ANOMALY DETECTION IN CAN DATA USING LSTM BASED MODELS

A Project Report

Submitted by

Ms. ARCHA JOSHY

REG NO : TKM21MEAI03

SEMESTER : IV

In partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY
IN
Mechanical Engineering (Artificial Intelligence)

Under the guidance of
Prof. SUMOD SUNDAR

Thangal Kunju Musaliar College of Engineering
Kerala

MAY 2023

DECLARATION

I, the undersigned hereby declare that the project report "ANOMALY DETECTION IN CAN DATA USING LSTM BASED MODELS", submitted for partial fulfillment of the requirements for the award of the degree of Master of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me, under the supervision of Prof. Sumod Sundar. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed as the basis for the award of any degree, diploma or similar title of any other University.

Place: Kollam
Date: ARCHA JOSHY

Thangal Kunju Musaliar College of Engineering
Centre for Artificial Intelligence

C E R T I F I C A T E

This is to certify that, this report titled ***ANOMALY DETECTION IN CAN DATA USING LSTM BASED MODELS*** is a bonafide record of the **Project** presented by **ARCHA JOSHY (TKM21MEAI03)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **M.Tech in Mechanical Engineering (Artificial Intelligence)** in **APJ Abdul Kalam Technological University** .

Project Guide	Project Coordinator	Head of the Department
Prof. Sumod Sundar Assistant Professor Centre for AI	Prof. Chinnu Jacob Assistant Professor Centre for AI	Dr. Imthias Ahamed T P Professor Centre for AI
Internal Examiner		External Examiner

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my seminar a memorable one. I thank him for steering me to the shore of fulfilment under his protective wings.

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of T.K.M College of Engineering for his immense support. I would like to thank **Dr. Imthias Ahamed**, Professor and Head of the Department, Centre for AI for his constant support and encouragement throughout the work.

With a profound sense of gratitude, I would like to express heartfelt thanks to my project guide **Prof. Sumod Sundar**, Assistant Professor, Centre for AI for his valuable suggestions and guidance. I would also like to express my profound gratitude to the project coordinator **Prof. Chinnu Jacob**, Assistant Professor, Centre for AI, for her cooperation, and immense encouragement. I would like to thank **Mr. Sreejith Pai P S**, Project Manager, Tata Elxsi, and **Ms. Chandralekha**, Project Mentor, Tata Elxsi, for their expert guidance, and cooperation. I also extend my thanks to the entire faculty and staff of the Centre for AI, T.K.M College of Engineering, who have encouraged me throughout this work.

I also express my thanks to my loving parents, brother and friends, for their support and encouragement in the successful completion of this work.

ARCHA JOSHY

Abstract

The evolution of technologies in the automobile industry has increased the complexity of the in-vehicle electrical network. This poses a new challenge for testers to do troubleshooting work in massive log files of CAN (Controller Area Network) data obtained from cars. The Controller Area Network (CAN) bus is one of the in-vehicle communication networks that enable each ECU to communicate with all other ECUs in a system. Data from these components are required to be monitored to detect any anomalies or faults in the system during testing. Manual testing for detecting anomalies has limitations such as - high cost and high time consumption. To overcome these limitations AI-based anomaly detection techniques can be employed. This project deals with an anomaly detection mechanism in which various anomalies (manually introduced) in the captured CAN data logs are identified using an LSTM autoencoder. The detected anomalies can further be classified using BiLSTM classifier. The normal CAN data logs from the HIL simulator are used for training the LSTM AE model. Optuna framework is employed for hyperparameter tuning. The model which is trained using the optimal hyperparameters is tested using anomaly-introduced data and finally performing dynamic thresholding on the reconstruction error obtained from the original and the predicted values of the model, anomalies can be identified. The anomaly-injected CAN logs are used to train the BiLSTM model, which can classify the anomaly type from the anomaly-detected data. Experimental results have shown that the LSTM autoencoder model obtained an accuracy of 99%, precision of 81.22%, recall of 98.2%, and F1-Score of 88%, and the BiLSTM classifier can classify the anomalies with an accuracy of 99%.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The number of sensors found in vehicles has dramatically increased over time as a result of enhancements in the field of autonomous vehicles, according to [1]. The CAN (Controller Area Network) bus, in modern vehicles, is a type of communication network within a vehicle that facilitates inter-ECU communication. This network allows for smooth communication between all ECUs present within the system.

In order to facilitate communication between micro-controllers and devices, the CAN bus was developed for usage in automotive and industrial settings. The Controller Area Network bus is one of the in-vehicle communication networks that enable each ECU (Electronic Control Unit) to communicate with all other ECUs in a system [2].

The CAN bus is utilised in modern automobiles to interconnect multiple ECUs that control various operations, including but not limited to engine management, transmission control, airbag deployment, and anti-lock braking systems. These ECUs establish communication with one another through the CAN bus, with the aim of ensuring seamless integration and coordination of diverse systems.

CAN bus is a type of serial communication bus that employs a pair of twisted wires to facilitate the transmission and reception of data. The bus functions through a protocol that is based on messages, wherein every message comprises an identifier that enables distinct ECUs to identify the message and respond accordingly. The implementation of a standardized communication protocol facilitates the easy integration of new systems and components into the network.

Anomaly detection algorithms can be used on CAN data to find patterns that vary from the system's intended behavior. These patterns can include unexpected spikes or decreases in signal values, missing or delayed communications, and other abnormalities that may indicate a system breakdown.

Detecting anomalies in CAN data allows testers to take preventive measures to diagnose and repair defects before they cause major damage or system failure. Anomaly detection can also be used as a technique for predictive maintenance, allowing prospective flaws to be identified before they become significant concerns. The identification of faults in these ECU

signals on the CAN bus can frequently be accomplished through the analysis of CAN data logs. There may be changes in these ECU signals when any faults or anomalies occur, which should be detected during testing for the proper maintenance of the system or vehicle. It is essential that these systems must have built-in redundancy, fault tolerance, and health awareness as the variety of sensors that these applications need to operate increases.

There is an increasing concern that the complexity of vehicle software may increase the likelihood of malfunctions. Human lives can be severely compromised by software failure or fault in a safety-critical automotive system, such as a brake system.

The test drive is the most common real-world testing technique that may be used to satisfy the functional safety criteria of ISO 26262. As a result, before mass-producing a car, manufacturers build prototypes and test them on public roads, capturing the systems' reactions as a sequence of measures over time.

This method of testing, however, has considerable limitations, including a high cost per test kilometre, a substantial time commitment, and a high danger to the test driver. There are two basic options by which a vehicle can be examined in abnormal situations or to identify any kind of failure or malfunction. This can be done in one of two ways:

- In the real vehicle
- In a full vehicle simulator

The full vehicle simulator is a massive setup that includes all of the ECUs that will be present on the actual vehicle. These are integrated into the HILS (Hardware-in-the-loop Simulator) in a lab environment, and the lab environment itself is connected to all of them, like the car connected to all of its systems.

The HIL simulator is designed to imitate the behaviour of a real ECU by responding to input signals in the same manner as a real ECU. In an HIL simulator, the real control unit or Electronic Control Unit (ECU) is substituted with a hardware device that simulates the ECU's behaviour. The HIL simulator is linked to a software-based simulation model of the system under test, which creates the HIL simulator's input signals. The HIL simulator then responds to these input signals by creating the appropriate output signals that would be produced by the real ECU.

Because of the HIL platform's critical role in delivering safe, adaptable, and reliable real-time validation of ECUs, digital test drives may be done electronically in the lab, standardising mileage and greatly decreasing expenses [3]. Yet, the varied sensors, embedded computers, networks, and actuators that comprise the ASS's complicated architecture generate a large amount of data [4].

Usually, a DTC is run in order to find the faults in the ECU signals during the time of testing. DTC stands for Diagnostic Trouble Code, and it is a code created by the ECU to indicate a fault or malfunction in the system. DTC checking is a two-step process in which initially an automated process, using any software tools will be run to detect the faults or anomalies. In some cases, this automated process may fail and for those cases, manual

checking needs to be done, which is time-taking and requires more manual labour.

While considering the case of anomaly detection in fleet vehicles, where a massive amount of logged data of multiple vehicles are being logged, a DTC capture at the time of failure may not happen i.e. CAN data logs are captured continuously from these vehicles but DTC capture may not occur for the faults in real-time. The DTC will be logged only if the corresponding snapshots are present at the time of failure. A snapshot will contain the details of the fault that occurred. For fleet data, most of the time there will not be an associated snapshot at the time of failure and the testers need to run the logs captured and find out where the fault occurred.

As a result, manufacturers can no longer rely on the time-consuming, labour-intensive, approaches used in the past to discover and categorise errors in test drive recordings, raising the danger of catastrophic accidents. AI-based anomaly detection algorithms can be utilised to overcome these constraints.

1.1 Objectives

The objectives of the anomaly detection technique are :

1. To design develop a deep learning-based methodology that analyses, detects and classifies functional failures or anomalies in the system behaviour from CAN bus data logs.
2. The technique should be primarily used for the testing of Vehicles / Full Vehicle Simulators.

1.2 Organisation of the report

The remainder of this report is organized into the following chapters: Chapter 2 reviews different approaches for anomaly detection in vehicular and sensor data. A background study is discussed in Chapter 3. A detailed explanation of the proposed methodology is described in Chapter 4. Chapter 5 deals with the implementation steps that are done. The experimental results are discussed in Chapter 6. Conclusions of this study are presented in Chapter 7.

Chapter 2

Literature Review

Faults or anomalies that develop in-vehicle ECU signals have the potential to spread unchecked throughout control systems if undetected. As automotive applications develop increasingly autonomous, automatic failure detection and health monitoring algorithms will be required. This section deals with the anomaly detection techniques that are being used nowadays. Current fault diagnostic algorithms are ineffective for complex systems such as autonomous vehicles, where faults or anomalies in several sensors are very likely.

An M-CNN (Modified Convolutional Neural Network) is proposed in [5] for the detection of anomalies in autonomous vehicles. The Safety Pilot Model Deployment (SPMD) dataset is used to train this M-CNN for anomaly detection in which anomalies are randomly injected into speed, vertical acceleration, and GPS signals. There is a significant challenge for CNNs when dealing with time series data that requires high contextual learning.

Accordingly, a new intelligent fault detection and classification (FDC) model is proposed by [6] using a hybrid CNN-LSTM to be used throughout the system-integrated testing phase of the V-cycle development process. Here faulty data is generated using a real-time fault injection framework based on HIL without changing the original system model. Eight different types of anomalies in engine RPM, vehicle speed, and intake manifold variables are detected using the hybrid model. While dealing with multiple signals, a hybrid model often encounters difficulties in real-time implementation due to an increase in the inference time, which may lead to anomalies going undetected.

The identification of faults in an electromechanical conversion chain for traditional or autonomous EVs (Electric Vehicles) with reduced inference time is demonstrated in [7]. Data such as currents and voltages retrieved from the EVs are used to detect any short circuit or open circuit faults by employing an LSTM which can accurately learn the time dependencies within the data. Further improvements can be made to the detection task by considering the distribution of various signals using an encoder-decoder structure.

As a result, [8] proposes a method for detecting brake cylinder (BC) pressure relief time BOU anomalies on metro vehicles by employing a one-class LSTM autoencoder, using BC pressure data extracted from the BOU dataset. Detecting anomalies in BC pressure signals in the braking system of metro trains' Brake Operating Units (BOU) is critical for train

reliability and safety. However, anomalies may occur in more than one signal. In such cases, an efficient methodology is needed for detecting anomalies in multiple signals.

Therefore, in [9], a novel fault detection, isolation, and identification architecture for multi-fault in multi-sensor systems with a low computing overhead for real-time implementation is developed. Support Vector Machine approaches are utilized to detect and diagnose faults in sensors used in the control systems of autonomous vehicles. Artificially generated drift, hard-over, erratic, spike, and stuck faults are injected into the driving torque, brake pressure, and steering wheel angle signals.

Multi-sensor signal training can lead to improved results in anomaly detection tasks as mentioned in [10]. CNN with STFT (Short Time Fourier Transform) is used to detect vibrational faults in the bumper and wheels. Upon increasing the number of features from a single feature to multiple features, the detection accuracy is found to increase. A hierarchical model for the detection and classification of anomalies provides better results in real-time.

A hierarchical model combining LSTM AE and CNN is proposed in [11] detects and classifies anomalies in traffic signals as abrupt, intermittent, and gradual. The loss function used for the identification of anomalies is MSE (Mean Square Error) and MAPE (Mean Absolute Percentage Error), which may lead to an increase in the false positive rate.

SVDD (Support Vector Data Description) can be used for the efficient detection of anomalies, as mentioned in [12], where the LSTM network transforms variable-length sequences of data to a particular length and the anomalies in the data are detected using SVDD by utilizing five features. However, when the number of features or variables where the anomaly is to be detected increases, it becomes a challenge to understand the complex underlying distribution of the data.

Anomaly detection methods employing a thresholding technique are discussed in [13]. Thresholding is applied to the prediction error obtained from LSTM RNN, and for reduction of false positives pruning is utilized, where the false positives are removed by excluding the data points that show less absolute deviation from the previous data point, from the detected anomalies.

Nevertheless, for pruning, the parameters need to be found using the trial and error method and may not be able to generalize for more rail transit devices. Whereas [14] deals with a method of using anomaly score for the identification of anomalies in the electric power steering data. An LSTM AE is used for the reconstruction of the data and an anomaly score based on the reconstruction error is used for the detection of anomalies. Regardless, since only the distribution of normal data is considered, the classification of distinct types of anomalies is difficult.

Concisely, considering time series data where the time dependency of the distribution of the data is to be learned, the choice of an LSTM-AE model for the detection of anomalies with a suitable classifier can provide more accurate results.

Chapter 3

Background Study

This chapter gives an overview of the background information related to anomaly detection and classification of anomalies in CAN data.

3.1 Overview of CAN bus

CAN bus is one of the communication protocols that are used mainly in automotive and other industrial applications. The main aim of CAN bus is to allow for the transmission of data between different Electronic Control Units. ECU can be identified as an electronic device which can control a specific function in an automobile, including the engine, transmission, or braking system. Multiple ECUs in the CAN bus communicate with each other over a single pair of wires [15].

ECU signals in the CAN are specific electrical signals which are being sent and received by the various individual ECUs in the Controller Area Network. These signals might be analog or digital and are utilised to operate various operations of the vehicle. Some examples of ECU signals include:

- Fuel injector: This signal is used in the engine to control the timing and duration of fuel injection.
- Throttle position: This signal is utilised to determine the location of the accelerator pedal and appropriately adjust the engine throttle.
- Wheel speed sensor: This signal is used to calculate the speed of each wheel on the vehicle, which is necessary for operations like traction control and anti-lock braking, etc.

Various communication protocols, such as the CAN bus, LIN bus, and FlexRay, are used to send signals between ECUs. These communication protocols enable ECUs to communicate in a standardised manner while also ensuring that the signals are accurately received by the receiving ECU [16].

The LIN bus is developed for networks with low speed, low cost, and low complexity. It operates on a single cable, which saves money but limits data rates [17]. LIN is frequently

used for non-critical activities such as window control, climate management, and so on.

The CAN bus is developed for high-speed, real-time communication and is compatible with more complicated networks. It uses a differential two-wire system and is hence it is used in more sensitive systems such as engine management and safety-critical systems.

CAN bus signals are encoded using a bit-wise arbitration method, in which the most significant bit (MSB) takes precedence over the least significant bit (LSB). This means that if two ECUs try to transmit data at the same time, the one with the higher priority message will prevail.

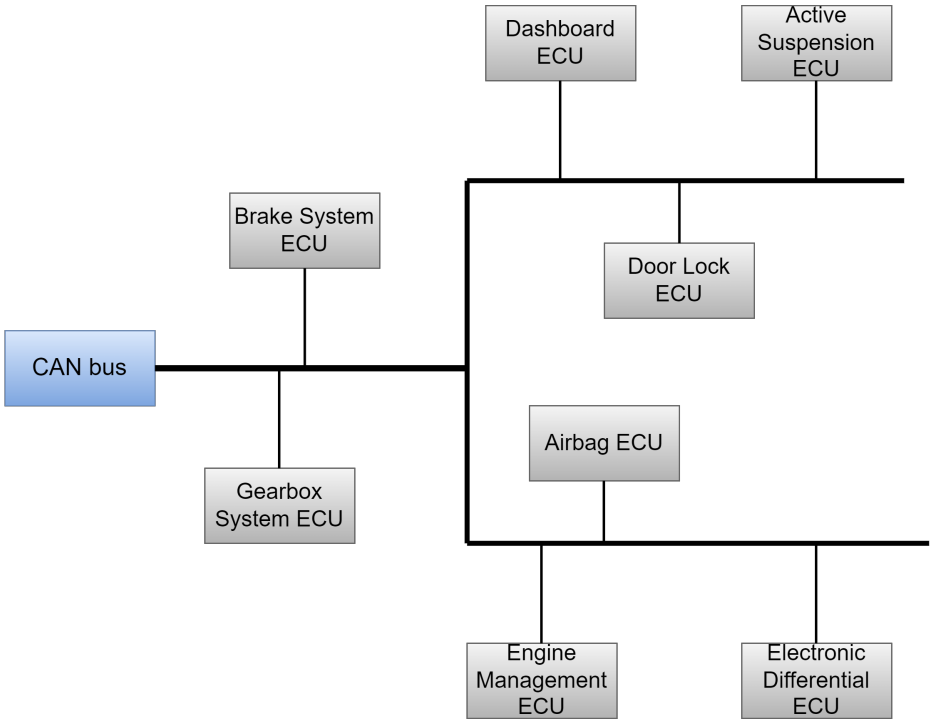

Figure 3.1: Example of a Controller Area Network connection

Figure 3.1 shows an overview of the various ECUs that can be connected to a CAN bus. The CAN bus serially transfers data through twisted wires, with each bit transferred consecutively over the same channel. This serial communication technology enables high-speed data transmission as well as dependable communication between several ECUs linked to the same bus. The CAN bus signals are terminated with resistors at each end of the bus to eliminate signal reflections and assure reliable connection.

FlexRay is designed for extremely fast, deterministic communication in complicated and safety-critical systems. It uses a dual-channel, time-triggered communication technology

that allows for more deterministic and fault-tolerant communication [18]. FlexRay can be used in applications that require real-time communication, although it is often more expensive and difficult to deploy than CAN bus.

Anomalies can occur in the ECU signal data of CAN bus occur due to different factors such as including electrical interference, communication errors hardware failures etc. A spike anomaly can be induced by a number of sources, including electromagnetic interference (EMI), voltage transients, or power surges. Spikes can distort data transferred via the CAN bus, causing system faults.

When a message sent over the CAN bus is not received by the intended recipient, a packet loss anomaly can occur. The reasons behind the cause packet loss, may include network congestion, hardware problems, and software defects. Packet loss can cause system delays and mistakes, as well as data corruption.

When a signal on the CAN bus remains at a specific logic level and does not change state, this is referred to as a stuck-at anomaly. A stuck-at anomaly mainly occurs in counters and can be produced by a hardware failure, such as a short or open circuit, software flaw that prevents the signal from changing state.

Because of the growing level of complexity of modern vehicles and the increasing use of electronic control units (ECUs) that generate and transmit massive amounts of data over the CAN network, anomaly detection approaches for CAN bus data are becoming increasingly relevant. It is critical for assuring the safety, dependability, and security of modern cars that communicate through the CAN bus.

3.2 Deep Learning Techniques

The various deep-learning techniques such as LSTM Autoencoders and BiLSTMs used in this study are detailed as follows.

3.2.1 LSTM Autoencoders

An LSTM (Long Short-Term Memory) autoencoder is a form of recurrent neural network (RNN) that encodes and decodes data using LSTM units [19].

Autoencoders are neural networks that can be used for data compression, feature extraction, and anomaly detection by learning a compressed representation of input data.

The Autoencoder (AE) is composed of three layers that are linked together sequentially: the input layer, the hidden layer, and the output layer, also called the reconstruction layer. The AE is commonly used for generating data or as a generative model. To train the AE, there are two phases: encoding and decoding. In the encoding phase, the input data is transformed into the hidden layer representation, which creates a latent vector of the input data. In contrast, the decoding phase involves reconstructing the input data from the representation created in the encoding phase.

LSTMs are a form of RNN that can handle long-term dependencies in sequential data better, making them ideal for modelling time series and sequential data.

The input sequence is initially encoded into a latent space representation using an LSTM encoder network in an LSTM autoencoder. An LSTM decoder network then decodes this latent space representation to reconstruct the original input sequence. The decoder network learns to recreate the output sequence from the encoder network's representation of the data.

A generalised representation of an LSTM Autoencoder is given in Figure 3.2. The LSTM-AE architecture is composed up of two LSTM layers, one of which acts as an encoder and the other as a decoder. The encoder layer encodes an input sequence (x_1,x_2,...,x_n) into a learnt representation vector. The decoder layer then takes that vector as input and attempts to recreate the input sequence as $(\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_n)$.

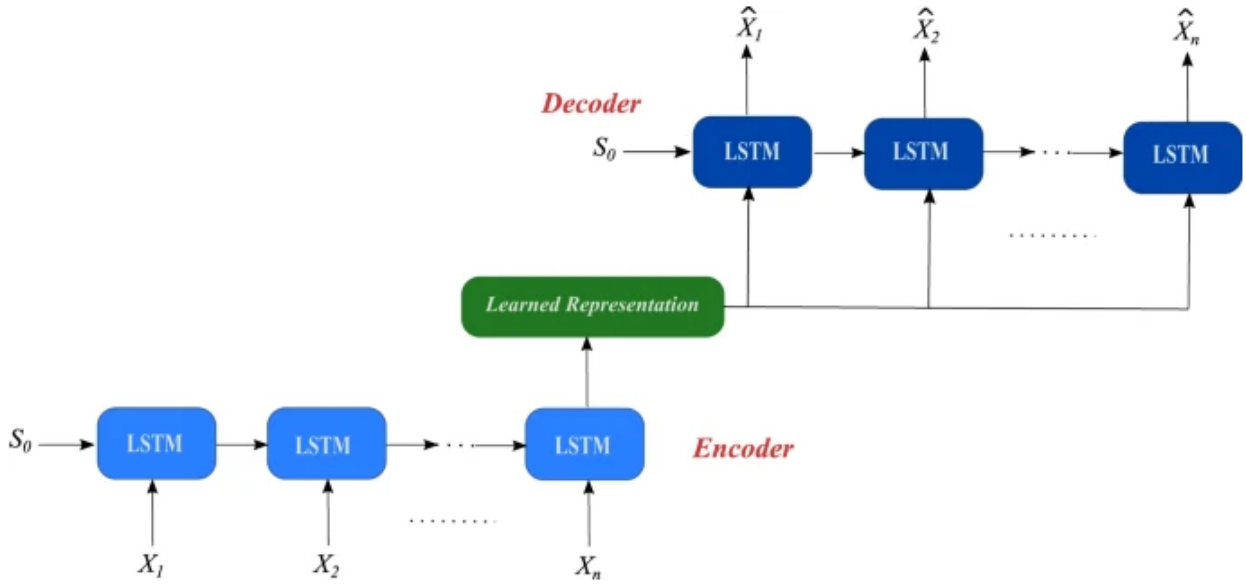

Figure 3.2: Generalized representation of LSTM Autoencoder [20]

The ability to handle variable-length input sequences is one advantage of LSTMs in autoencoders. Another benefit of utilising LSTM Autoencoders is their ability to detect temporal dependencies in input data. This is beneficial for time-series data, where the order of the input sequence is critical. LSTMs can learn to recall significant information from previous time steps and use it to forecast values of future time steps.

LSTM autoencoders have been utilised in anomaly detection for identifying unusual patterns in time-series data [21]. The autoencoder is trained on normal data and learns to accurately reconstruct it. When given anomalous data, the autoencoder generates a reconstruction error that is greater than the error generated by normal data, thus detecting anomalies efficiently.

3.2.2 BiLSTM Classifiers

Bidirectional Long Short-Term Memory (BiLSTM) is one of the different types of RNN (Recurrent Neural Networks) that can process sequential data such as time series data.

The BiLSTM classifier can analyse data sequences and categorise them into several classes. It performs by processing the input sequence in both forward and backward directions, allowing it to record both the sequence's past and future context. This increases its effectiveness in modelling complex temporal dependencies in data.

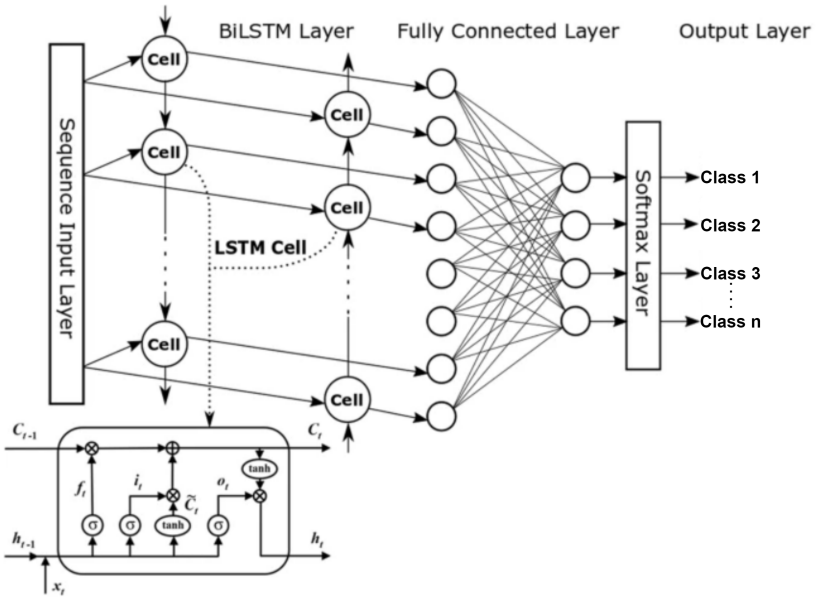

Figure 3.3: Generalized representation of BiLSTM Classifier [22]

The generalized representation of the BiLSTM classifier shown in Figure 3.3 is comprised of two LSTM layers, one that processes the input sequence forward and the other that processes it backwards. Each LSTM layer's output is concatenated, and the resulting sequence is given to fully connected layers, which generates the final output.

Chapter 4

Methodology

This chapter describes the detailed methodology of anomaly detection and classification techniques.

4.1 Proposed Model

The proposed methodology employs the use of an LSTM AE for the detection of anomalies and a BiLSTM classifier for classifying the anomalies.

An **LSTM Autoencoder** is an implementation of an autoencoder for sequence data using an Encoder-Decoder LSTM architecture. Since LSTMs are good at capturing long-term dependencies between various elements of a sequence, they are capable of processing sequential data more effectively.

When using **BiLSTMs**, past and future dependencies can be captured as the input data sequence is processed simultaneously in both forward and backward directions. Figure 4.1 shows the basic block diagram of the methodology to be used for anomaly detection and classification.

A detailed block digram of the different steps involved in the methodology is shown in Figure 4.1 The raw CAN logs logged from the HIL simulator are used for the experimentation. This data can be collected using specialised hardware interfaces or other data collection technologies. Once the data is collected, it is analysed and decoded to determine the various signals and messages being transmitted. This is done through various CAN analysing tools and softwares.

This data contain normal ECU signals, without any anomalies introduced. Pre-processing of the data is done by removing the null values and doing feature scaling using StandardScalar. This data (normal) is given to the LSTM Autoencoder model during the training phase.

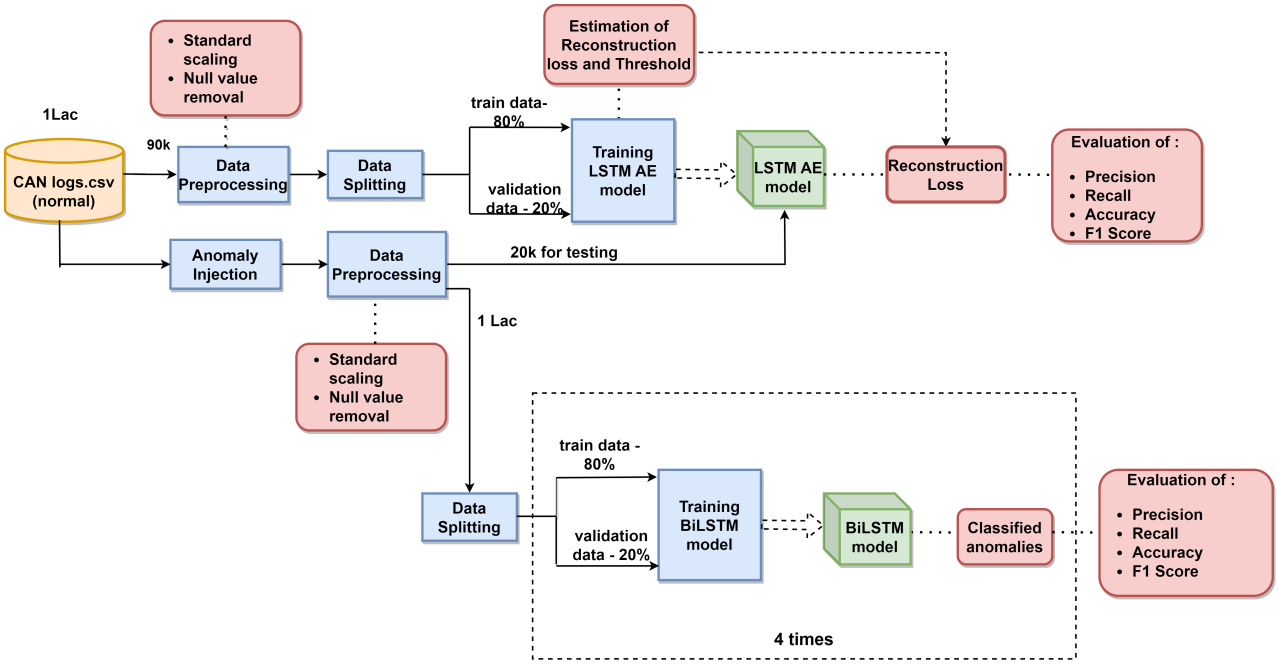

Figure 4.1: Proposed Model

The hyperparameter values which are obtained using the Optuna hyperparameter tuning framework are used to train the LSTM AE model. This LSTM Autoencoder model then reconstructs the normal data. The reconstruction loss of an LSTM Autoencoder is calculated using a loss function as the difference between the original input and the reconstructed output. Here, the reconstruction loss is obtained by taking the absolute difference between the predicted values and the ground truth values.

A threshold is calculated using the mean and standard deviation of the reconstruction error distribution. Anomalous data is obtained by manually introducing anomalies to the normal CAN data logs at random locations. An algorithm is developed such that anomalies can be randomly introduced to the normal CAN data logs. The different types of anomalies considered are:

- **Stuck-at Anomalies**: The signal value remains at a fixed value for a short period of time
- **Spike Anomalies:** The signal value is significantly above the true value for a single point. The density of spike faults within the signal can increase over time.
- **Packet loss Anomalies**: The signal value is null i.e. the data is lost at different time instances.

The anomalous data thus generated is given to the model during the testing phase and the anomalous reconstruction loss is obtained.

Smoothing of the reconstruction loss is done using EWMA (Exponential Weighted Moving Average) technique such that false positives and negatives can be reduced to an extent.

The threshold found using the reconstruction loss of the normal data is applied to the smoothed anomalous reconstruction loss. The data points that are above this threshold are considered anomalous points and those that are below are considered normal data.

For the further classification of the anomalies, a BiLSTM classifier can be employed which is trained on the anomaly-injected data. This classifier can classify the class to which the detected anomaly is present.

4.2 Work Flow

The basic workflow of the process in a real-time test scenario is given in below Figure 4.2.

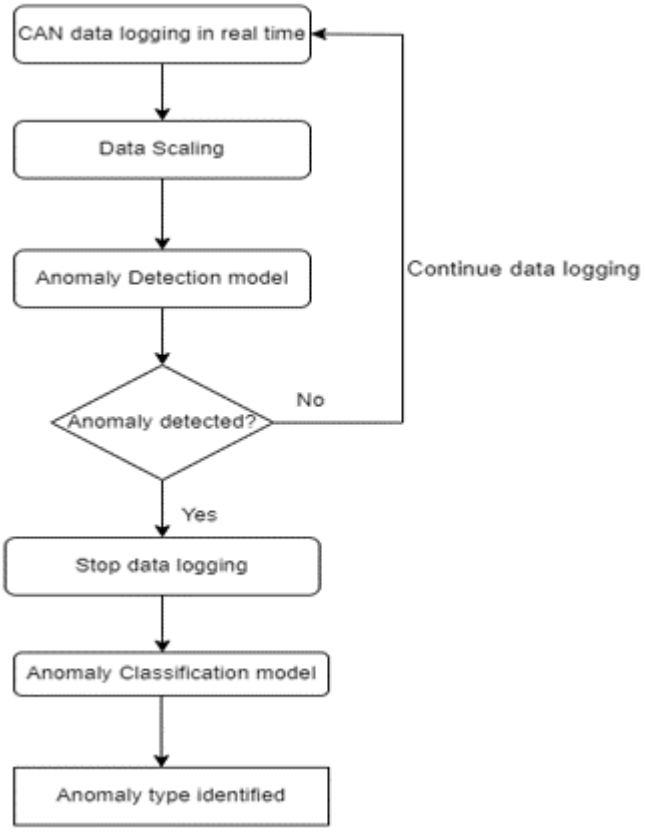

Figure 4.2: Flow Diagram - Test Scenario

The overview of the work flow in test scenario involves the use of real-time CAN data. Below mentioned are the steps of the work flow.

- **Step 1:**
  CAN data logging in real time is done.
- **Step 2:**
  Scaling of the logged data is done for each time step using Standard scalar.
- **Step 3:**
  The scaled data is then fed into the anomaly detection model, based on the LSTM Autoencoder architecture.
- **Step 4:**
  The model uses the data to compute the reconstruction error and determines whether the data is anomalous or normal.
- **Step 5:**
  If the data is identified as normal, the CAN data logging process continues without any interruption.
- **Step 6:**
  If the data is identified as anomalous, the logging process is stopped.
- **Step 7:**
  Anomaly identified data is passed to the anomaly classification model.
- **Step 8:**
  The BiLSTM classifier identifies the specific type of anomaly detected.

There would be no significant lag in the inference as the LSTM takes very less time for anomaly detection. The LSTM Autoencoder separates the whole workflow from the resource-heavy BiLSTM model and only invokes the BiLSTM model upon the detection of an anomaly. This helps in preventing any anomalies from flowing into the data processing or logging stages.

Chapter 5

Implementation

5.1 Dataset Used

Logged CAN bus data collected from a full vehicle simulator is for experimentation. The decoded CAN data in CSV format is unlabeled and is recorded for 1100 seconds with a sampling rate of 10ms. This CAN bus data contains signal values from various ECUs in the CAN network. There is a total of 426 signals and 113000 rows in the dataset. Some of the signals in the dataset include:

- BrakePressure_HS[Bar]
- BBrakePressureComp_HS
- LateralAcceleration_HS[m/s/s]
- MeanBrakePressures2_HS[Bar]
- UB_CorneringLampRequest_HS, etc

The recorded log data is normal data with no anomalies introduced.

5.1.1 Data Preparation and Pre-processing

Exploratory Data Analysis

By analyzing the different signals in the CAN data logs and calculating the variations of the signals at each time-step, these signals can be mainly grouped into three as shown in below Figures 5.1,5.2, 5.3. For calculating the variations in a signal, each signal for a particular timestep is taken. Then for each timestep, it is analyzed whether the value of this signal is changing from the previous timestamp. If it changes then it is counted as a variation. Similarly, the variations for the entire is signal are identified and thus they can be categorized into three:

- High Variational Signals: Signals with variation greater than 40%.

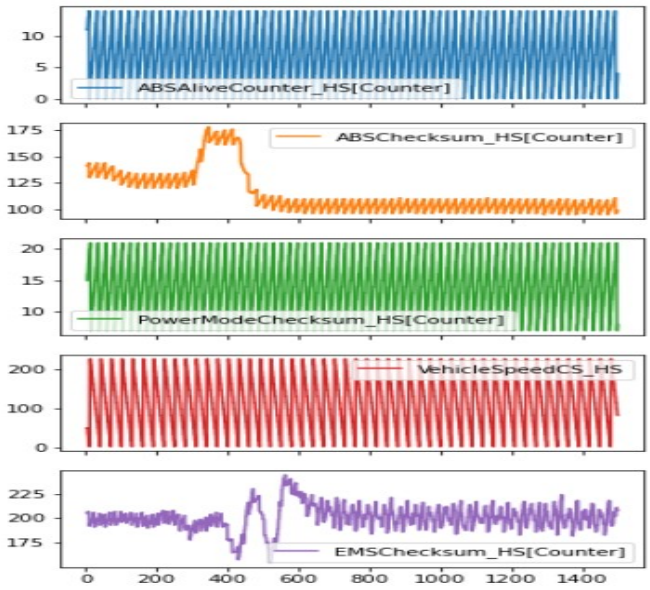

Figure 5.1: Sample of high variational signals

- Mid Variational Signals: Signals with variation between 10% and 40%.

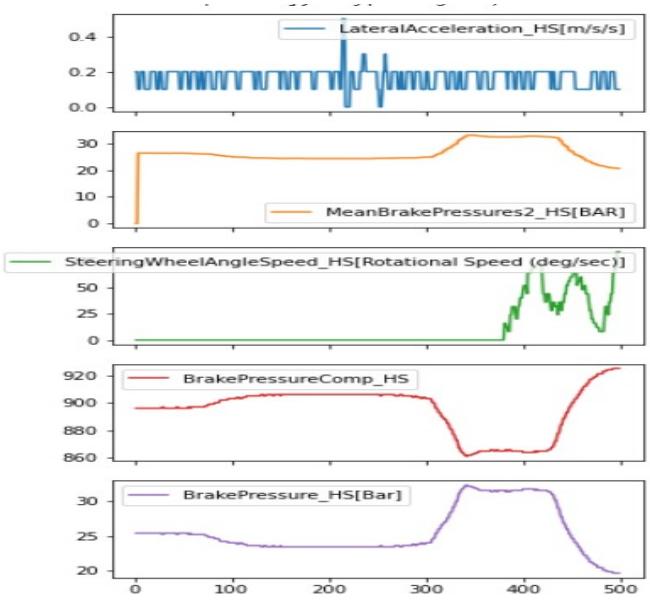

Figure 5.2: Sample of mid-variational signals

- Low Variational Signals: Signals with variation less than 10%.

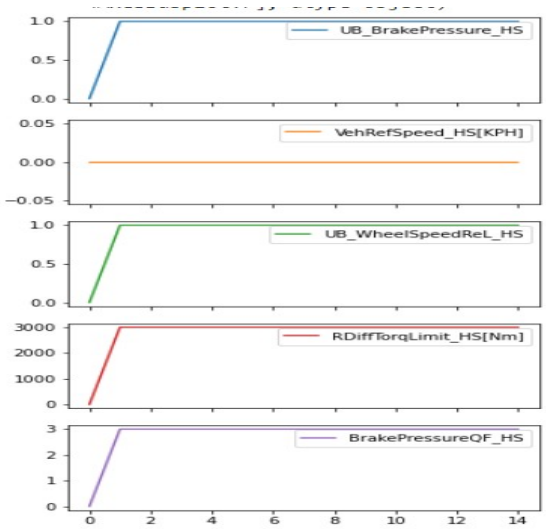

Figure 5.3: Sample of low variational signals

Pre-processing

Data Pre-porcessing can be done to increase the data quality i.e. the raw data may be containing missing values or inconsistencies needs to be removed to improve the performance of the model.

In the data pre-processing stage, empty features from the dataset are dropped. The entire data is scaled using StandardScalar technique. It is mainly used to transform numerical data into a standardized format, which makes it easier to compare and analyze the data.

With StandardScaler, the data will be transformed so that its distribution has a mean of 0 and a standard deviation of 1. Standard scaling is done by subtracting the mean from each data point and then dividing the results by the standard deviation. This is done in a feature-wise manner for multivariate data, i.e. that independent scaling for each column of the dataset is done.

5.1.2 Hyperparameter Tuning

Hyperparameter tuning optimises the performance of a machine learning/deep learning model by choosing the appropriate hyperparameters.

Optuna is a free and open-source hyperparameter optimisation (HPO) framework for determining the optimal values of hyperparameters for machine learning or deep learning models. The learning rate, number of units, optimizer, number of hidden layers, etc are all hyperparameters that needs to be specified before training the model.

The selection of hyperparameters has a significant impact on the performance of a model, and determining the optimal values of hyperparameters is sometimes a difficult and time-consuming task.

Optuna hyperparameter tuning that automates the process of hyperparameter tuning by searching for the ideal values of hyperparameters in a time-efficient manner [23].

It involves defining a search space for each hyperparameter, then developing an objective function which measures the model performance with different hyperparameters and finally executing the optimization using parallel processing.

To efficiently search the space of possible hyperparameters, Optuna employs a technique known as Bayesian optimisation. Bayesian optimisation works by constructing a probabilistic model of the objective function (i.e., the metric being optimised, such as accuracy or loss) and then using this model to recommend new sets of hyperparameters to evaluate. The objective function is scored utilising the entire set of accessible data or a subset of it.

Optuna also contains a visualisation tool that allows the user to view the optimisation process in real time and includes parallel processing capabilities to speed up the search.

The end result is a set of fine-tuned hyperparameters for the model's training phase. The hyperparameters that were tuned here are- Time Steps, Learning rate, Optimizer, No. Of layers, No. Of units, Activation function and Loss.

Hyperparameter tuning is done for 100 trials, and the loss function was considered as the objective function, such that the optimal hyperparameter values will be given for the trial with the minimum loss. As shown in Figure 5.4 below, the best value of the loss was obtained for trial 94. The blue dots represent the loss value at each trial and the red line passes through the minimum value of loss for the entire 100 trials.

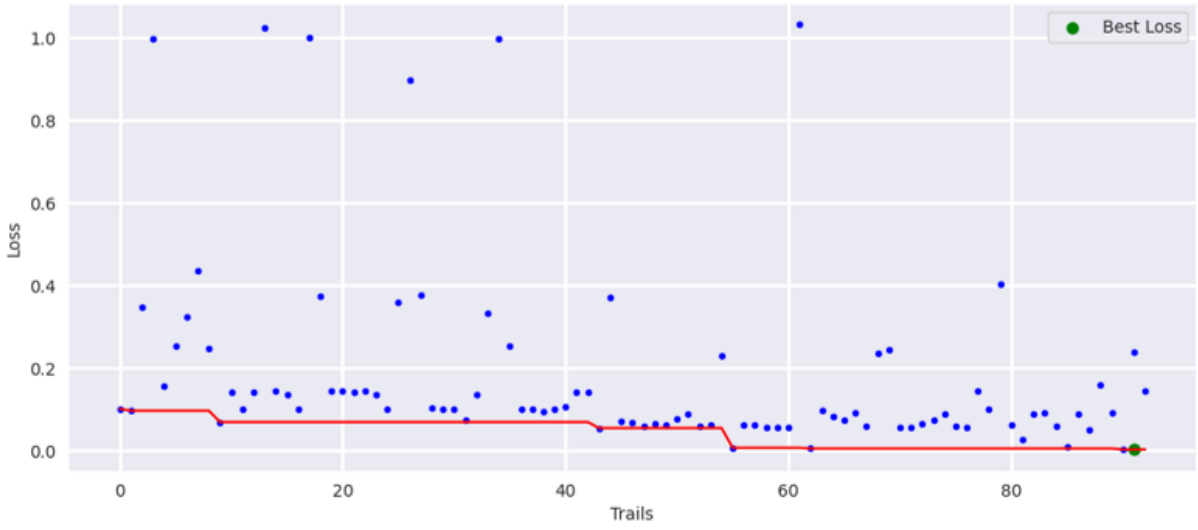

Figure 5.4: Optuna Score Plot

The optimal values obtained for the hyperparameters at trial 94 are shown in Table 5.1.

Table 5.1: Hyperparameter values obtained from Optuna

Hyperparameter	Value
Time Steps	10
Learning rate	0.0000519
No. Of layers	1
No. Of units	4096
Activation function	tanh
Loss	Huber

5.1.3 Creating Anomalous Data

For testing the model, manually introduced anomalous data is considered. The different anomalies that are taken into consideration are:

- Anomaly 1: **Spike Anomaly**

  The signal value is significantly above the true value for a very short period of time. Spike anomaly usually occurs in the range of 0.2s in a signal.

- Anomaly 2: **Stuck-at Anomaly** The signal value remains at a fixed value for a short period of time. Stuck-at anomaly usually occurs in a range of 3s in Alive counter or checksum signal.

- Anomaly 3: **Packet loss Anomaly** The signal value is lost or null at different time instances. Packet loss anomalies usually occur in a range of 2s in a signal.

Injection of Anomalies

An algorithm is proposed and implemented for the injection of anomalies are randomly selected points. The algorithm is as given in Algorithm 1.

Algorithm 1: Algorithm for anomaly injection

Data: *Normal ECU signals* S_{yx} *with* x *signals each of total size* y
Result: *Anomalous data*
Get number of spike(S), *packet loss*(P), *and stuck at*(K) *anomalies*;
$anomaly\_points \leftarrow 0$;
$S\_count \leftarrow 0$;
$P\_count \leftarrow 0$;
$K\_count \leftarrow 0$;
while $S\_count \leq S$ **do**
  $R \leftarrow random\ value\ in\ range(0, y - 20)$;
  $i \leftarrow random\ signal\ from\ x$;
  $A \leftarrow value\ greater\ than\ max(S[i])$;
  $S[j][i] \leftarrow A\ for\ j\ in\ range(R, R + 20)$;
  $Add\ values\ of\ range(R, R + 20)\ in\ anomaly\_points$;
  $S\_count \leftarrow S\_count + 20$;
end
while $P\_count \leq P$ **do**
  $R \leftarrow random\ value\ in\ range(0, y - 200)$;
  $i \leftarrow random\ signal\ from\ x\ ;\ S[j][i] \leftarrow -1\ for\ j\ in\ range(R, R + 200)$;
  $Add\ values\ of\ range(R, R + 20)\ in\ anomaly\_points$;
  $P\_count \leftarrow P\_count + 200$;
end
$V \leftarrow indices\ of\ Alive\ -\ counter\ and\ Ckecksum\ signals$;
while $K_count \leq K$ **do**
  $R \leftarrow random\ value\ in\ range(0, y - 300)$;
  $i \leftarrow random\ signal\ from\ V$;
  $A \leftarrow value\ greater\ than\ S[R][i])$;
  $S[j][i] \leftarrow A\ for\ j\ in\ range(R, R + 300)$;
  $Add\ values\ of\ range(R, R + 300)\ in\ anomaly\_points$;
  $K\_count \leftarrow K\_count + 300$;
end

The algorithm begins with taking the normal signals in which anomalies needs to be introduced. The points were anomalies are introduced are contained in anomaly_points.

The spike anomalies are introduced for a range of 0-20 data points, i.e for 0.2s. For introducing spike anomalies, the maximum value of the signal is considered. Anomaly values are introduced at each point in the randomly selected 20 consecutive data points such that the value of the anomaly is greater that the maximum value of the signal.

Packet loss anomalies are introduced for a range of 0-200 data points, i.e for 2s. For intro-

ducing packet loss anomalies, the values for randomly selected 200 consecutive data points are set to -1.

Stuck-at anomalies are introduced to Alive-counter and Checksum signals for a range of 0-300 data points, i.e for 3s. For introducing stuck-at anomalies, the values for randomly selected 200 consecutive data points are set to the previous value of the signal at the first data point in the range. These anomalies are injected to randomly selected points in the normal dataset.

After injecting the anomalies, this anomaly data is given for testing the LSTM AE model. By applying error smoothing and thresholding strategies on the reconstruction loss of the anomalous data obtained during testing, anomalies can be detected.

5.1.4 Error Smoothing and Thresholding

Smoothing the reconstruction error obtained while predicting the anomalous data by the LSTM AE can lead to an improvement in the results by reducing the false positive and false negative rates [24].

EWMA (Exponential Weighted Moving Average) can be used to smooth the error [25]. Exponential Weighted Moving Average (EWMA) is a technique that calculates a weighted average of past data, with the weight decreasing exponentially as the data become older.

For each timestamp, the smoothed error is calculated considering the smoothed error of the previous timestamp and a smoothing factor alpha. The value of alpha which usually ranges from 0 to 1, determines how important the current observation is for the calculation of EWMA. An increased value of alpha reduces the impact of older data and vice-versa.

The threshold is calculated by considering the mean and standard deviation of the normal reconstruction loss as given in the Equation 5.1.

$$Threshold = mean_{error} + 6 * std_{error} \tag{5.1}$$

where $mean_{error}$ represents the mean of the normal reconstruction loss and std_{error} represents the standard deviation of the normal reconstruction loss.

This threshold is applied on the smoothed error and the points above the threshold are considered anomalous points. An algorithm for error smoothing using EWMA and thresholding is given in Algorithm 2.

Algorithm 2: Algorithm for error smoothing and thresholding

Data: *Reconstruction loss of normal and anomalous data*
Result: *Anomaly points detected above threshold*
$error\_r \leftarrow$ *normal reconstruction loss*;
$\alpha \leftarrow 0.25$;
Get smoothed error at time t using EWMA;
$S[t] = \alpha * R[t] + (1 - \alpha) * S[t-1]$;
if $S[t] \geq R[t]$ **then**
  $S[t] \leftarrow R[t]$;
end
else
  $S[t] \leftarrow S[t]$;
end
if $S[t] > threshold$ **then**
  anomaly present at time t
end
else
  no anomaly present at time t
end

Initially, a threshold is set based on the reconstruction loss obtained from the prediction of normal data, by taking the mean and standard deviation of the data distribution.

Then, the reconstruction loss after predicting the anomalous data is smoothed by employing EWMA at each timestamp. The value of the smoothing factor is set as 0.25 obtained using the trial and error method.

After obtaining the smoothed error, a pruning method [26] is employed such that at each timestamp, both the smoothed error and the original reconstruction error is compared and the lesser of the two values is taken. Anomalies are detected by applying the threshold on this error, such that if the point is above the threshold, an anomaly is detected.

5.2 Classification of Anomalies

Anomalies can be detected using the LSTM AE model by employing thresholding on the reconstruction error obtained for anomalous data. For the further classification of the anomaly type, a BiLSTM classifier is utilized. This model is trained using the anomalous data obtained using the anomaly injection algorithm given in Algorithm 1.

The BiLSTM model is trained 4 times, each time anomaly being introduced at randomly selected points. This ensures that the model encounters anomalies at almost every point in the dataset while training.

Chapter 6

Results and Discussion

LSTM AE model is trained using the optimal values of hyperparameters obtained from Optuna. The architecture of the trained LSTM AE model is shown in Figure 6.1. The encoder-decoder structure with a dense layer produces a reconstruction of the input data (426 features with 10 time steps) fed to the LSTM AE model.

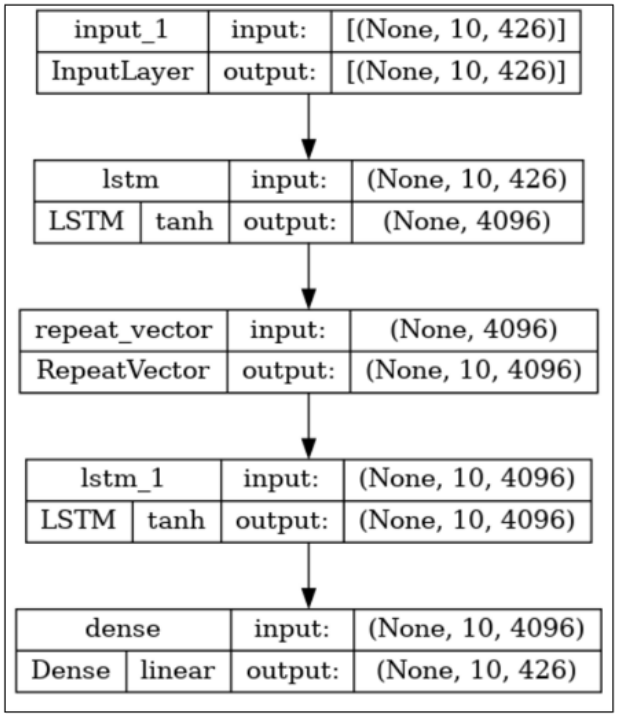

Figure 6.1: LSTM AE model Architecture

The temporal dependencies in the input sequence are modelled using the LSTM. While the Autoencoder can accurately learn the representation of the sequence and hence reconstruct the input sequence more efficiently. The normal CAN data logs are given for training the LSTM AE model. During the training of the model, early stopping was employed with a patience of 50. The training and validation curve of the model is shown in Figure 6.2.

Thus the training and validation of the model stopped at the 98th epoch since there was no further improvement in the loss. The training and validation losses obtained are 0.002 and 0.003 respectively.

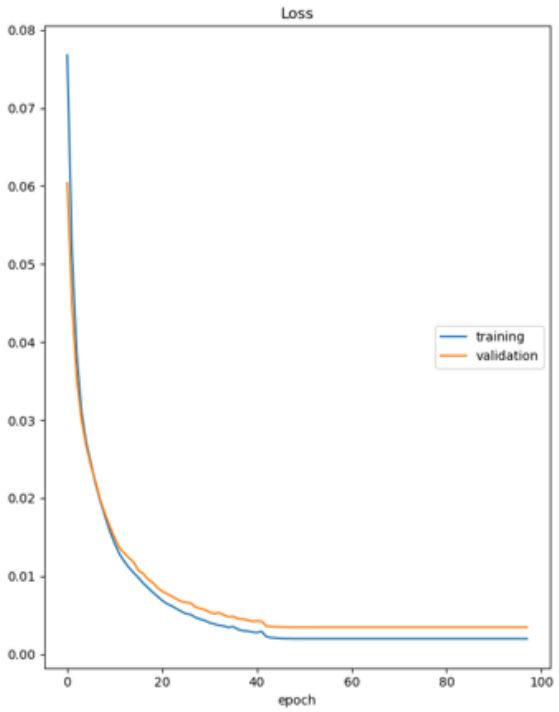

Figure 6.2: Training and Validation Loss Curve before injecting Anomaly

During the testing phase of the LSTM AE model, anomaly-introduced CAN data logs obtained by giving normal data to the anomaly injection algorithm, are given. A total of 20,000 data is taken for testing, out of which 10,000 data are anomalous data points. Reconstruction error is considered a parameter to evaluate the anomalous data. Reconstruction error calculates the difference between the input representation and the output representation.

The Huber loss for each data point is calculated as the reconstruction error. Huber loss is a combination of the mean squared error and the absolute error. The equation for huber loss is given in Equation 6.1.

$$L_\delta(e) = \begin{cases} \frac{1}{2}e^2, & \text{for } |e| \leq \delta \\ \delta(|e| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{6.1}$$

It includes a delta parameter that regulates when the loss function shifts from MSE loss to MAE loss.

The quadratic error (MSE) is calculated when the error is less than a parameter delta, else it is absolute (MAE) error is calculated.

When the error (difference between true and predicted values) is less than or equal to the threshold parameter *delta*, the loss function acts similarly to the MSE loss, which measures the squared difference between true and predicted values. When the difference exceeds *delta*, the loss function takes on the characteristics of the MAE loss, which evaluates the absolute difference between the true and predicted values.

On observing the reconstruction loss, it can be comprehended that the reconstruction loss for anomalous points will be higher than that of normal points.

Figure 6.3 shows the smoothed anomalous reconstruction loss with the threshold calculated from the normal reconstruction loss of the data, such that all data points above the threshold are considered to be anomalous.

Figure 6.3: Loss plot with anomalies

For classifying the anomaly type, a BiLSTM classifier is implemented. The BiLSTM classifier is trained on the anomaly-injected data. The architecture of the BiLSTM model is shown in Figure 6.4.

The Bidirectional LSTM layer with 2048 units with the tanh activation function can process the input data in both forward and backward directions. The final dense layer can classify to which class the anomaly belongs.

During real time inference, only the data that are classified as anomalies are input to the BiLSTM model. However, the BiLSTM model is trained on 4 classes, one normal and three anomaly classes. This is in relation to the fact that the model can accurately learn the distribution of the anomalous data only when it also able to learn the distribution of the normal data. In addition, The BiLSTM model can also act as a secondary point of scrutiny against the anomalous behaviour of data.

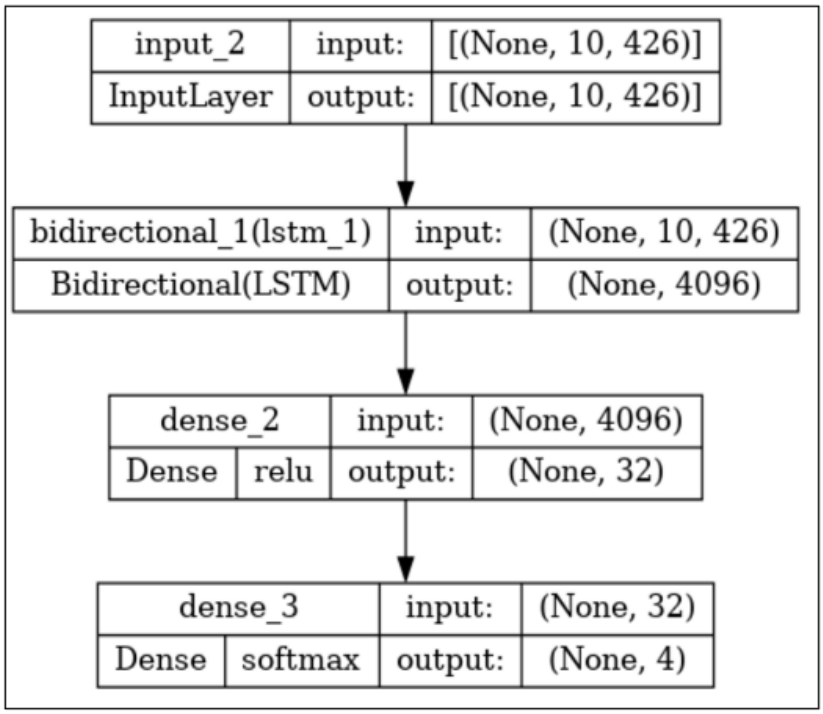

Figure 6.4: BiLSTM model Architecture

The training and validation curve of both accuracy and loss of the model plotted against the number of epochs is shown in Figure 6.5. A significant reduction in training and validation loss curves of the model indicated that the model is learning to accurately reconstruct the input data.

On analysing the accuracy graph, it can be seen that the accuracy shows an increasing trend and since early stopping with a patience 10 was employed during the training of the BiLSTM model, the training and validation of the model stopped at the 59th epoch since there was no further improvement in the accuracy.

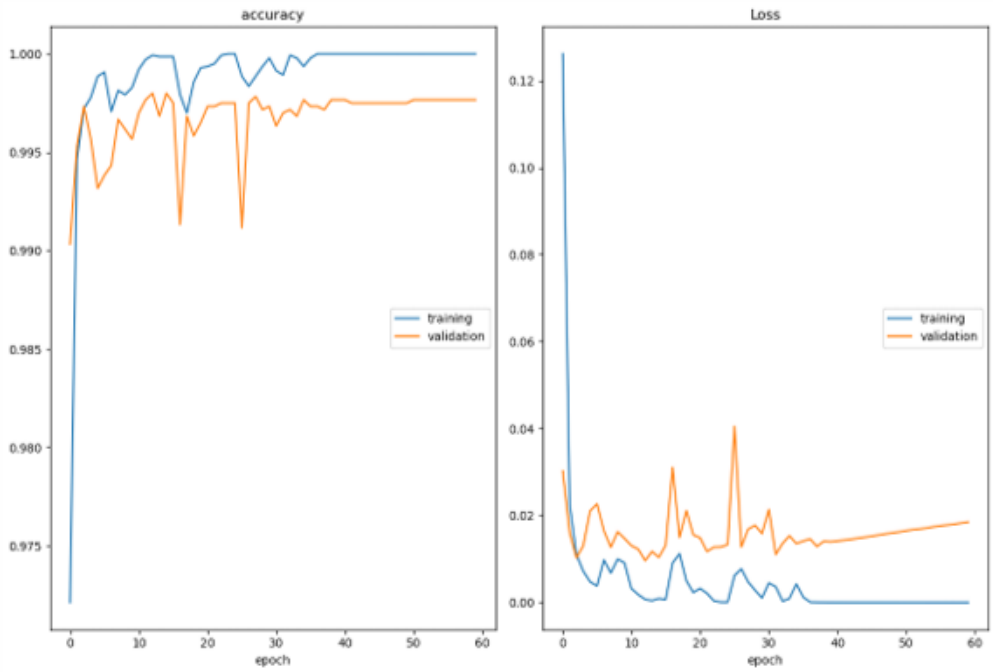

Figure 6.5: Training & Validation - Accuracy & Loss curve of BiLSTM model

6.1 Evaluation Metrics

The performance of the proposed model is evaluated using accuracy (A), precision (P), recall (R), and F1 score (F1) as the evaluation metrics. True Positives (TP), shows the number of anomalies that are correctly predicted, True Negative (TN), shows the number of normal data points that are correctly detected.

False Positives (FP), refers to the number of normal data points which are classified incorrectly as anomalies and False Negatives (FN) refers to the number of anomalies that are incorrectly classified as normal data. The equations of the various evaluation metrics used as shown in Equations 6.2, 6.3, 6.4, 6.5.

Accuracy is defined as a percentage of correctly identified instances in the dataset, or the ratio of correctly identified instances to total number of instances.

Precision is the ratio of accurately detected anomalous occurrences in relation to the total number of anomalous instances identified by the model i.e., precision assesses how well the model eliminates false positives or instances that are not anomalies but are detected as anomalies by the model.

Recall is the ratio of correctly detected anomalous cases in the dataset to the total number of anomalous instances. Recall evaluates how well the model recognizes all true positives or anomalous cases in the dataset.

F1 score is the harmonic mean of precision and recall. The F1 score combines the precision and recall measures into a single metric, providing a more balanced view of the performance of the model than accuracy alone.

$$\mathrm{A} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.2}$$

$$\mathrm{P} = \frac{TP}{TP + FP} \tag{6.3}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{6.4}$$

$$\mathrm{F1} = 2 * \frac{P * R}{P + R} \tag{6.5}$$

6.2 Performance Analysis

To analyze the performance of the LSTM AE model, a confusion matrix is computed using a test data of size 20,000, which includes 10,000 anomalous data. The confusion matrix of LSTM AE model with the percentage of values of TP, TN, FP and FN values obtained while testing the LSTM AE model is shown below in Figure 6.6. As evident in the confusion matrix, the framework is capable of accurately distinguishing between anomalous data and normal data with less false negatives.

The values obtained for the evaluation metrics for LSTM AE model are shown in Table 6.1. It can be observed that the model yields good results for all four evaluation metrics. Precision being the lowest evaluation metric (81.22%) can be attributed to the fact that the presence of false positives is noticeable. However in real time scenario, this would not be an issues as false positives are much more tolerable than false negatives in anomaly detection.

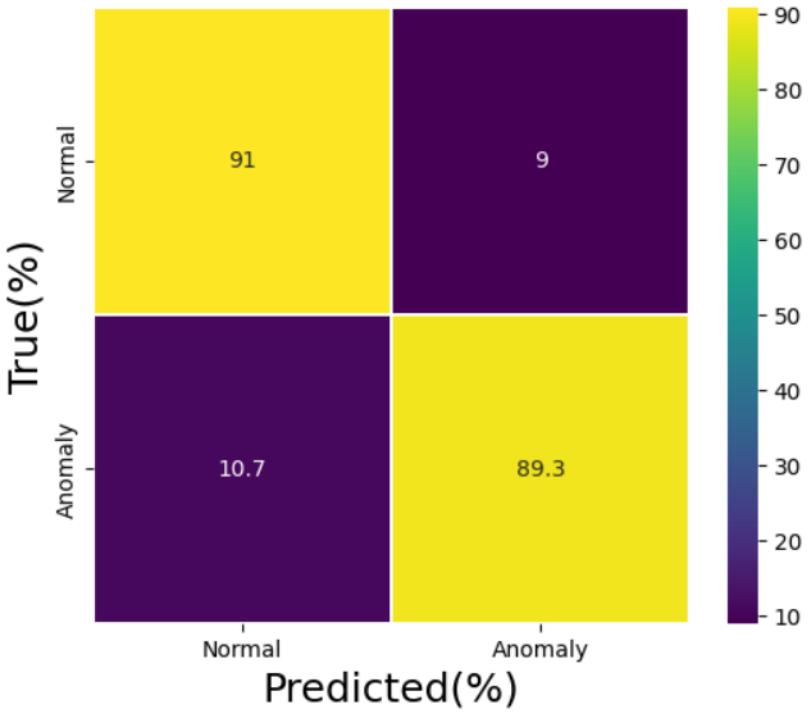

Figure 6.6: Confusion matrix of LSTM AE model

Table 6.1: Evaluation Metrics of LSTM AE model

Evaluation Metrics	Value (in %)
Precision	81.222
Recall	98.285
F1 Score	88.946
Accuracy	99.144

The performance of the BiLSTM classifier is analyzed by testing the model on 20,000 anomalous data, which includes equal distribution of the three anomaly classes - Spike anomaly, Packet loss anomaly, and Stuck-at anomaly.

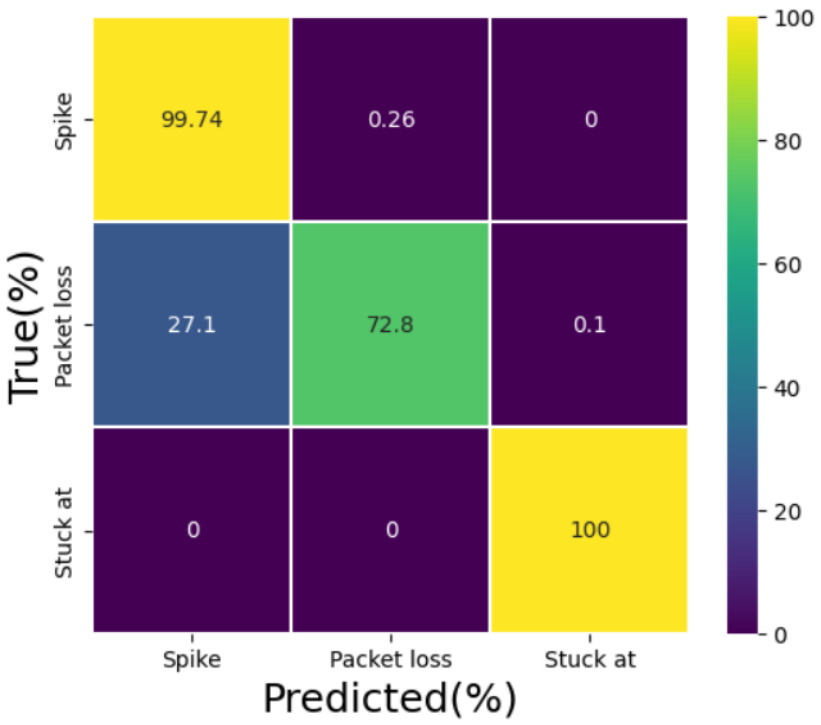

Figure 6.7: Confusion matrix of BiLSTM model

Table 6.2: Evaluation Metrics of BiLSTM model

Classes	Precision(%)	Recall(%)	F1-Score(%)	Accuracy(%)
Spike	79	100	88	91
Packet loss	100	73	84	
Stuck at	100	100	100	

A confusion matrix showing the percentage of values of TP, TN, FP and FN values obtained while testing the BiLSTM model is shown in Figure 6.7. As observed in the conusion matrix, the model shows higher variation in classifying the packet loss anomalies and it accurately classifies stuck at anomalies, among the three classes. The values obtained for the various evaluation metrics are shown in Table 6.2. The classifier shows significant results with an overall accuracy of 91%.

Chapter 7

Conclusion

Anomalies or faults can occur in the ECU signals in a CAN bus and the identification and classification of different anomalies that occur during the testing phase of a vehicle or a full vehicle simulator is important. Anomalies can be manually injected into the CAN data in many ways by giving stuck-at anomalies, packet loss anomalies and spike anomalies. An algorithm for randomly introducing anomalies to the normal CAN data logs is developed. The LSTM Autoencoder shows potential for application in the area of anomaly detection through the process of reconstructing the input sequence and subsequently monitoring the reconstruction error. The identification of an anomaly occurs when the reconstruction error of data points exceeds a specific threshold. Data points that fall below this threshold are classified as normal data. Experimental results using LSTM Autoencoder have given an accuracy of 99.4%, a precision of 81.2%, a recall of 98.2%, and an F1 score of 88.9%. Error smoothing using EWMA before thresholding has led to an improvement in the results obtained. The data points in which anomalies are detected can be further classified using a BiLSTM classifier model with an accuracy of 99%. The inference time of the detection of anomalies in real-time deployment can be significantly reduced using a combination of the two models as only the LSTM AE model needs to be deployed in real-time. The classifier model is needed to be utilized only when an anomaly is detected by the LSTM AE model. Thus, LSTM AE and BiLSTM improve the efficiency of anomaly detection and classification of time series CAN bus data. Additionally, as a future extension, research can be done to check whether a finer sampling rate of data logging would yield a better result. Another future work that can be considered is expanding the framework so as to be implemented in vehicles with various other ECU signals, since the framework utilized here cannot be employed on vehicles whose ECU signals vary form the ones used in this study.

References

[1] M. Fahim and A. Sillitti, "Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review," *IEEE Access*, vol. 7, pp. 81 664–81 681, 2019.

[2] S. Guo, "The application of can-bus technology in the vehicle," in *2011 international conference on mechatronic science, electric engineering and computer (MEC)*. IEEE, 2011, pp. 755–758.

[3] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[4] R. Isermann, J. Schaffnit, and S. Sinsel, "Hardware-in-the-loop simulation for the design and testing of engine-control systems," *Control Engineering Practice*, vol. 7, no. 5, pp. 643–653, 1999.

[5] S. Rajendar and V. K. Kaliappan, "Sensor data based anomaly detection in autonomous vehicles using modified convolutional neural network," *INTELLIGENT AUTOMATION AND SOFT COMPUTING*, vol. 32, no. 2, pp. 859–875, 2022.

[6] M. Abboush, D. Bamal, C. Knieke, and A. Rausch, "Intelligent fault detection and classification based on hybrid deep learning methods for hardware-in-the-loop test of automotive software systems," *Sensors*, vol. 22, no. 11, p. 4066, 2022.

[7] X. Gu, Z. Hou, and J. Cai, "Data-based flooding fault diagnosis of proton exchange membrane fuel cell systems using lstm networks," *Energy and AI*, vol. 4, p. 100056, 2021.

[8] J. Kang, C.-S. Kim, J. W. Kang, and J. Gwak, "Anomaly detection of the brake operating unit on metro vehicles using a one-class lstm autoencoder," *Applied Sciences*, vol. 11, no. 19, p. 9290, 2021.

[9] L. Biddle and S. Fallah, "A novel fault detection, identification and prediction approach for autonomous vehicle controllers using svm," *Automotive Innovation*, vol. 4, no. 3, pp. 301–314, 2021.

[10] Ö. Gültekin, E. Cinar, K. Özkan, and A. Yazıcı, "Multisensory data fusion-based deep learning approach for fault diagnosis of an industrial autonomous transfer vehicle," *Expert Systems with Applications*, vol. 200, p. 117055, 2022.

[11] R. Oucheikh, M. Fri, F. Fedouaki, and M. Hain, "Deep real-time anomaly detection for connected autonomous vehicles," *Procedia Computer Science*, vol. 177, pp. 456–461, 2020.

[12] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 3127–3141, 2019.

[13] Y. Wang, X. Du, Z. Lu, Q. Duan, and J. Wu, "Improved lstm-based time-series anomaly detection in rail transit operation environments," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9027–9036, 2022.

[14] L. W. Alabe, K. Kea, Y. Han, Y. J. Min, and T. Kim, "A deep learning approach to detect anomalies in an electric power steering system," *Sensors*, vol. 22, no. 22, p. 8981, 2022.

[15] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of can bus security challenges," *Sensors*, vol. 20, no. 8, p. 2364, 2020.

[16] R. Sharma, "Big data for autonomous vehicles," *Deep Learning and Big Data for Intelligent Transportation: Enabling Technologies and Future Trends*, pp. 21–47, 2021.

[17] S. R. Gadekar and S. Kodgire, "Lin protocol-emerging trend in automotive electronics," *Research India Publications*, 2013.

[18] K. Anjan, R. Arpitha, P. Keya, and K. Arti, "Flexray protocol in automotive network communications," *International Journal of Engineering Research & Technology*, vol. 3, no. 2, pp. 2278–0181, 2014.

[19] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder based anomaly detection for indoor air quality time series data," *IEEE Sensors Journal*, 2023.

[20] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," *Scientific reports*, vol. 9, no. 1, pp. 1–16, 2019.

[21] H. D. Nguyen, K. P. Tran, S. Thomassey, and M. Hamad, "Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management," *International Journal of Information Management*, vol. 57, p. 102282, 2021.

[22] A. Medvedev, G. Agoureeva, and A. Murro, "A long short-term memory neural network for the detection of epileptiform spikes and high frequency oscillations," *Scientific reports*, vol. 9, no. 1, p. 19374, 2019.

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[24] X. Luo, Y. Jiang, E. Wang, and X. Men, "Anomaly detection by using a combination of generative adversarial networks and convolutional autoencoders," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, no. 1, p. 112, 2022.

[25] G. Xiang and R. Lin, "Robust anomaly detection for multivariate data of spacecraft through recurrent neural networks and extreme value theory," *IEEE Access*, vol. 9, pp. 167 447–167 457, 2021.

[26] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, "Tadgan: Time series anomaly detection using generative adversarial networks," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 33–43.