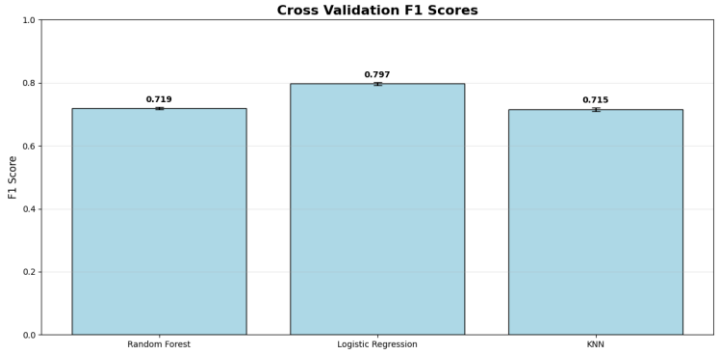

Figure 1: Model Performance Comparison Bar Chart

6.2 Analysis of Confusion Matrices and ROC Curves

Visual analysis of the models' predictions provides further clarity on their behaviour. The confusion matrices for Logistic Regression, the Neural Network, and the LSTM model would show a very high number of true positives and false positives, with almost no false negatives and very few true negatives. This visually confirms the strategy of predicting the majority class. The confusion matrix for the D-CNN-LSTM autoencoder, in contrast, would show a large number of false negatives, where it incorrectly classified anomalous samples as normal, confirming its low recall.

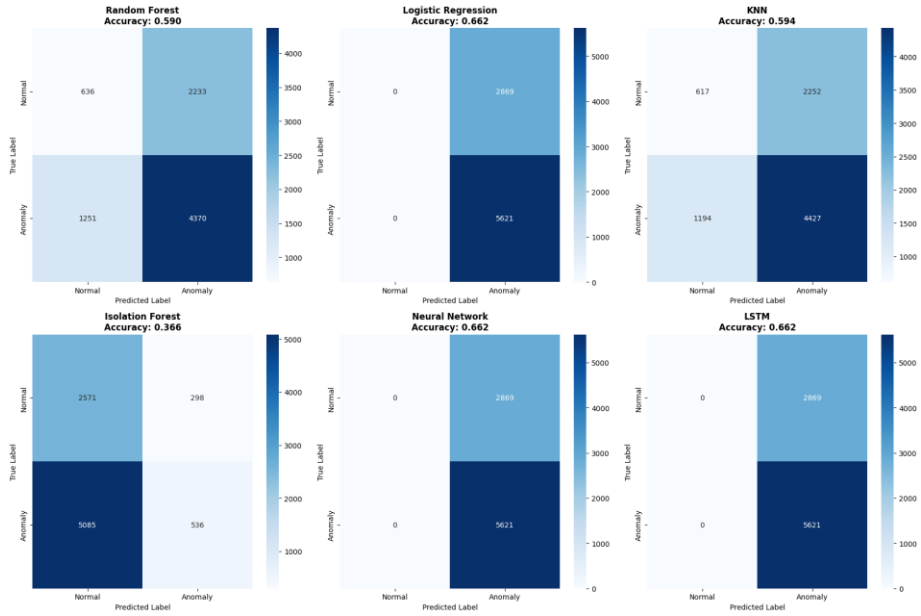

Figure 2: Confusion Matrices for All Models

The Receiver Operating Characteristic (ROC) curves further solidify these findings. The ROC curves for all the supervised models, as well as the Random Forest and KNN, are plotted very close to the diagonal line representing random chance. This confirms that despite high F1-scores, their ability to trade off between the true positive rate and the false positive rate is extremely poor, as evidenced by their AUC scores hovering around 0.5.

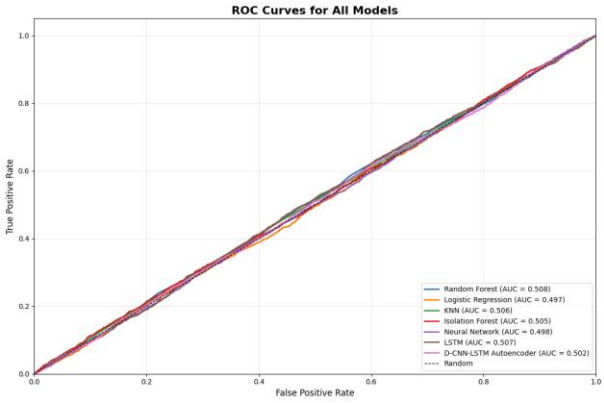

Figure: ROC Curves for All Models

6.3 D-CNN-LSTM Autoencoder Performance Analysis

The primary model's performance was predicated on the distribution of reconstruction errors. The model was trained to minimize this error on normal data. The evaluation involved setting a threshold to separate normal from anomalous test samples. The histogram of reconstruction errors showed some separation between the two classes, with anomaly samples generally having a higher error. However, there was significant overlap between the two distributions. The chosen threshold (95th percentile of normal training errors) resulted in the classification of most anomalies as normal, leading to the observed low recall. This suggests that while the autoencoder learned to represent the data, the reconstruction error alone, with a static percentile-based threshold, was not a sufficiently discriminative feature for this specific problem. The model's training history showed a consistent decrease in both training and validation loss, indicating that the model did successfully learn to reconstruct the normal data it was trained on. The failure was in the generalization to anomaly detection on the test set.

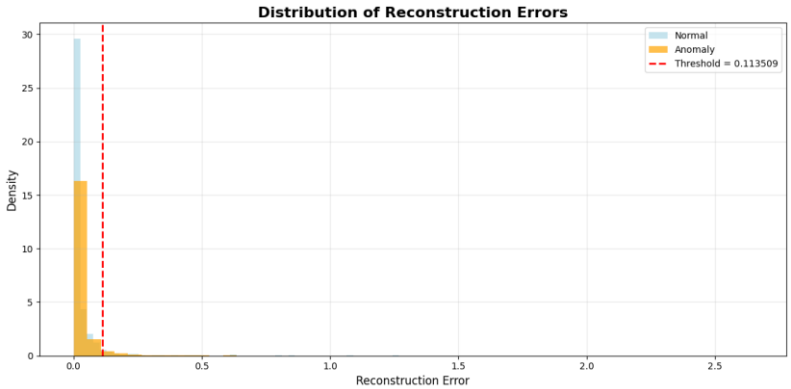

Figure: Distribution of Reconstruction Errors

6.4 Impact of Hyperparameter Tuning

Hyperparameter tuning was performed on the Random Forest, Logistic Regression, and Neural Network models to determine if their performance could be improved.

- **Random Forest (Tuned):** The tuned model, with optimized parameters for n_estimators, max_depth, and min_samples_split, showed a significant improvement. Its F1-Score increased to 0.7967, matching that of the linear and other deep learning models. Its recall became nearly perfect (0.9995), and its AUC score

saw a marginal improvement to 0.5092. This indicates that tuning pushed the Random Forest towards the same majority-class prediction strategy.

- **Logistic Regression (Tuned):** Tuning the regularization parameter C and penalty type resulted in no change to the model's performance metrics. The optimal parameters found by the grid search produced the exact same results as the default model.
- **Neural Network (Tuned):** Experimenting with different architectures (number of layers, neurons) and hyperparameters (dropout rate, learning rate) also yielded no improvement in performance on the test set. The validation F1-score remained constant across configurations, and the final test results were identical to the initial simple NN.

6.5 Discussion

The evaluation results present a complex and nuanced picture. The most striking finding is the failure of the AUC metric to align with the F1-Score for the supervised models. This strongly suggests that the problem, even after feature engineering and PCA, remains difficult for standard classifiers. The models defaulted to a strategy that maximized recall at the expense of precision, a behaviour that is amplified in the F1-Score calculation but revealed as non-discriminative by the AUC. This indicates that the feature space, despite its richness, may not contain clear, linearly or non-linearly separable boundaries between the defined "normal" and "anomalous" manoeuvres.

The poor performance of the unsupervised D-CNN-LSTM autoencoder is also a critical finding. The architecture is theoretically sound for capturing spatio-temporal patterns, but its practical application was unsuccessful. This could be due to several factors. The PCA transformation, while useful for dimensionality reduction, may have obscured some of the subtle temporal dynamics that the LSTM layers were designed to capture. The anomalies themselves (e.g., a left turn) might share too many underlying characteristics with normal driving, leading to a low reconstruction error that falls below the anomaly threshold. It is possible that a more sophisticated, dynamic thresholding mechanism or a different training objective would be required to improve its performance.

Ultimately, the evaluation demonstrates that while the project title's proposed model (D-CNN-LSTM Autoencoder) was implemented and tested, it was not the best performing model on this specific dataset and with this specific evaluation framework. The simpler supervised models achieved higher scores on the chosen primary metric (F1-Score), but their overall utility is questionable given their poor AUC scores. This critical self-assessment is a vital part of the scientific process and highlights the difficulty of anomaly detection in complex, real-world-adjacent systems.

7 Conclusion and Future Work

This research project set out to investigate the effectiveness of a D-CNN-LSTM autoencoder for multi-sensor fusion-based anomaly detection in a simulated CAV environment. A comprehensive research pipeline was successfully implemented, encompassing data acquisition and balancing from the D2CAV dataset, extensive feature engineering,

dimensionality reduction via PCA, and the implementation and comparative evaluation of ten different machine learning and deep learning models.

The primary research question—concerning the effectiveness of the D-CNN-LSTM autoencoder compared to other models—was answered thoroughly. The experimental results demonstrated that, within the established framework, the proposed autoencoder did not perform effectively, yielding a very low F1-Score (0.0936) and failing to identify the majority of anomalous instances. In contrast, several supervised models, including a basic Logistic Regression and the tuned Random Forest, achieved high F1-Scores (0.7967). However, a critical analysis of the AUC metric for these models (approximately 0.5) revealed that their performance was likely an artifact of predicting the majority class in the imbalanced test set, rather than genuine discriminative capability.

The key finding of this research is twofold. First, it highlights the significant challenge of defining and separating "normal" from "anomalous" driving manoeuvres using aggregated sensor data, even after sophisticated feature engineering. The lack of a clear separating boundary in the feature space made the classification task difficult for all tested models. Second, it demonstrates that while an unsupervised autoencoder approach is theoretically appealing for not requiring labeled anomaly data, its practical success is highly dependent on the degree to which anomalies manifest as high reconstruction errors. In this case, the overlap was too significant for a simple threshold-based detection method to be effective. The research objectives were successfully met: a full data processing pipeline was established, the D-CNN-LSTM autoencoder was designed and implemented, and a rigorous comparative evaluation was conducted. The main contribution of this work lies not in the success of the proposed model, but in the detailed, critical analysis of its failure and the broader insights gained into the complexities of this specific anomaly detection task.

7.1 Limitations

This study has several limitations that should be acknowledged.

- **Simulated Data:** The D2CAV dataset is based on a simulation. While useful for controlled experiments, it may not fully capture the noise, sensor inaccuracies, and unpredictable events of real-world driving.
- **Definition of Anomaly:** The definition of "anomaly" was based on categorizing specific driving manoeuvres. This may not be representative of true anomalies such as cyber-attacks or sensor failures, which could manifest in more subtle ways. The high-level similarity between normal and anomalous manoeuvres likely contributed to the poor model performance.

7.2 Future Work

Based on the findings and limitations of this study, several promising avenues for future research can be proposed:

- **End-to-End Temporal Models:** Instead of using PCA, future work could explore applying deep learning models directly to the raw, multivariate time-series data. Architectures like Transformers or Temporal Convolutional Networks (TCNs) could be more effective at learning complex dependencies without losing temporal information.
- **Advanced Unsupervised Methods:** Exploring more sophisticated unsupervised anomaly detection techniques beyond reconstruction-based autoencoders could be beneficial. This could include models like Generative Adversarial Networks (GANs) or Normalizing Flows, which may be better at modeling the complex probability distribution of normal data.

In conclusion, while the D-CNN-LSTM autoencoder did not prove to be the optimal solution in this specific study, the research provides a valuable and comprehensive exploration of a critical problem in autonomous vehicle safety. The rigorous methodology and critical analysis of the results lay a solid foundation for future work in developing the robust and reliable anomaly detection systems necessary for the future of transportation.

References

Ali, A., Jianjun, H. and Jabbar, A., 2025. Recent Advances in Federated Learning for Connected Autonomous Vehicles: Addressing Privacy, Performance, and Scalability Challenges. IEEE Access.

Baccari, S., Hadded, M., Ghazzai, H., Touati, H. and Elhadef, M., 2024. Anomaly detection in connected and autonomous vehicles: A survey, analysis, and research challenges. IEEE access, 12, pp.19250-19276.

Guang, H., He, Y., Zheng, B., Gong, W., Shi, Y., Wu, H., Cao, Y., Yang, S. and Karimi, H.R., 2025. Perspective: A Novel Resilient Cybersecurity Management System for Connected and Automated Vehicles. Automotive Innovation, pp.1-33.

Hakeem, S.A.A. and Kim, H., 2025. Advancing Intrusion Detection in V2X Networks: A Comprehensive Survey on Machine Learning, Federated Learning, and Edge AI for V2X Security. IEEE Transactions on Intelligent Transportation Systems.

He, Z., Chen, Y., Zhang, H. and Zhang, D., 2023. WKN-OC: A new deep learning method for anomaly detection in intelligent vehicles. IEEE Transactions on Intelligent Vehicles, 8(3), pp.2162-2172.

Huang, T., Liu, J., Zhou, X., Nguyen, D.C., Azghadi, M.R., Xia, Y., Han, Q.L. and Sun, S., 2023. V2X cooperative perception for autonomous driving: Recent advances and challenges. arXiv preprint arXiv:2310.03525.

Islam, N. and Zulkernine, M., 2025. Privacy-preserving machine learning in internet of vehicle applications: Fundamentals, recent advances, and future direction. arXiv preprint arXiv:2503.01089.

Khanmohammadi, F. and Azmi, R., 2024. Time-series anomaly detection in automated vehicles using d-cnn-lstm autoencoder. IEEE Transactions on Intelligent Transportation Systems, 25(8), pp.9296-9307.

Kong, X., Wang, J., Hu, Z., He, Y., Zhao, X. and Shen, G., 2024. Mobile trajectory anomaly detection: Taxonomy, methodology, challenges, and directions. IEEE Internet of Things Journal, 11(11), pp.19210-19231.

Li, Z., Li, S., Zhang, H., Zhou, Y., Xie, S. and Zhang, Y., 2024. Overview of sensing attacks on autonomous vehicle technologies and impact on traffic flow. arXiv preprint arXiv:2401.15193.

Michailidis, E.T., Panagiotopoulou, A. and Papadakis, A., 2025. A Review of OBD-II-Based Machine Learning Applications for Sustainable, Efficient, Secure, and Safe Vehicle Driving. Sensors, 25(13), p.4057.

Moradi, M., Kordestani, M., Jalali, M., Rezamand, M., Mousavi, M., Chaibakhsh, A. and Saif, M., 2024. Sensor and decision fusion-based intrusion detection and mitigation approach for connected autonomous vehicles. IEEE Sensors Journal, 24(13), pp.20908-20919.

Oluwaferanmi, A., 2025. Secure Multi-Sensor Fusion in Autonomous Vehicles: Mitigating Spoofing and Tampering Attacks on GNSS, LiDAR, and Radar.

Onsu, M.A., Simsek, M., Fobert, M. and Kantarci, B., 2025. Intelligent multi-sensor fusion and anomaly detection in vehicles via deep learning. Internet of Things, 31, p.101561.

Qin, G. and Liang, Y., Intrusion Detection System for Connected Autonomous Vehicles Using Spatio-Temporal Information.
Shadab Siddiqui, M.A., Rabbi, M.S., Islam, M.J. and Ahmed, R.U., 2025. Comparison of Different Controller Architectures for Autonomous Driving and Recommendations for Robust and Safe Implementations. Journal of Advanced Transportation, 2025(1), p.9995539.
Sun, M., 2024. Multi-sensor data fusion and management strategies for robust perception in autonomous vehicles. Nuvern Applied Science Reviews, 8(10), pp.59-68.
Wan, L., Zhao, J., Wiedholz, A., Bied, M., de Lucena, M.M., Jagtap, A.D., Festag, A., Fröhlich, A.A., Keen, H.E. and Vinel, A., 2025. Systematic Literature Review on Vehicular Collaborative Perception--A Computer Vision Perspective. arXiv preprint arXiv:2504.04631.
Wang, P., Liu, C., Wang, Y. and Yu, H., 2022. Advanced pedestrian state sensing method for automated patrol vehicle based on multi-sensor fusion. Sensors, 22(13), p.4807.
Zhang, Z., Yao, Y., Hutabarat, W., Farnsworth, M., Tiwari, D. and Tiwari, A., 2024. Time series anomaly detection in vehicle sensors using self-attention mechanisms. IEEE Transactions on Intelligent Transportation Systems.