MALMÖ UNIVERSITY

Faculty of Technology and Society
Department of Computer Science and Media Technology
MS in Computer Science: Applied Data Science

A DEEP LEARNING APPROACH TO VEHICLE FAULT DETECTION BASED ON VEHICLE BEHAVIOR

RAFI KHALIQI & IULIAN COZMA

Master's Thesis
30 Credits

Examiner: Arezoo Sarkheyli-Hägele
Supervisor: Reza Khoshkangini
Date of Submission: June 21, 2023

Abstract

Vehicles and machinery play a crucial role in our daily lives, contributing to our transportation needs and supporting various industries. As society strives for sustainability, the advancement of technology and efficient resource allocation become paramount. However, vehicle faults continue to pose a significant challenge, leading to accidents and unfortunate consequences. In this thesis, we aim to address this issue by exploring the effectiveness of an ensemble of deep learning models for supervised classification. Specifically, we propose to evaluate the performance of 1D-CNN-Bi-LSTM and 1D-CNN-Bi-GRU models. The Bi-LSTM and Bi-GRU models incorporate a multi-head attention mechanism to capture intricate patterns in the data. The methodology involves initial feature extraction using 1D-CNN, followed by learning the temporal dependencies in the time series data using Bi-LSTM and Bi-GRU. These models are trained and evaluated on a labeled dataset, yielding promising results. The successful completion of this thesis has met the objectives and scope of the research, and it also paves the way for future investigations and further research in this domain.

Keywords: Deep learning; attention mechanism; vehicle fault detection, CNN, Bi-LSTM, Bi-GRU, Supervised classification

Acknowledgment

We would like to express our sincere gratitude to our thesis supervisor Reza Khoshkangini for their invaluable guidance, support, and expertise throughout the course of my research. Their insightful feedback, constant encouragement, and dedication have been instrumental in shaping this thesis and our overall academic journey. Their extensive knowledge and passion for the subject matter have inspired us to delve deeper into the field of deep learning. We are truly grateful for their patience, encouragement, and unwavering belief in our abilities. Their mentorship and guidance have not only contributed to the success of this thesis but have also played a significant role in our personal and professional development. We are incredibly fortunate to have had such an exceptional supervisor, and we extend our heartfelt appreciation for their valuable contributions and the opportunity to work under their supervision.

Contents

List of Figures

List of Tables

1 Introduction

This section provides an overview and understanding of the necessity to study vehicle fault detection as well as go through the aim and research questions investigated in this study.

1.1 Background

Engineering professionals have focused their attention primarily on the use and upkeep of machinery since the industrial revolution. Some of the traditional management techniques include reactive maintenance, preventive maintenance, and predictive maintenance [1]. Reactive maintenance is performed only after a failure is detected, while preventive maintenance is time-based and is performed scheduled-based to prevent a failure from happening. Predictive maintenance, on the other hand, employs condition-monitoring tools to measure vehicle performance and construct models that can help in early failure detection. Early failure detection in vehicles saves time and resources and helps entities to plan better and optimize their resources. Despite automobile safety advancing over the last few decades, vehicle defects can still cause accidents. A defect is defined as an unpermitted deviation from the normal condition of at least one essential attribute or parameter of the system [2]. Moreover, autonomous vehicles are capable of reducing and even eliminating human error. Despite, the low levels of failures, autonomous vehicle failures can still happen and they can be caused by mechanical issues or sensors that have broken down [3]. Initially, engineers used their senses to find flaws and malfunctions, such as using their hands to feel vibrations or overheating or their noses to find odors. Later, measuring tools were introduced to provide precise data regarding physical variables. The automated procedure used computers and microprocessors to process the data that had been obtained and improve safety, connectivity, and convenience.

In general, predictive maintenance is developed in four phases:

1. data collection from sensors of the vehicle,
2. data preprocessing,
3. faults diagnosis and prognosis,
4. decision-making on the maintenance strategy.

From the above phases, the first three phases are considered crucial. The first phase is important for achieving accurate and trustworthy data directly from the vehicles. In the second phase, that data is cleaned and prepared for further analysis in the third phase. The third phase is where most of the research has been done, to identify and discover a prognosis method that can achieve accurate and on-time prediction.

By recognizing the precise pattern of behaviors in the observed data, fault diagnosis in engineering systems is related to fault detection in equipment. Predictive maintenance is gradually dislodging more traditional periodic maintenance solutions, such as reactive maintenance and preventive maintenance, due to its greater cost-saving benefits [4] [1]. Vehicle fault diagnosis relies on the processing of signals like magnitude, oscillation, frequency, slope, and so on. Other external sensors and systems include but are not limited to Radio Detection and Ranging (RADAR), Laser Range Finder System (LIDAR), cameras, ultrasonic, Global Positioning System (GPS), Vehicle to Vehicle (V2V), and the fusion of these signals as discussed in [5].

Fault diagnosis and prognosis have attracted huge attention from the academia and industry researchers. The sole purpose of diagnostics is fault detection, isolation, and identification. The first step in diagnostics would be feature extraction and selection, followed by classification of the faults. Prognostics, on the other hand, is based on observing variations in sensor data and identifying abnormalities [1].

A novel strategy for the early detection of quality problems was put forth by Khoshkangini [6], utilizing machine learning to predict failures of a specific component inside a unit. They combined sensor data with warranty claim information, using an auto-regressive model to analyze failure rates and mapping of sensor data to component failure probability.

Khoshkangini [7], in another approach, suggested using the operating records of the vehicle to predict vehicle faults using a snapshot-stacked ensemble deep neural network technique. Three modules of the study were completed. First, the data preprocessing module integrated the data and extracted hidden information. Second, the dimensionality reduction module combined the features using a heuristic optimization approach. Third, the ensemble learning module used the selected features to map the vehicle usage to the breakdowns for the prediction. The Warranty Claim Data (WCD) and the Logged Vehicle Data (LVD) were used as the source data. Their conclusions were reliable and supported the system's capability to detect faults.

Fang et al. [8] used a hybrid approach as another method for finding defective vehicles. To identify the border curve and distinguish between the safe and hazardous domains, they first employed Support Vector Machine (SVM). Using the Kalman filter, they predicted the current position of the vehicle, and upon getting the residuals between the prediction and measurement, the Jarque-Bera test was applied to check the normality of the residual probability distribution and whether the trajectory deviated. They employed a fuzzy method to find errors in the updated neural network. Their results were satisfactory and suggested that it is effective to use such fault detection methods. Berghout [9] asserts that the majority of studies used ensemble approaches to increase accuracy, which implies they used one, two, or even three methods in addition to one or two prediction models. The cost of accuracy enhancement in terms of processing requirements and time-consuming labor-intensive work is not sufficiently covered by these studies on ensemble methods [10]. There is a shortage of knowledge-based method integration, as stated in [10]. Knowledge-based tools can be utilized to supplement the risk data

that the deep learning model infers. A crucial area of study for car diagnosis systems is how to build prediction models with short training time and high identification rates [4].

1.2 Aim & Research Questions

The **aim** of this thesis is to utilize and put together deep learning methods along with knowledge-base graphs to predict vehicles faults with low training time and higher accuracy. Said predictions may then be used outside the scope of this thesis for a confidence-level approach to fault detection.

Deep learning refers to deep neural network, which is a specific configuration where neurons are organized in multiple successive layers, according to [11]. A knowledge graph, often referred to as a "semantic network," depicts the relationships between a network of real-world elements, such as objects, events, situations, or concepts. The name "knowledge graph" was coined since this information is typically stored in a graph database and shown as a graph structure [12].

The novelty of this study is that, the combination of the above-mentioned methods, to the best of our knowledge and the literature we reviewed, have not been studied before. Therefore, it makes this study unique and opens new doors to exploring ensemble of deep learning models together with knowledge bases within the vehicle fault detection domain.

Delimitations: the ensemble models were tested on a small dataset which made the training and testing hard. The dataset did not have 52 weeks of recorded data for each vehicle and a set of interpolation techniques were used to balance the data. The features were also not known to us which posed limited us in normalization and grouping of the sensor features.

To explicitly dictate the intention of this thesis, we articulate and tend to answer to following research questions:

RQ1: *To what extent could an ensemble deep neural network contribute to early vehicle fault detection?*

RQ2: *How can deep learning models be trained and optimized to achieve high accuracy in detecting vehicle faults?*

2 Related Works and Research Gap

This section presents the review of the relevant studies made in this area. It includes previous studies related to neural networks, in particular, convolutional neural networks, long short-term memory, gated recurrent unit, and an ensemble of these methods applied to vehicle fault

detection. There has been numerous research on vehicle fault detection and each one adds value to the importance and cruciality of the phenomena.

González [13] proposed a model for detecting vehicle faults using a multiclass SVM and an Auto-associative Neural Network (AANN). The simulations produced by the model, which made use of historical data for a vehicle, were encouraging. In the context of the AANN framework, they suggested using a Non-Linear Principle Component Analysis (NLPCA) to locate and eliminate correlations between the problematic variables as a tool for dimensionality reduction, visualization, and exploratory data analysis. The idea was tested using incoming VE-DYNA data (a vehicle simulator). There were two stages to the diagnosis. Phase one employed the AANN to gather residuals between the previously learned normal operation behavior and sensor data. When an issue is found, the second phase was launched to provide the definitive diagnosis. It classified the fault that is present and the moment it occurred using a multiclass SVM.

Since sensors are used to collect data, it is recognized that the preprocessing will affect the data quality. Berghout [9] asserts that it is necessary to research the following topics: data availability, data complexity, data drift, and model complexity. In addition to working with Convolutional Neural Networks (CNN), Long Short-term Memory (LSTM) has frequently been utilized to handle dynamic data for remaining life prediction [9] [10] [4]. In this context, CNN is considered for feature extraction from the dataset. It is also recommended to use Gated Recurrent Unit (GRU) in addition to CNN for better accuracy [9].

The sensor data from the vehicles usually come in a sequential and time series format. Recurrent Neural Networks (RNN) can benefit from the sequential nature of the data, but they are unable to handle lengthy time periods [10]. According to Serradilla et al. [14], CNNs are suitable for modelling individual sensor relations with one-dimensional filters and can also model time-based relations among sensors by using two-dimensional filters. CNN is simple and effective and performs faster than other machine learning models requiring less training time and data due to weight-sharing. Berghout [9] suggests compressing the dataset and using auto-encoders for feature compression and estimating the lifespan of components. This is usually done, if the dataset is huge and has a great number of features.

The vehicle's system components are made up of numerous rotating structures and temperature and pressure make components more susceptible to wear and tear [15]. The bearing is the component that is most likely to be damaged in a car, which can result in accidents. Researching bearing failures, battery faults, engine faults, and other issues falls under the umbrella of studying vehicle defects. Predictive maintenance research and the domain of component remaining life prediction are related to the domain of fault prediction in vehicles and industrial equipment. Three types of attention mechanisms can be used to determine how long a component will be useful: time weighting, feature (sensor) weighting, and combined two-dimensional and time weighting [15].

In a study by Zhang et al. [16], the data are compiled into a temporal window. On this foundation, the training data—which serves as the input to train the neural network—is generated using the time window data and accompanying Remaining Useful Life (RUL). Higher weights are given by the temporal self-attention layer to time events that significantly affect RUL prediction. Through the Bi-directional Gated Recurrent Unit (Bi-GRU), the data's time dependence is learned. Feature interactions between sensors and weights across sequences can both be learned via a multi-head attention method (with LSTM layers and Multilayer Perceptron (MLP)) that has been presented in [15]. Their goal is to utilize attention mechanisms, however, before applying the CNN and LSTM the data needs to be cleaned, as the raw signals are noisy [17].

Eknath & Diwakar [18] proposed a prediction method for rolling bearings combining a Deep Convolutional Neural Network (DCNN) and a GRU, in which feature extraction utilizes the DCNN to extract vibration signal characteristics, where the connected layer is substituted with the max-pooling layer to increase extraction accuracy. In addition, they used a Bi-GRU that can concentrate on past data while embedding future information to enhance the GRU model's potential to assimilate data. Their results show that the implementation of DCNN with Bi-GRU is shorter and has a lower error value of the predicted results.

On a similar approach, Gong and Wang [19] proposed a Temporal Convolutional Neural Network (TCN)-Bi-GRU model for remaining useful life prediction using the attention mechanism. A TCN is a variant of CNN, which can capture the long-term and short-term characteristics of time series more flexibly and effectively and solve the structural effects of the RNN model network. In this study, the attention mechanism is used to capture the relationship between multisensory signals, and the features with the attention mechanism are input into the TCN-Bi-GRU network to complete the RUL prediction. Their experiments verify that the proposed model is suitable for multi-sensory scenarios, and it achieves better RUL prediction than the comparison method.

Sun and Wang [20] studied multi-source fault detection based on multi-level knowledge graph and Bayesian theory reasoning. Their study consisted of four phases: building an offline multi-level knowledge graph, mining deep associated paths, fault detection, and fault diagnosis. In the first phase, they made sure that the data is comprehensive, and using Pearson Correlation, they found the correlation between entities. In the second phase, they calculate the variable correlation coefficient of each variable in the knowledge graph as the weight coefficient of the variable. For the fault detection phase, they introduced a discriminant coefficient R rule. This discriminant coefficient needs to be greater than an acceptable fault tolerance rate ε. For the final phase, the highly correlated variable pairs are selected in the knowledge graph relationship, to construct a multi-level knowledge graph fault model. Then, for each candidate fault reason, a posterior probability based on Bayesian theory is calculated. This probability needs to be under a certain threshold otherwise, it is a likely cause of the failure. The study results show that, the

discriminant coefficient R completely separated the data of the normal state and faulty state and the method works well for reasoning the multi-source failure of complex systems.

3 Problem Formulation

Automobiles and machinery have become an important part of our daily life and we humans have, somewhat, become dependent on them. Constant improvement and efficient allocation of the maintenance resources would help us with the sustainability of human life in the long run. Vehicle and machinery fault accidents are still taking human lives and causes life and property damages. It is important to identify and solve faults before they happen to prevent such disasters. Vehicle early fault detection has gained popularity among researchers in recent years, and many researchers have proposed different algorithms and machine learning models to predict faults before it happens. Their experiments are based on the vehicles' historical data with an aim to identify trends that could cause a potential vehicle fault. In this study, we aim to use vehicles' historical data and perform deep learning methods to extract features and identify trends that could cause faults. This thesis seeks to test an ensemble of deep learning models on a supervised classification problem, with the aim to extract knowledge and be able to propose a general framework for vehicle fault detection. We propose to test an ensemble of attention-based 1D-CNN-Bi-LSTM, 1D-CNN-Bi-GRU, and Graph Neural Network (GNN). The dataset used in this thesis lacks long-time series predictions, therefore we intend to utilize Bi-LSTM and Bi-GRU to supplement 1D-CNN.

4 Data Representation

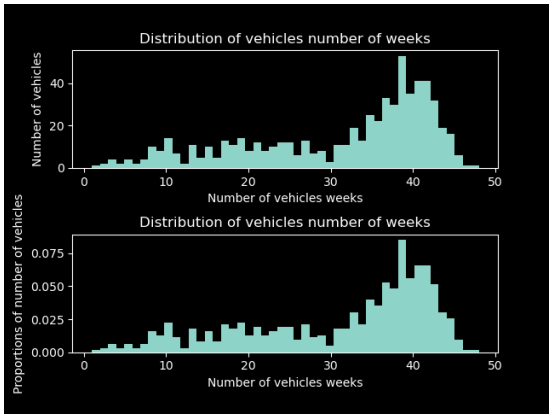
This section provides an overview of the dataset, the preprocessing steps, and the interpolation used for filling in the missing values.

The dataset used in this thesis was anonymous and provided by our supervisor. The dataset is from 2019 and represents recorded data for 637 vehicles over a period of 52 weeks. For each vehicle, there are 141 recorded signals (features) and the vehicles are labeled as either healthy or faulty. In Figure 1, you can see the distribution of recorded signals for each vehicle. In Figure 1a, it shows that majority of the vehicles have recorded signals from 35 to 45 weeks.

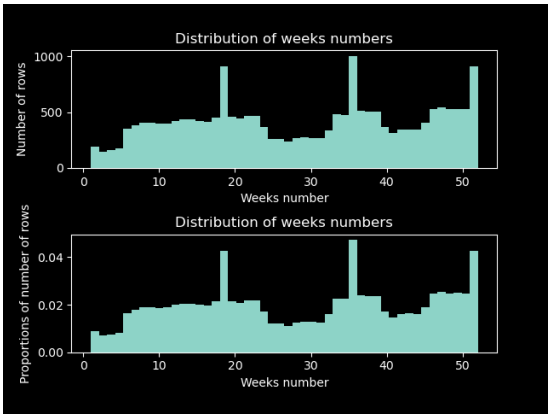
The goal is to achieve a standard dataset with the same number of recorded weeks for all vehicles to be trained by the models.

The initial steps to begin were, to check for the following:

- The dataset is clean.
  - Three feature sensors were dropped that had only zero values.



(a) Vehicles Number of Weeks



(b) Dataset Number of Weeks

Figure 1: Weeks Distributions

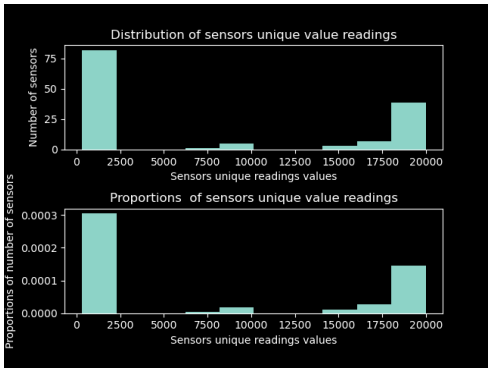
  - One feature that was a duplicate of the vehicles id feature was also dropped.
  - One feature that represents only year 2019 was also dropped.

- There are no duplicates or Not a Number (NaN).
- The dataset has two balanced categories for training.
- The dataset has different number of weekly sensor readings for each vehicle ranging between (1 - 48) weeks.
- The dataset is standardized.
- The dataset is normalized.
- A min-max scaling is performed.
- Quantile transformer is performed.

Unfortunately, we are unable to know what kind of sensor data (features) we have, due to the dataset being anonymized. This creates problems in understanding the dataset and its normalization. In addition, it would be hard to use this dataset for knowledge base graphs, as the edges and components of this dataset is not known to establish relationships.

The number of unique values for many sensors are less than the total number of rows (20 000). There are 81 sensors with unique values less than 800. As it is shown in Figure 2, there are some sensors that have 20 000 unique values, while other sensors values are interpolated or adjusted to fill the missing values.

The distribution of the two labels for the sensors unique values are of 50% and for some sensors it is 60% and 40%. Therefore, the sensors unique values are well balanced between the



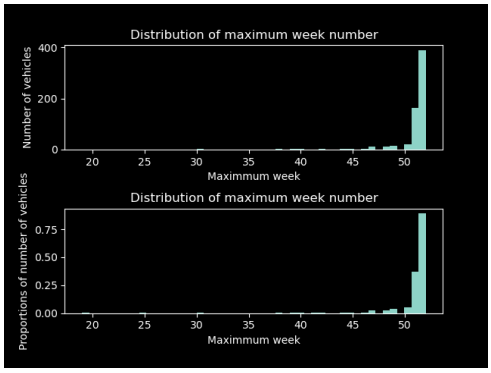
(a) Sensors Unique Value Readings

(b) Sensors Value Readings < 800

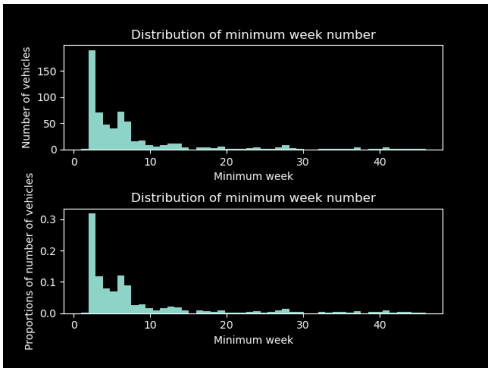
Figure 2: Distribution of Recorded Signals Unique Readings

two labels. The majority of vehicles have around 37 weeks of sensors readings shown in Figure 2b.

The majority of vehicles have maximum number of weeks at the end of the year and the minimum number of weeks at the beginning of the year as shown in Figure 3.



(a) Vehicles Maximum Number of Weeks



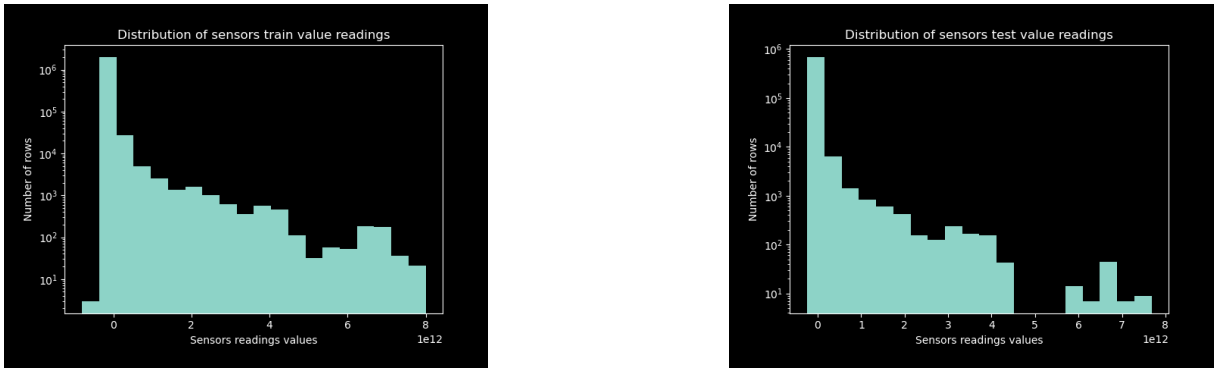
(b) Vehicles Minimum Number of Weeks

Figure 3: Maximum and Minimum Number of Weeks

The dataset was divided into two separate subsets: a training dataset comprising 75% of the original dataset and a test dataset containing the remaining 25%. This division ensures that a significant portion of the data is used for training the model, while a separate portion is reserved for evaluating its performance.

In Figure 4, the distribution of the number of vehicles in the dataset is visualized after applying a logarithmic transformation. The log transformation is employed to achieve a more normalized distribution and potentially enhance the model's ability to capture meaningful patterns and relationships within the data.

The train dataset is unbalanced by 19 vehicles (3.983%). From the train dataset, the 19 vehicles having minimum number of weeks were removed, so that the resulting train dataset has both labels with the same number of vehicles. The test dataset is unbalanced by 4 vehicles



(a) Train Dataset Sensor Values (b) Test Dataset Sensor Values

Figure 4: Dataset Sensor Log Transform Values

(2.5%). The train and test datasets values shown in Figure 4 are left skewed and have many zeros as a result of an anonymous interpolation of some sensors to match other sensors real range values, performed before the data was given for this project. A summary of the vehicles in the train and test dataset is presented in Table 1, where label 0 refers to healthy vehicles and label 1 refers to faulty vehicles.

Train Label 0: 229 vehicles
Train Label 1: 248 vehicles
Test Label 0: 78 vehicles
Test Label 1: 82 vehicles

Table 1: Train and Test Splits of the Dataset

4.1 Dataset Interpolation

The training and test datasets are interpolated to balance the number of sensor readings across all vehicles. Since the majority of the vehicles (390 vehicles) had data recorded for 52 weeks and 163 vehicles had data recorded for 51 weeks, 52 weeks were chosen as the number of interpolated readings.

The algorithms and interpolation methods were designed such that; for each vehicle, interval bins were constructed that took into consideration the week numbers of the previous readings and had bins that started at 1 and ended at 52. For instance, if the vehicle's sensor readings are 1, 5, 15, 23, 37, and 52, the following bins are generated: 1, 5, 15, 23, 37, and 52. The weeks that need to be interpolated are then added to all of the bins, increasing the number of interpolations added to a larger bin as needed. As a result, the algorithm distributes several readings over the interval (1, 52).

The number of weeks for all the cars in the train and test datasets was raised by interpolation to 52.

Initially, the training dataset was of 458 vehicles of 14,792 rows with 139 total features(136 sensor features) and after weeks interpolation the train dataset increased to 23,816 rows.

The test dataset was of 160 vehicles of 5,057 rows and after weeks interpolation the test dataset increased to 8,320 rows.

The datasets interpolated with NaNs is processed in two ways:

1. The NaNs are filled with zeros.

2. The NaNs are imputed with the time series interpolation tool as shown in [21].

The NaNs to be filled had 0.378% and 0.392% proportions for train and test dataset respectively.

4.1.1 Interpolation with Zeros

The train and test datasets underwent two specific preprocessing steps. First, they were interpolated with zeros to address missing values or gaps within the data. This interpolation process involves filling the missing data points with zeros, allowing for a continuous and complete representation of the dataset.

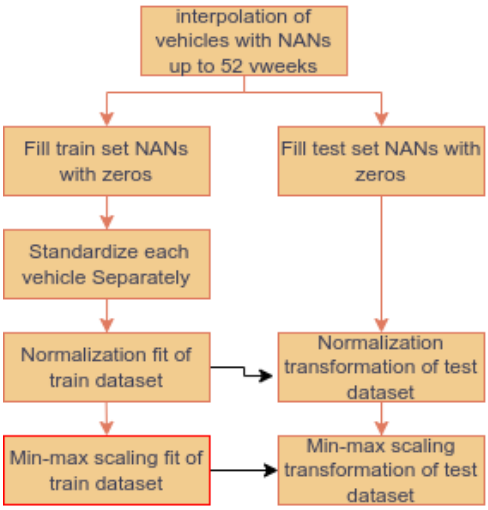


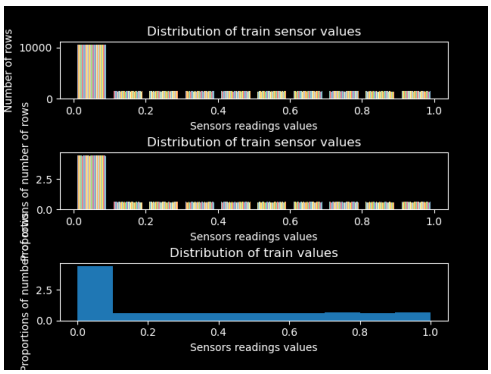
Figure 5: Interpolation with Zeros Processes

Additionally, the datasets were transformed using certain processes, as illustrated in Figure 5. These transformation processes aim to modify the dataset in a way that enhances its suitability for subsequent analysis or modeling tasks.

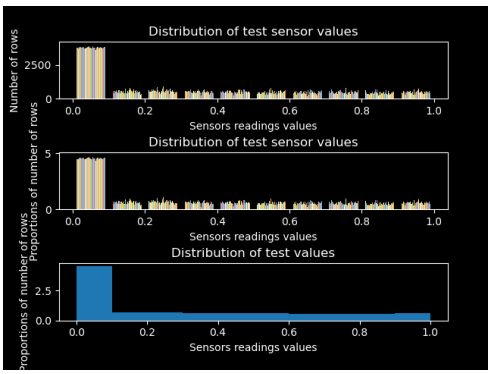
By visualizing Figure 5, one can gain insights into the specific transformations applied to the train and test datasets. These transformations may include normalization, feature scaling,

dimensionality reduction, or other preprocessing techniques tailored to the specific requirements of the data and the chosen analysis or modeling approach.

Figure 6 illustrates how the interpolation with zeros enhanced the left skewness of the train and test datasets.

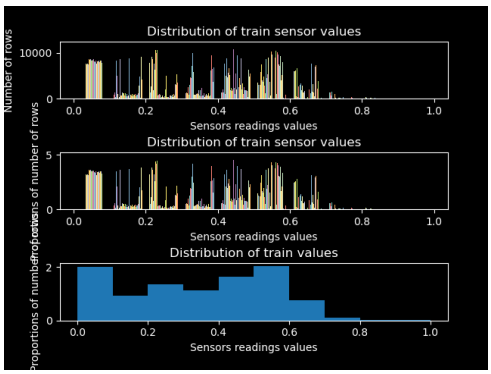


(a) Train Interpolation with Zeros

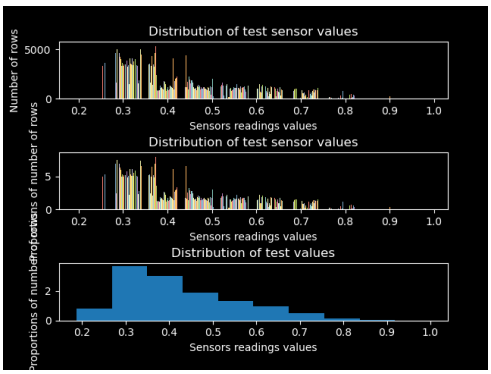


(b) Test Interpolation with Zeros

Figure 6: Dataset Interpolation with Zeros



(a) Train Interpolation with Zeros



(b) Test Interpolation with Zeros

Figure 7: Dataset Interpolated with Zeros and Normalized

The left skewness is still present after the normalization process as seen in the Figure 7. Therefore, interpolation with imputed values were performed in the next stage.

4.1.2 Interpolation with Imputed Values

The process of train dataset interpolation with values had similar process as interpolating with zeros. The results of the interpolation with values is shown in Figure 8.

The interpolation on train data did not add outliers, and the data is similar to a normal distribution. After the interpolation stage and after filtering only the train dataset with an Unscented Kalman Filter, the train dataset and the test data interpolated with zeros were normalized.

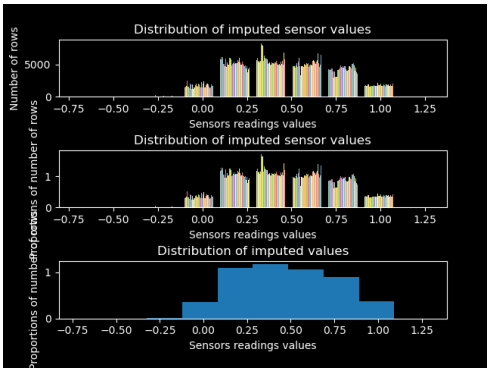
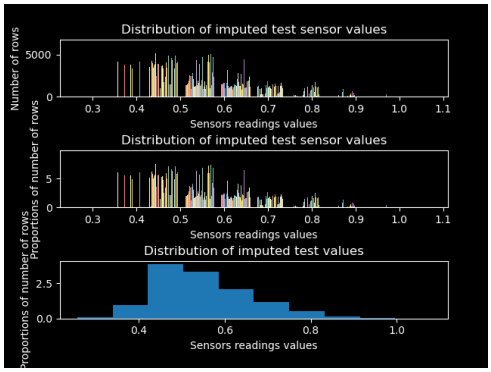
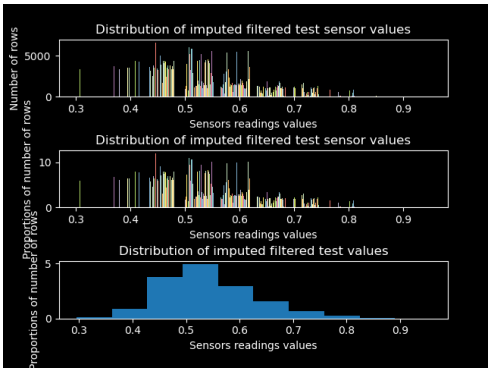


Figure 8: Train Dataset Interpolation

The comparison of the effects of the two process on test data is shown in Figure 9 and Figure 10.

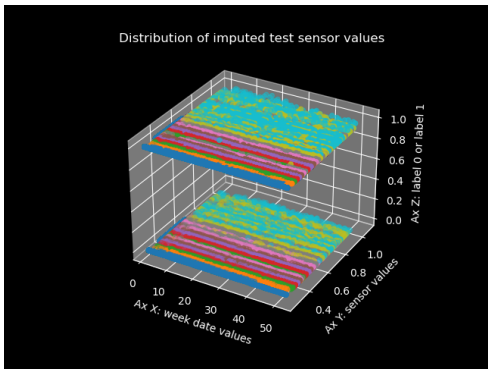


(a) Test Data Interpolated Standardized

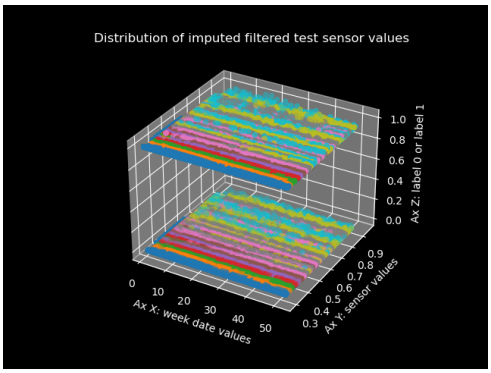


(b) Test Data Interpolated and Filtered

Figure 9: Dataset Interpolated or Filtered Standardized



(a) Test Data Interpolated Standardized



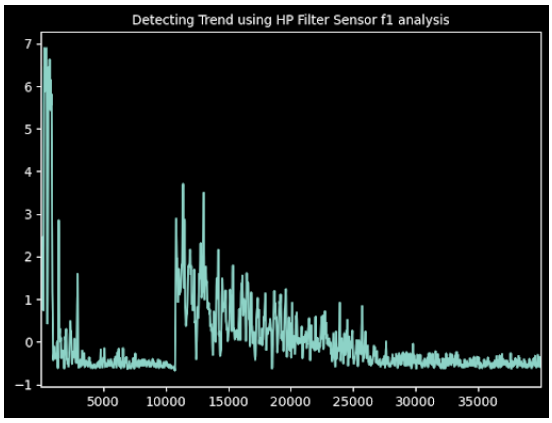
(b) Test Data Interpolated & Filtered

Figure 10: Test Data Interpolated or Filtered Standardized

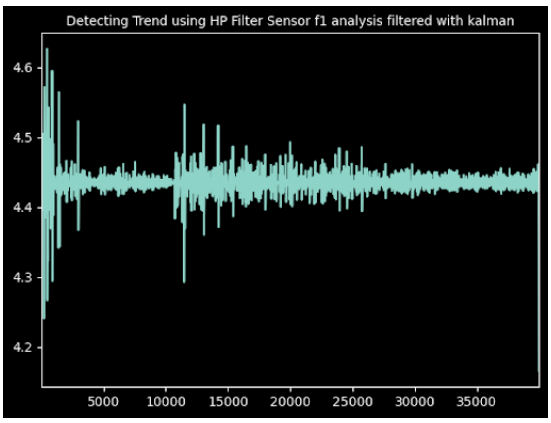
As it can be observed from Figure 9b, part of the residual skewness was removed by the filtering with Unscented Kalman filter, bringing the data closer to a normal distribution.

To remove trends and noisy readings, the Unscented Kalman Filter is applied separately to each sensor feature column.

An illustration of one sensor's readings using this method is shown in Figure 11.



(a) F1 Sensor without Kalman Filter



(b) F1 Sensor with Kalman Filter

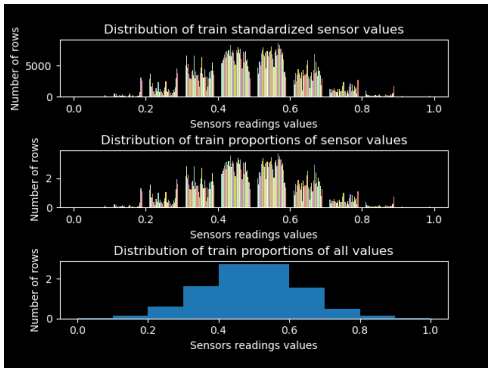
Figure 11: Application of Kalman Filter

4.1.3 Feature Extraction

The feature extraction approach uses a deep autoencoder with encoder and decoder made from dilated Temporal Convolutional Neural Network (TCN) to automate feature extraction using deep unsupervised learning [22].

The feature extraction was applied to 136 sensors, and 34 features were obtained. The feature extraction was fitted to the train dataset and applied to the train and test datasets. The features extracted were scaled with the min-max scaler from Python library, Scikit-Learn.

The data distribution of the extracted features of the Kalman-filtered data is shown in Figure 12.



(a) Train Data Extracted Features

(b) Test Data Extracted Features

Figure 12: Dataset Extracted Features

4.2 Graph Dataset Representation

For implementing Graph Neural Network (GNN), the dataset needed to be transformed. The nodes of the graph do not have a dependence on each other, due to the vehicles being independent and therefore, is an undirected, static, and homogeneous graph [23]. The computational methods that are improving the model include propagation and pooling modules.

The experiments in this study will do graph-level fault diagnosis, which better fits sensor data. For knowledge graph functionality, in other words, identifying faulty vehicles, node level, and edge level tasks have to be designed. The intention is to employ a Graph Auto-Encoder (GAE) to improve the K-Nearest Neighbor (KNN) approach and locate neighbor nodes [24].

5 Technical Background

This section provides an overview and detailed description of the methods that are used in this study. It begins with CNN and continues with Bi-LSTM, Bi-GRU, attention mechanism, ensemble of these models, and knowledge-base graphs.

5.1 Convolutional Neural Networks

CNN is mainly used for parameter reduction in Artificial Neural Network (ANN) and feature extraction [25]. Deep CNNs were originally introduced for object recognition tasks for 2-Dimentional (2D) signals such as images and video frames and is extensively used in many Computer Vision and Pattern Recognition tasks. To utilize CNN with a 1-Dimentional (1D) signal processing application, it is required to do a conversion from 2D to 1D [26]. A common technique for this conversion is to directly reshape the vibration signal into an $n \times m$ matrix called "the vibration image". 1D-CNNs work on 1D signals with limited labeled data and high signal variations. 1D-CNNs consist of two types of layers: 1. CNN-layers, where both 1D convolutions and pooling occur, and 2. Fully-Connected layers that are identical to the layers of a typical MLP and hence called MLP-layers. The configuration of a 1D-CNN is determined by the number of hidden CNN and MLP layers, filter size in each CNN layer, sub-sampling factor in each CNN layer, and selection of pooling and activation operators.

Neurons of the hidden CNN layers are extended to enable both convolution and sub-sampling operations without the MLP layers and having the freedom of any input layer dimension. Therefore, feature extraction and classification operations are fused into one process to maximize classification performance. The main advantage of 1D-CNN is the low computational complexity, since it takes linear weighted sums of two 1D arrays and can perform forward and back-propagation in parallel. Forward Propagation (FP) from the previous CNN layer $l-1$, to

create the input of the kth hidden neuron on the next layer, l, can be expressed as [26] [27]:

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D(w_{ik}^{l-1}, s_i^{l-1}) \tag{1}$$

, where x_k^l is the input, b_k^l is the bias of the kth neuron at layer l, and s_i^{l-1} is the output of the ith neuron at layer $l-1$. w_{ik}^{l-1} is the 1D kernel from the ith neuron at layer $l-1$ to the kth neuron at layer l.

For Back Propagation (BP) we let $l = 1$ be the input layer and $l = L$ be the output layer. The Mean-Square-Error (MSE) in the output layer can be expressed as:

$$E = E(y_1^L, \cdots, y_{N_L}^L) = \sum_{i=1}^{N_L} (y_i^L - t_i)^2 \tag{2}$$

To minimize the error using the gradient descent method, the partial derivatives of the error with respect to an individual weight w_{ik}^{l-1}, and bias of the neuron k, b_k^l is taken. Upon determining the delta errors in each MLP layer the weights and bias of each neuron are updated by the gradient descent method. Δ_k^l is used to update the bias of kth neuron and all weights of the neurons in the previous layer connected to that neuron, following:

$$\frac{\partial E}{\partial w_{ik}^{l-1}} = \Delta_k^l y_i^{l-1} \text{ and } \frac{\partial E}{\partial b_k^l} = \Delta_k^l \tag{3}$$

From the input MLP layer to the output CNN layer, the regular BP is performed as:

$$\frac{\partial E}{\partial s_k^l} = \Delta S_k^l = \sum_{i=1}^{N_{l+1}} \frac{\partial E}{\partial x_i^{l+1}} \frac{\partial x_i^{l+1}}{\partial s_k^l} = \sum_{i=1}^{N_{l+1}} \Delta_i^{l+1} w_{ki}^l \tag{4}$$

After the first BP is performed from the next layer, $l+1$, to the current layer, l, we can further back-propagate it to the input delta Δ_k^l. Let zero-order up-sampled map be $us_k^l = up(S_k^l)$, then it can be written as:

$$\Delta_k^l = \frac{\partial E}{\partial y_k^l} \frac{\partial y_k^l}{\partial x_k^l} = \frac{\partial E}{\partial us_k^l} \frac{\partial us_k^l}{\partial y_k^l} f\prime(x_k^l) = up(\Delta s_k^l) \beta f\prime(x_l^l) \tag{5}$$

, where $\beta = (ss)^{-1}$ since each element of s_k^l was obtained by averaging ss number of elements of the intermediate output y_k^l. The BP among CNN layers of the delta error can be expressed as follows:

$$\Delta s_k^l = \sum_{i=1}^{N_{l+1}} conv1Dz(\Delta_i^{l+1}, rev(w_{ki}^l)) \tag{6}$$

, where $rev(\cdot)$ reverses the array and $conv1Dz(\cdots)$ performs full convolution in 1D with $K-1$ zero padding. At last, the weight and bias sensitivities can be expressed as follows:

$$\frac{\partial E}{\partial w_{ki}^l} = conv1D(s_k^l, \Delta_i^{l+1}) \text{ and } \frac{\partial E}{\partial b_k^l} = \sum_n \Delta_k^l(n) \tag{7}$$

The iterative flow of the BP can be formulated as:

1. Initialize weights (usually randomly)
2. For each BP iteration Do:
   2.1 For each item in the dataset, DO
      i. FP: Forward propagate from the input layer to the output layer to find outputs of each neuron at each layer.
      ii. BP: Compute delta error at the output layer and back-propagate it to the first hidden layer to compute the delta errors.
      iii. Post-process to compute the weight and bias sensitivities
      iv. Update the weights and biases with the sensitivities found above scaled with the learning factor, ε

$$w_{ik}^{l-1}(t+1) = w_{ik}^{l-1}(t) - \varepsilon \frac{\partial E}{\partial w_{ik}^{l-1}} \tag{8}$$

$$b_k^l(t+1) = b_k^l(t) - \varepsilon \frac{\partial E}{\partial b_k^l} \tag{9}$$

Papers by Kiranyaz et al. [27] and Ince et al. [28], also support and imply that 1D-CNNs can optimize both feature extraction and classification in a single learning body and show that a real-time monitoring and anomaly detection can be accomplished with higher accuracy.

5.2 Bi-Directional Long Short-term Memory

LSTM network is a powerful class of RNN that allows feedback connections and overcomes the vanishing gradient problems arising from learning long-term dependencies [29]. RNNs are designed to work with sequential and time series data. The connection to the vehicle fault detection problem stated here makes RNN networks an attractive model structure to explore. According to a review by Ersöz et al. [30], LSTM is the most used model for predictive maintenance followed by Artificial Neural Network.

Time series data can also be used to train a classifier as Lei et al. [31] did a case study on fault diagnosis. They used a LSTM-based framework for condition monitoring of wind turbines and capturing long-term dependencies using their sensor data. Their experimental results show that LSTM outperformed other state-of-the-art methods. The LSTM architecture consists of three gates (i.e., input gate, output gate, and forget gate). The three gates are calculated as follows:

$$f_t = \sigma(w_f[h_{t-1};x_t] + b_f) \tag{10}$$

$$i_t = \sigma(w_i[h_{t-1};x_t] + b_i) \tag{11}$$

$$o_t = \sigma(w_o[h_{t-1};x_t] + b_o) \tag{12}$$

, where w_f, w_i, and w_o are the weighted matrices, b_f, b_i, and b_o are the biases to be learned during the training, i_t, f_t, and o_t are the input gate, forget gate, and output gate, respectively. The hidden state of LSTM is h_t and $[h_{t-1};x_t]$ is a longer vector connected by the two vectors; σ is the sigmoid function.

$$\tilde{c}_t = tanh(w_c[h_{t-1};x_t] + b_c) \tag{13}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{14}$$

Equation 13 is the candidate memory unit that generates alternative update information, and equation 14 is the process of updating the state of the cell. The information to be retained from the previous cell state is chosen in forget gate and the input gate determines how much of the input x_t is saved to c_t. $\odot$ in the equation above denotes element-wise multiplication.

The final output value is calculated in the following manner:

$$h_t = o_t \odot tanh(c_t) \tag{15}$$

, where the output gate o_t controls how much of the current cell state is discarded.

Bi-Directional Long Short-term Memory (Bi-LSTM) divides the RNN neurons into forward state and backward state. The output layer can simultaneously obtain information from the forward and backward states [32]. In that sense, Bi-LSTM is more suitable for time series prediction. The input time series from x_1 to x_t is read and a sequence of forward hidden states $(\overrightarrow{h_1}, \cdots, \overrightarrow{h_1})$ is calculated in the forward LSTM. The backward LSTM, however, reads the input

sequence in reverse order from x_t to x_1 and generates a sequence of backward hidden states $(\overleftarrow{h_1}, \cdots, \overleftarrow{h_1})$. The final step is to concatenate the forward hidden state $\overrightarrow{h_i}$ and the backward hidden state $\overleftarrow{h_i}$ to achieve the final latent vector representation as $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]^T$.

5.3 Bi-Directional Gated Recurrent Unit

The GRU is a type of RNN with lesser gates compared to LSTM [33]. For each position t, GRU computes h_t with input x_t and previous state h_{t-1} as:

has only two gates namely the reset gate r_t, and update gate z_t calculated in the following manner:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{16}$$

$$u_t = \sigma(W_u x_t + U_u h_{t-1}) \tag{17}$$

$$\tilde{h}_t = tanh(W_c x_t + U(r_t \odot h_{t-1})) \tag{18}$$

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t \tag{19}$$

, where r_t, u_t, and h_t are reset gate, update gate, and d-dimensional hidden state, respectively. W_r, W_u, W_c, and U_r, U_u, U_c are the parameters of the GRU, $\odot$ denotes the element-wise multiplication and σ is the sigmoid function.

Bi-directional Gated Recurrent Unit (Bi-GRU) also divides the RNN neurons into forward and backward states. The input time series from x_1 to x_t is read and a sequence of forward hidden states $(\overrightarrow{h_1}, \cdots, \overrightarrow{h_1})$ is calculated in the forward GRU. The input sequence from x_t to x_1 is read in reverse order and backward hidden states $(\overleftarrow{h_1}, \cdots, \overleftarrow{h_1})$ is generated.

Then, the forward hidden state $\overrightarrow{h_i}$ and the backward hidden state $\overleftarrow{h_i}$ is concatenated to achieve the final latent vector representation as $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]^T$ [34].

5.4 Attention Mechanism

Attention Mechanism (AM) is used to strengthen the impact of important information. AM can assign different weights to the implied states of LSTM and GRU through mapping weight and parameter learning. Each input data has a corresponding weight value, and the weight value is multiplied by the input data to get the corresponding attention degree. The greater the

weight, the stronger the attention. AM addresses two drawbacks of LSTM. First, the traditional recurrent way to reconstruct the depth of LSTM is substituted by AM. Second, AM is used over the output layers of LSTM to model long-term dependencies [35]. The neural network that is used for AM is usually called an attentive neural network. AM is applied to LSTM and GRU to concentrate on features that have a great impact on the output variables and to increase the accuracy of the method [36]. The deep learning model combined with AM is also called a weight matrix.

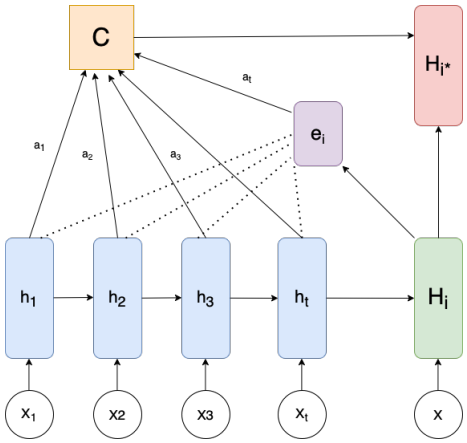


Figure 13: Structure of AM

The structure of AM is presented in Figure 13, where $(X_1, X_2, \cdots, X_t)$ denotes the LSTM/GRU input, and $(h_1, h_2, \cdots, h_t)$ is the LSTM/GRU output, which is used as input for AM, to get the attention weight distribution. It is important to mention that time series sensor data contains more complex temporal information, therefore Sun et al. [32] suggests using a *soft attention* to derive a context vector c_t to capture relevant information for predicting future value $\hat{y}_t$.

The AM is calculated in the following way:

$$\alpha_i = \frac{e^{e_i}}{\sum_i e^{e_i}} \tag{20}$$

After calculating the corresponding weight score, the softmax function is used to normalize the score to obtain the conditional probability distribution α_i, which represents the importance of ith time window for prediction. The score e_i is calculated below:

$$e_i = tanh(W^T h_i + b) \tag{21}$$

, where W and b are parameters to be learned. The attention mechanism computes the context vector c_t as a weighted sum of all the hidden states h_1 to h_{t-1}.

$$c_t = \sum_i \alpha_i h_i \tag{22}$$

, given the context vector c_t and the current hidden state h_t, we combine the information of the two vectors to generate an attentional hidden state as below:

$$h^* = tanh(w_h[c_t; h_t]) \tag{23}$$

The above attentional vector h^* is then fed into the dropout layer to prevent overfitting and thereafter, the dense layer is used to predict the next data.

AM has some limitations such as the attention weight being calculated using the whole information of the embedding. LSTM and GRU can gain access to the key places in the time series sequence and benefit from attention-based information retrieval. The multi-head attention method, which was also employed in fault classification previously, will provide more detailed information [37]. This method consists of splitting the encoded representations of the sequence into homogeneous sub-vectors called heads. The multi-head attention can be formulated with queries (Q), keys (K), and values (V) as follow:

$$MultiHead(Q,K,V) = Concat(head_1, \cdots, head_h)W^O \tag{24}$$

For h sets of learned projection metrics, there are h attention head, where $head_i = Attention$ (QW_i^Q, KW_i^K, V_i^W) with the queries and keys having the same dimensionality due to the dot product, and can be represented with $W_i^Q \in R^{D\times d_k}, W_i^K \in R^{D\times d_k}$, and $W_i^V \in R^{D\times d_v}$. The h outputs are concatenated and multiplied by the weight matrix W^O, which is a squared matrix represented by $W^O \in R^{hd_v\times D}$ [38].

5.5 Knowledge-based Graph

Knowledge graph has gained popularity among researchers due to being a powerful tool to represent knowledge in the form of a labeled directed graph and to give semantics to textual information. A knowledge graph is a kind of semantic network, an artifact whose scope, characteristics, features, and even uses, remain open and are in the process of being defined. A knowledge graph as defined in [39], is a graph constructed by representing each item, entity, and user as nodes, and linking those nodes that interact with each other via edges. A knowledge graph describes real-world entities and defines possible relations of entities in a schema. Knowledge graphs improve the explainability of machine learning models.

Knowledge graph has five main components: knowledge representation learning, knowledge storage, knowledge graph construction, knowledge updating, and knowledge reasoning. A knowledge graph needs a certain constraint and norm to form logical architecture and is divided into a pattern layer and a data layer [12]. The pattern layer represents the data structure of knowledge and definitions of knowledge classes. The knowledge triples in the data layer are

units of information storage. Therefore, knowledge graphs can be formally expressed as below:

$$G = E, R, F \tag{25}$$

, where E represents the set of entities $e_1, e_2, \cdots, e_i$, and is the basic element of a knowledge graph. It represents things that can be distinguished from each other. R, which is the edge of a certain connection between two different entities, represents the set of relationships $r_1, r_2, \cdots, r_i$. F represents the set of facts $f_1, f_2, \cdots, f_i$, and each fact is defined as a triple $(h, r, t) \in F$, where h, r, and t is the head entity, relationship, and tail entity, respectively.

Graph Neural Network (GNN)s are deep learned-based methods that operate on the graph domain. A GNN is a neural network where nodes are connected to their neighbors in the data graph. GNNs support end-to-end supervised learning for specific tasks and can classify elements of the graph. Knowledge graph reasoning methods based on neural networks emerge endlessly due to their generality of structure and convenience of reasoning. Based on CNN and graph embedding, variants of GNN are proposed to aggregate information from the graph structure. GNNs are divided into four categories: recurrent graph neural networks, convolutional graph neural networks, graph autoencoders, and spatial-temporal graph neural networks [23].

The general design of a GNN model includes graph structure, graph type and scale, designing loss function, and building a model using computational modules. The first step is to identify the graph structure as either a structural scenario or a non-structural scenario. In the second step, we have three categories.

1. Directed/Undirected Graphs: Edges from directed graphs are all directed from one node to another, which provides more information than undirected graphs.
2. Homogeneous/Heterogeneous Graphs: Nodes and edges in homogeneous graphs have the same types, which is different in heterogeneous graphs.
3. Static/Dynamic Graphs: Dynamic graphs have varying input features or topology.

The third step is to design a loss function. This step deals with task type and training settings. For graph learning tasks, there are three types of tasks:

1. Node-level: includes node classification, node regression, node clustering, etc.
2. Edge-level: includes edge classification and link prediction, which requires the model to classify edge types or predict if an edge exists between two nodes.
3. Graph-level: includes graph classification, graph regression, and graph matching.

In terms of training, we have three types of training settings:

1. Supervised setting
2. Semi-supervised setting
3. Unsupervised setting

Upon specifying the task type and training setting we can specify the loss function, such as cross-entropy.

In the fourth step, we build the computational module. Some commonly used modules are:

1. Propagation Module: used to propagate information between nodes to capture feature and topological information.
2. Sampling Module: used to conduct propagation on larger graphs.
3. Pooling Module: is used to extract information from nodes to represent high-level subgraphs or graphs.

The above architecture is general to all GNNs. With a combination of the above steps, a typical module is built. Spatial-based GNNs rely on a message propagation and aggregation mechanism between neighboring nodes and creates spectral graph convolutions.

6 Proposed Methodology

This section provides an overview and description of the methodology used in this study and goes through the model structure, training method, and model testing.

6.1 Model Structure

In this study, we evaluate two ensemble models and perform a comparison of their accuracy and efficiency. The models are based on CNN-Bi-LSTM, CNN-Bi-GRU, and on AM. One self-attention layer composed of 1D-CNN query, keys and values first extracts the data features, and then the 1D-CNN layer, which further enhances the features from the original vehicle data, are then used as inputs of the Bi-LSTM and Bi-GRU. Upon extracting the temporal features through the models, the results are then input to a multi-head AM layer, and then the output is calculated.

Figure 28 presents the model structure of the CNN-Bi-LSTM. It consists of 1 to 3 layers of 1D-CNN (average pooling, batch or weight normalization, and Leaky ReLU), 1 layer of Bi-LSTM (Leaky ReLU), multi head attention mechanism with a dense layer, and output layer with a sigmoid function. The model structure of CNN-Bi-GRU is similar to CNN-Bi-LSTM

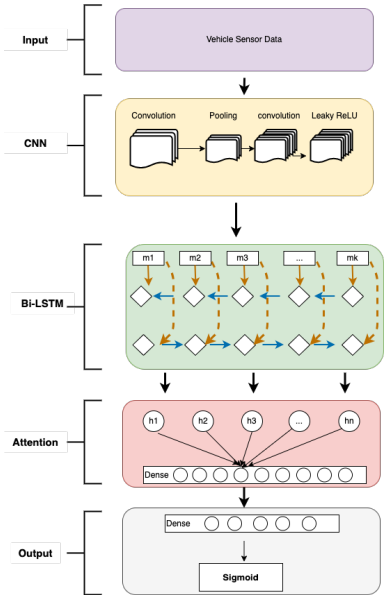


Figure 14: Architecture of CNN-Bi-LSTM Model

and both models use multi head attention mechanism in the style of transformers. A detailed overview of the CNN-Bi-GRU model is presented in Appendix B.

6.2 Model Training

6.2.1 Model Training Background

The study make use of both backpropagation and forward propagation to train the models. The training is an optimization process to minimize the loss and maximize the accuracy and f-score. The parameter set of the proposed model includes hyperparameters, weight parameters, and bias parameters.

6.2.2 Model Training Optimization

The optimization process was conducted using Optuna [40], employing two multi-objective samplers: one based on the TPE (Tree-structured Parzen Estimator) algorithm and another based on the NSGA-III algorithm.

Initially, a single optimization variable, either accuracy or f1-score, was maximized for models based on Bi-GRU or Bi-LSTM, using data obtained from the test dataset. Comparable results were observed for both model types, leading to modifications in the parameter intervals for optimization.

Subsequently, the two optimizations were combined in the subsequent phase. For the final optimization process, two variables, accuracy and f1-score, were simultaneously maximized.

During each trial phase of the optimization process, random examinations were performed on models constructed using Bi-LSTM and Bi-GRU.

The following hyperparameters were selected and utilized to optimize the training process of the model:

- Choice of the optimizer : RAdam, AdamW, and NAdam.
- Optimizer learning rate and weight decay
- Scheduler learning rate: Cosine Annealing Warm Restarts or OneCycleR
- Optimization of the scheduler iterations restart and scheduler learning rate

The hyperparameters selected for optimization and utilized in the construction of the models are as follows:

- Number of 1D-CNN layers from 1 to 3
- Choice of either Bi-GRU or Bi-LSTM
- Hidden size of Bi-GRU and Bi-LSTM
- Number of neuron units for 1D-CNN layers and fully convolutions layers
- Number of strides and kernels for each 1D-CNN layer component
- Drop percentages of Bi-GRU, Bi-LSTM, attention and 1D-CNN layers
- Number of groups for 1D-CNN, which enlarges the with of first attention layer and the next 1D-CNN layers

The building of the deep learning model and most of the hyperparameter components were dynamically generated at runtime to make it easier to adjust the number of neurons in the layers. A summary of the model architecture is presented in Appendix A.

Optimization trial tests were conducted on the datasets obtained through the processing stage:

- Datasets interpolated with zeros
- Interpolated train data with imputed values
- Interpolated train data with imputed values and filtered with the Unscented Kalman filter
- Interpolated train data with imputed values with added 34 extracted features

- Interpolated train data with imputed values and filtered with the Unscented Kalman filter with added 34 extracted features

The filtered dataset produced the best results, hence this dataset was used for the remainder of the optimization procedure.

7 Experimental & Evaluation Results

This section provides an overview of the evaluation metrics used in this study, the hyper-parameter optimization methods, results from CNN-Bi-LSTM, and CNN-Bi-GRU.

7.1 Evaluation Metrics

For the model evaluation, we used *TorchMetrics*, which is a collection of PyTorch metric implementations. These metrics include, but are not limited to, confusion matrix, accuracy, recall, precision, f-score, binary cross-entropy for loss, and Area Under the Receiver Operating Characteristics (AUROC).

The confusion matrix is presented below, where TP, TN, FP, & FN refers to True positive, True negative, False positive, & False negative values.

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

Table 2: The Confusion Matrix

The first metric calculated was accuracy, which means the fraction of predictions our model got right, and it is calculated in the following way:

$$\text{Accuracy } \% = \frac{TP+TN}{TP+TN+FP+FN} \cdot 100 \tag{26}$$

The f-score, which is the measure of the model's accuracy on a dataset, can be calculated from:

$$F_1 = 2 \cdot \frac{precision \times recall}{precision + recall} \tag{27}$$

, given that precision and recall are calculated with:

$$precision = \frac{TP}{TP+FP} \tag{28}$$

$$recall = \frac{TP}{TP+FN} \tag{29}$$

For measuring the loss, the Binary Cross-Entropy loss function between the target and the input probabilities was used. This function compares the predicted class probability to the actual class desired and a score/loss is calculated that penalizes the probability based on how close it is to the expected value. Binary cross-entropy loss is used when adjusting model weights during training and aims to minimize the loss. Binary cross-entropy is defined as:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1+y_i) \cdot log(1-p(y_i)) \tag{30}$$

, where N is the number of training examples, y_i is the label, and $p(y)$ is the probability for ith class.

A ROC curve is a graph showing the performance of a classification model at all classification thresholds. The curve plots the **True positive rate (recall)** on the y-axis, and **False Positive Rate** on the x-axis. The False Positive Rate (FPR) is calculated as:

$$FPR = \frac{FP}{FP+TN} \tag{31}$$

The AUROC measures the entire two-dimensional area underneath the entire ROC curve from $(0,0)$ to $(1,1)$. AUROC is desirable as it measures how well predictions are ranked and the quality of the model's prediction irrespective of the classification threshold chosen [41].

AUC = 0.5	No discrimination
$0.6 \geq$ AUC > 0.5	Poor discrimination
$0.7 \geq$ AUC > 0.6	Acceptable discrimination
$0.8 \geq$ AUC > 0.7	Excellent discrimination
AUC > 0.9	Outstanding discrimination

Table 3: AUROC Performance Interpretation

7.2 Hyperparameter Optimization

All the optimization experiments conducted utilized a batch size of 6 and consisted of 12 training epochs for each trial.

The results from the multiobjective optimization are shown in Figure 15, where objective 0 refers to accuracy and objective 1 refers to f1-score. It can be seen that the attention drop percentage played an important role both for accuracy and f1-score. It is worth mentioning that the drop percentage for CNN and RNN comes second and third in terms of hyperparameter importance.

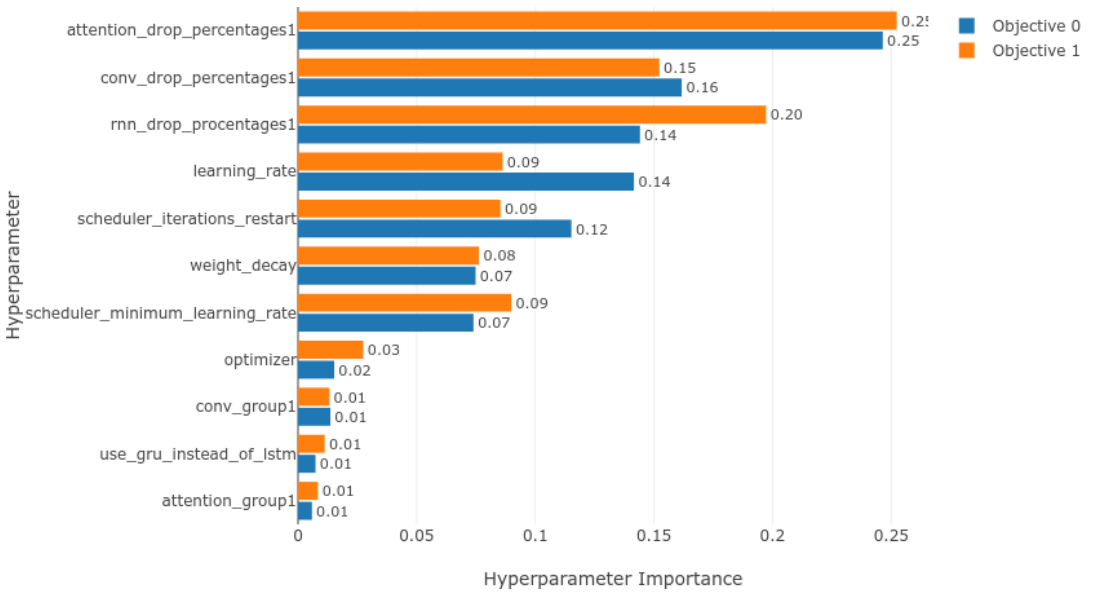


Figure 15: Hyperparameter Importance

In Figure 16, the performance of different optimizers is compared in terms of accuracy and f1-score. Among the various optimizers tested, "NAdam" stands out with the highest values for both accuracy and f1-score.

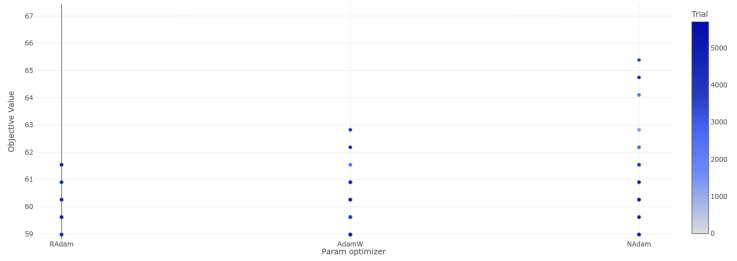
The choice of optimizer plays a crucial role in training deep learning models, as it determines how the model's weights are updated during the learning process. The optimizer's effectiveness directly impacts the model's ability to converge towards an optimal solution and improve its performance on the given task.

The results depicted in Figure 16 indicate that "NAdam" consistently produces the best performance across both accuracy and f1-score metrics. This suggests that the "NAdam" optimizer effectively optimizes the model's parameters, leading to higher accuracy in predicting the correct class labels and achieving a balance between precision and recall as reflected in the f1-score.

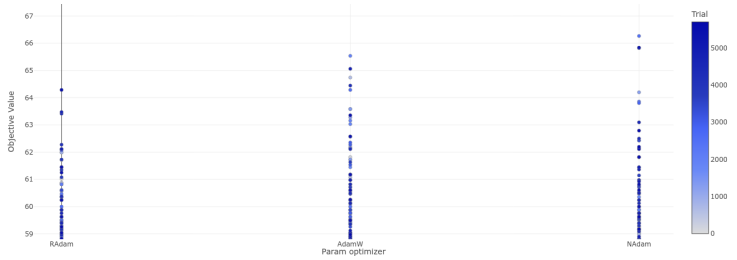
The comparison between LSTM and GRU models reveals distinct patterns in their performance. In terms of f1-score, LSTM demonstrates superior results compared to GRU within a shorter range of optimization trials, as depicted in Figure 17b. This indicates that LSTM quickly adapts and captures complex patterns in the data, resulting in higher f1-score values.

On the other hand, when considering accuracy as the metric, GRU outperforms LSTM, as indicated in Figure 17a. However, achieving higher accuracy values for both models requires a greater number of optimization trials for GRU compared to LSTM. This suggests that GRU benefits from an extended optimization process to reach its peak performance in terms of accuracy.

Overall, these observations highlight the nuanced differences between LSTM and GRU



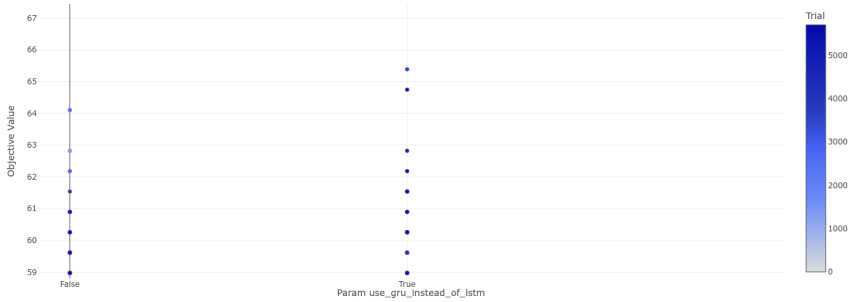
(a) Optimizer Relationship with Accuracy



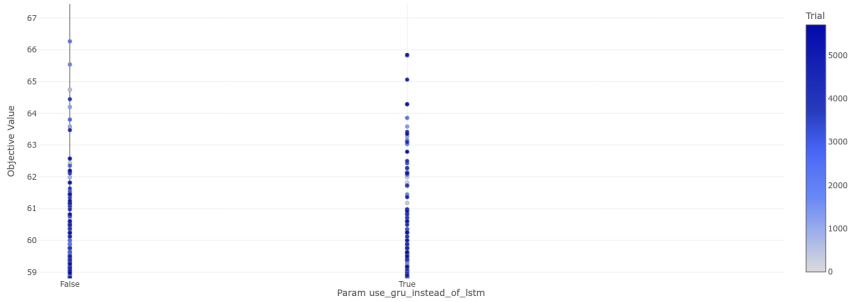
(b) Optimizer Relationship for F1-score

Figure 16: Hyperparameter Relationship of Optimizer

models in terms of their performance on different evaluation metrics and the varying number of optimization trials required to achieve optimal results.



(a) GRU or LSTM Choice Relationship with Accuracy



(b) GRU or LSTM Choice Relationship with F1-score

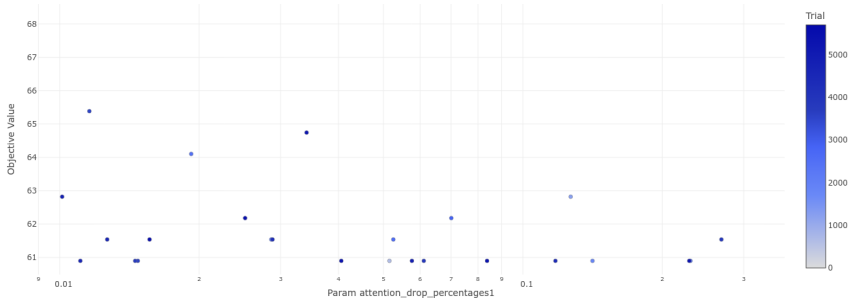
Figure 17: Choice of LSTM or GRU and its Relevance

The attention layer drop percentages were explored within the range of 0.01 to 0.3. Examining the relationship between drop percentages and performance metrics, we observed intriguing patterns. In Figure 18a, it becomes evident that the attention layer's high accuracy is associated with low drop percentages. This suggests that lower dropout rates within the specified interval

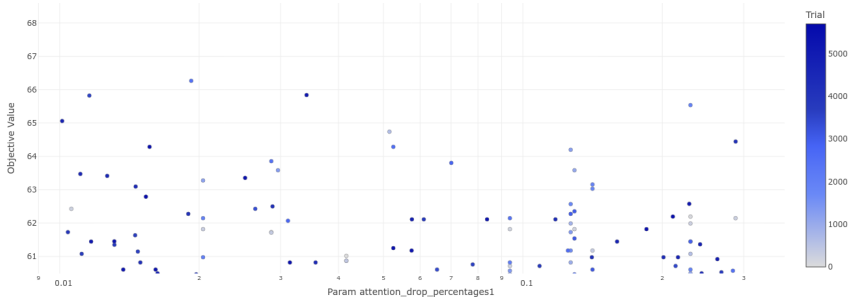
result in improved accuracy for the model when the attention layer is included.

In contrast, when evaluating the f1-score, as depicted in Figure 18b, we found that high f1-score values were achieved across the full range of drop percentages. This implies that the performance of the model in terms of f1-score is less influenced by the specific drop percentage used within the considered interval.

These findings emphasize the importance of selecting an appropriate drop percentage for the attention layer, taking into account the desired performance metric. While high accuracy is primarily achieved with low drop percentages, the f1-score remains consistently high throughout the entire range of drop percentages. These insights provide valuable guidance for optimizing the attention layer in order to maximize the desired performance metric for the model.



(a) Attention Drop Percentages Relationship with Accuracy



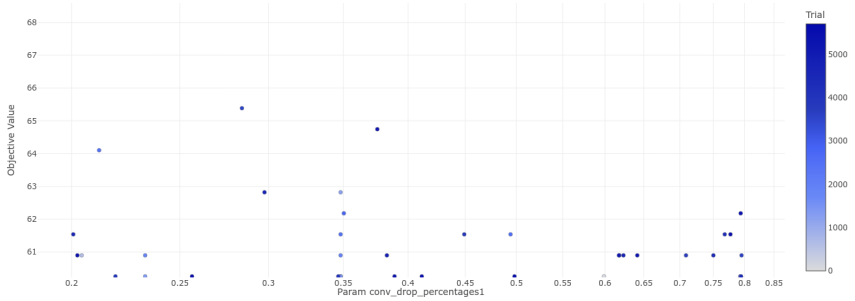
(b) Attention Drop Percentages Relationship with F1-score

Figure 18: Attention Drop Percentages and its Relevance

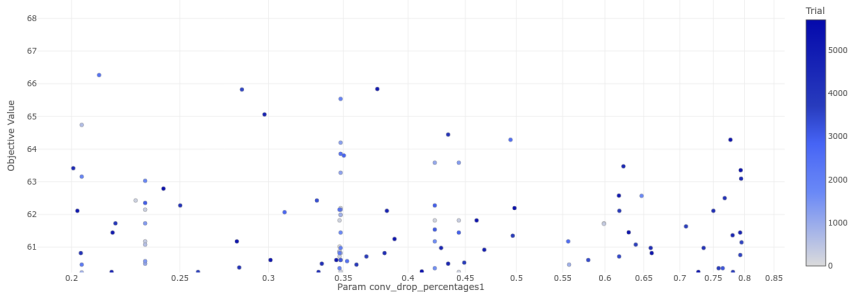
The drop percentages for the convolution layer were explored within the interval of 0.01 to 0.8. Analyzing the relationship between drop percentages and performance metrics, it was observed that high accuracy and f1-score values were primarily achieved for lower values within the specified interval. This indicates that lower drop percentages for the convolution layer led to improved accuracy and f1-score, as depicted in the evaluation results.

It is worth noting that the convolution layer typically had higher drop percentages compared to other layers in the model as shown in Appendix A. This can be attributed to the fact that the convolution layer consists of approximately three convolution modules, which introduce additional complexity and variations to the data. The higher drop percentages in the convolution layer contribute to enhanced regularization and help prevent overfitting by selectively dropping a certain proportion of the connections.

The results highlight the significance of carefully selecting drop percentages for the convolution layer, considering the desired performance metrics. Lower drop percentages within the specified interval proved to be effective in achieving high accuracy and f1-score. The increased drop percentages in the convolution layer underscore the importance of regularization and handling the complexity introduced by multiple convolution modules in the layer.



(a) Convolution Layer Drop Percentages Relationship with Accuracy



(b) Convolution Layer Drop Percentages Relationship with F1-score

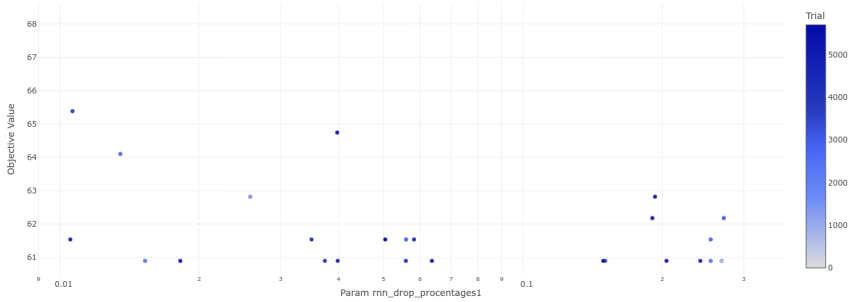
Figure 19: Convolution Layer Drop Percentages and its Relevance

The RNN layer drop percentages were explored within the interval of 0.01 to 0.3. Examining the relationship between drop percentages and performance metrics, we found interesting patterns. In Figure 20a, it is evident that the RNN layer's high accuracy is associated with low drop percentages. This suggests that lower dropout rates within the specified interval result in improved accuracy for the model when the RNN layer is included.

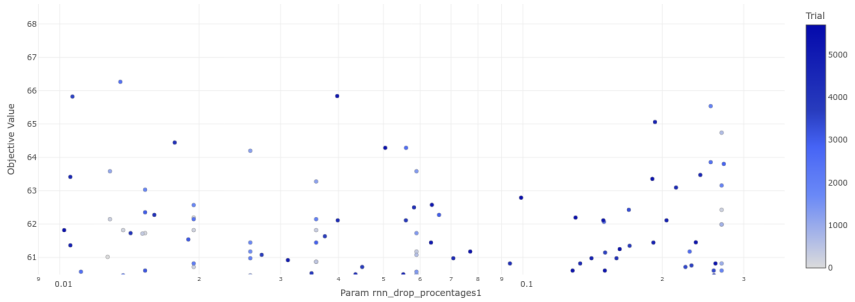
Contrarily, when evaluating the f1-score, as shown in Figure 20b, we observed that high f1-score values were achieved across the full range of drop percentages. This indicates that the performance of the model in terms of f1-score is less influenced by the specific drop percentage used within the considered interval.

These findings indicate that for achieving high accuracy, it is crucial to use low drop percentages in the RNN layer. However, for optimizing the f1-score, a broader range of drop percentages can be explored.

The learning rate, within the interval values of 0.001 to 0.1, consistently demonstrates high scores for both metrics, indicating its effectiveness in training the model as shown in Figure 21. This range of learning rates proves to be optimal for achieving high performance in terms of both accuracy and F1-score.



(a) RNNs layer drop percentages relationship with accuracy



(b) RNNs layer drop percentages relationship with f1-score

Figure 20: RNNs Drop Percentages and its Relevance

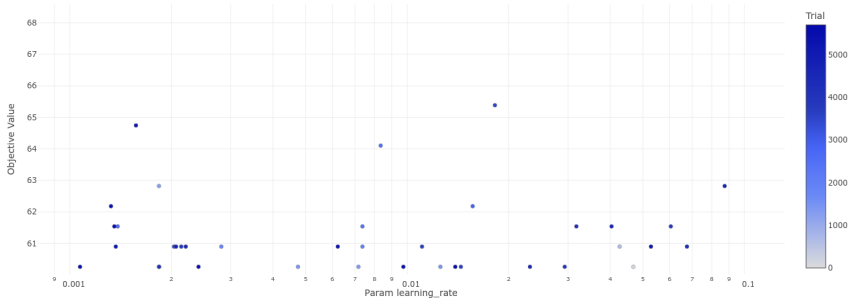
However, when comparing the two metrics, the learning rate tends to yield higher F1-scores compared to accuracy. This suggests that the model's ability to capture and classify positive instances (true positives) is particularly enhanced by the learning rate within the specified interval. The F1-score, which takes into account both precision and recall, reflects the model's performance in balancing these two aspects and achieving a higher overall performance.

The scheduler iterations, when set to low values, consistently produce the highest accuracy and f1-score values as depicted in Figure 22. This indicates that a smaller number of iterations for the scheduler leads to optimal performance for the model.

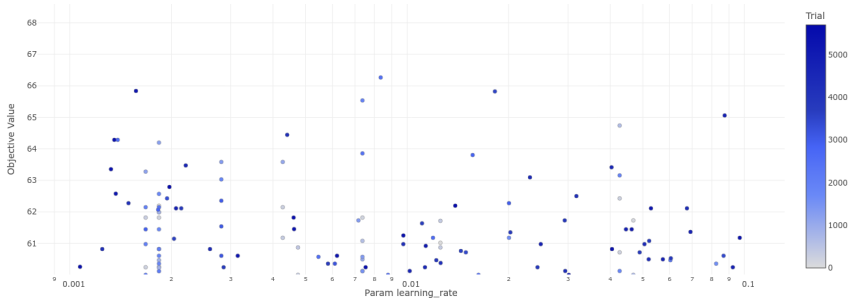
The scheduler iterations play a crucial role in adjusting the learning rate during the training process. When the scheduler is configured with low iteration values, it allows for more frequent updates to the learning rate, enabling the model to adapt and converge faster.

The observed trend of achieving higher accuracy and f1-score values with low scheduler iteration values suggests that more frequent adjustments to the learning rate positively impact the model's ability to optimize its performance. By updating the learning rate more frequently, the model can effectively navigate the training process and converge towards an optimal solution.

A Parallel Coordinate plot is a data visualization technique used to visualize multivariate data. It is particularly useful for understanding the relationship between multiple variables and their impact on a specific outcome. Figure 23, depicts the relationship between different learning rates and their corresponding performance metrics, such as accuracy or loss. Each learning rate is represented by a vertical axis, and the performance metric values are plotted as

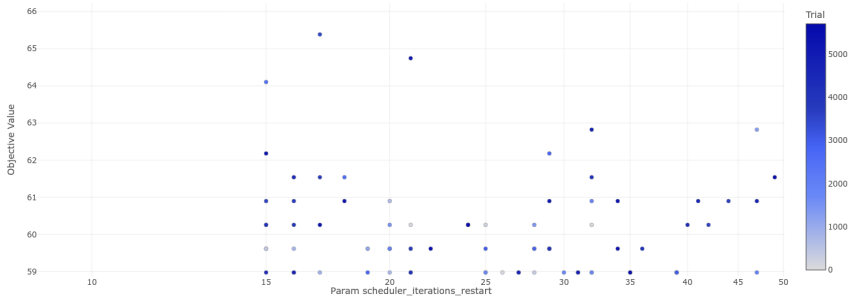


(a) Learning Rate Relationship with Accuracy

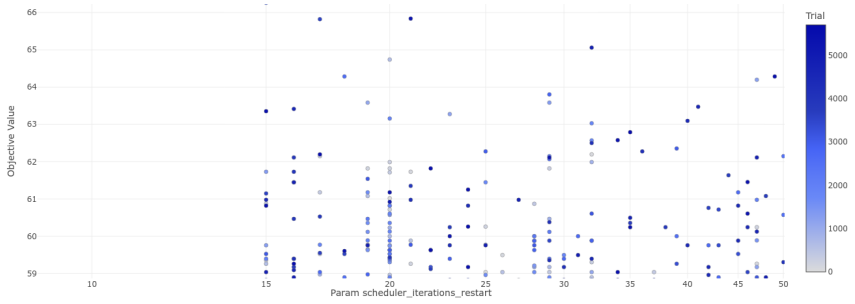


(b) Learning Rate Relationship with F1-score

Figure 21: Learning Rate and its Relevance



(a) Scheduler Iterations Restart Relationship with Accuracy



(b) Scheduler Iterations Restart Relationship with F1-score

Figure 22: Scheduler Iterations Restart and its Relevance

points along each axis.

In Table 4, we present the highest optimization results obtained from all the trials conducted for the deep learning models. Each row in the table represents a specific trial, and the columns display relevant information about the trial's performance.

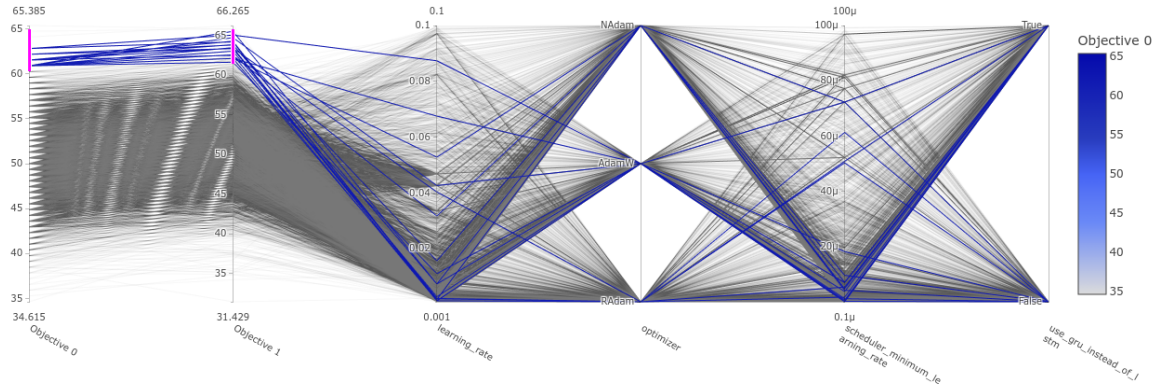


Figure 23: Parallel Coordinate Plot for Learning Rate

The table includes a True/False indicator, indicating whether the deep learning model is based on GRU or LSTM. This allows for easy identification and comparison of the model types.

Upon examining the table, it becomes apparent that the f1-score values consistently surpass the accuracy values. This indicates that the models' ability to capture and classify positive instances (true positives) is notably higher than their overall accuracy. The f1-score provides a more comprehensive evaluation of the models' performance by considering both the true positive rate and the precision.

The higher f1-score values suggest that the models excel in correctly identifying and categorizing positive instances, even if their overall accuracy may be slightly lower. This implies that the models are effective in minimizing false negatives and ensuring a higher level of precision in detecting and classifying specific outcomes.

These findings emphasize the importance of considering both accuracy and f1-score when evaluating the performance of deep learning models. While accuracy provides a general measure of correctness, the f1-score provides a more nuanced assessment, highlighting the models' ability to correctly identify positive instances.

LSTM	GRU	Accuracy	F1-score
False	True	65.38%	65.82%
False	True	64.74%	65.84%
True	False	64.10%	66.27%
True	False	62.82%	64.20%
False	True	62.82%	65.06%
True	False	62.18%	63.80%
False	True	62.18%	63.35%

Table 4: Summary of Best Performances of the Trials Results

The accuracy and f1-score obtained from each optimization experiment, which employed

a learning rate finder optimization framework, exhibited similar performance to the aforementioned results. However, it is important to note that the corresponding loss values were approximately eight times higher. Consequently, the learning rate finder was excluded from the optimization procedure to prevent the inflated loss values from negatively impacting the overall optimization process.

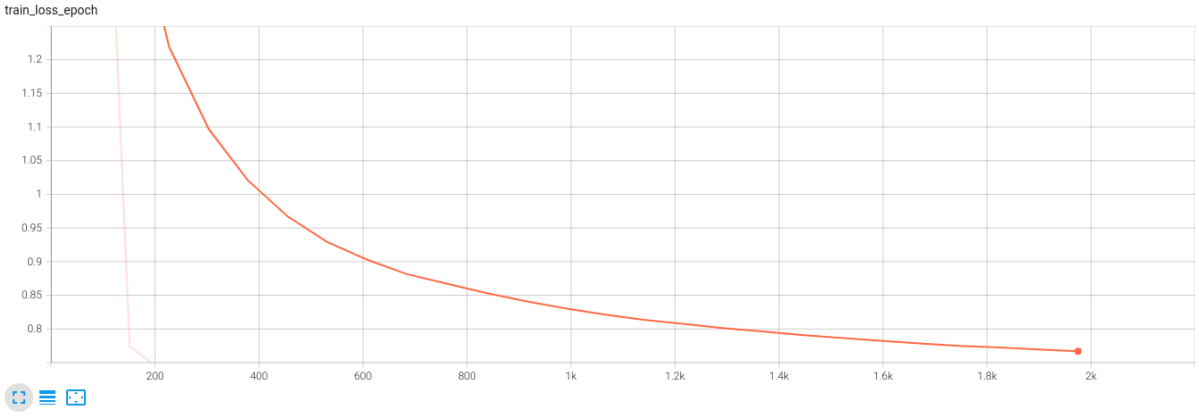


Figure 24: Training Loss Plot

Figure 24, presents the loss of the model that changes over successive epochs during the training phase. The loss function quantifies the disparity between the predicted outputs of the model and the actual ground truth labels.

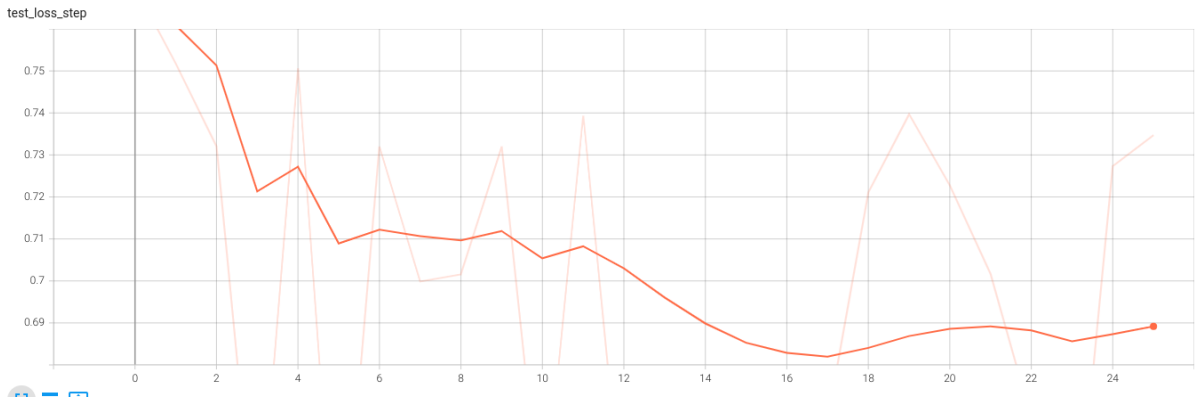


Figure 25: Test Loss Plot

Figure 25, presents how the loss of the model changes over successive epochs during the testing phase. It can be seen that the test and train losses are decreasing.

Figure 26, depicts the accuracy of the model over the epochs. It shows that the model's performance is improving over successive epochs.

Finally, Figure 27, presents the F1-score which is a measure that balances both precision and recall, providing a single value that represents the model's overall performance in correctly classifying the data.

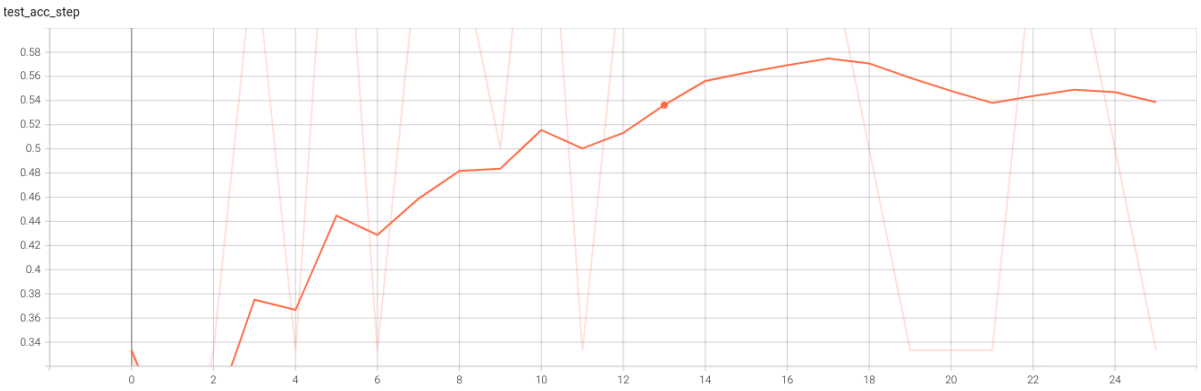


Figure 26: Test Accuracy Plot

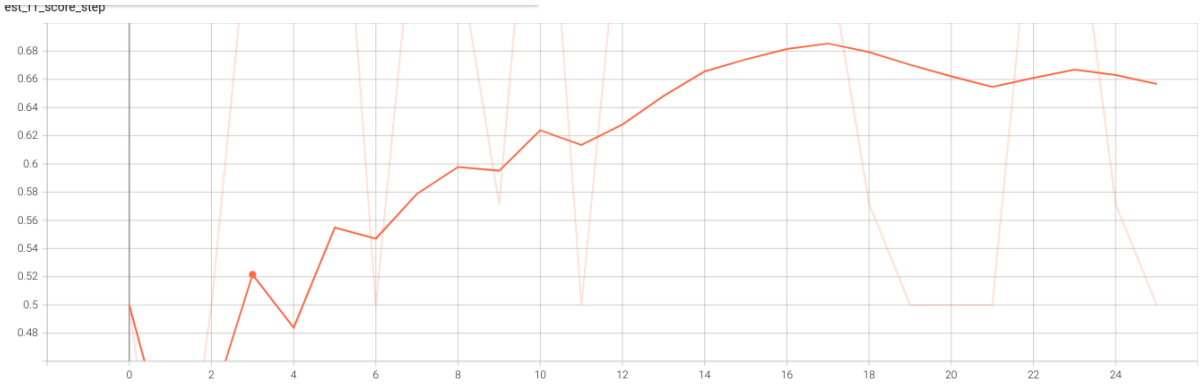


Figure 27: Test F1-score Plot

8 Discussion

This part attempts to present a thorough analysis and interpretation of the findings from the research on deep learning-based automotive defect identification. In this section, the research goals, major discoveries, interpretation of the findings, limitations, and ramifications of the study will all be covered.

This study's goals were to create and assess deep learning models for identifying vehicle faults. The study's specific goal was to find out how well deep learning algorithms like 1D-CNN-Bi-LSTM and 1D-CNN-Bi-GRU work at identifying car defects from sensor data.

8.0.1 RQ1 Discussion

Initially, we conducted preprocessing on the sensor data collected from vehicles to address our research questions. This included performing a normalization process and extracting relevant features. We observed that most sensors had a narrower range of distinct values compared to the length of the dataset. The data distribution remained unaffected even after interpolating missing values in the training dataset. We evaluated models such as 1D-CNN, Bi-RNN, and multi-head

attention on the dataset. However, the inclusion of multi-head attention after Bi-RNN hindered the training process. To overcome this issue, we successfully resumed the training process by removing the attention layer and transferring it to the first layer. We achieved satisfactory results with low dropout percentages for the attention layer across different tiers.

In terms of performance improvement, we explored the use of Unscented Kalman filtering on the training dataset without fine-tuning its parameters. While this approach raised the expectations for model performance, there is room for further improvement in this area. The literature suggests that wavelet filtering could be a superior alternative for sensor filtering. Additionally, we explored the application of a temporal convolution neural network as a feature extraction model for time series prediction. Although the feature extraction method did not perform as well as anticipated, we believe that the temporal convolution neural network warrants further investigation in future experiments.

During the feature extraction procedure, we obtained 34 features, which were utilized for both model training and computing the k-nearest neighbors between vehicles to convert the dataset into a graph dataset. However, due to limitations in memory and processing speed, this conversion process encountered difficulties, leading to a halt in the processing.

Overall, while we encountered challenges and limitations in certain aspects of the methodology, our study has laid the foundation for further exploration and improvement in the future.

8.0.2 RQ2 Discussion

By utilizing the dataset, we conducted training and evaluation of various deep learning architectures, specifically focusing on CNNs and RNNs. Our study demonstrated that these selected deep learning models effectively captured intricate patterns present in the sensor data, enabling successful detection and classification of defects.

The obtained results from the models were highly encouraging, and we further improved their functionality through optimization strategies. For optimization, we employed the single-objective Optuna optimization technique, which primarily supports a single optimization measure. Additionally, we explored multi-objective optimization, involving the utilization of multiple optimization metrics such as accuracy, f1-score, and loss.

During the optimization process, the models underwent optimization with one to three layers. It is worth mentioning that the models produced through an ensemble of models might benefit from being single-layer models, indicating potential performance advantages in certain scenarios.

Throughout the optimization, we observed an increase in the accuracy value, although it progressed at a slower pace compared to the f1-score. LSTM and GRU achieved comparable results, with GRU exhibiting superior performance after more extensive optimization trial ex-

periments. Notably, among the training optimizers considered, nAdam emerged as the only suitable choice based on the optimization outcomes.

8.0.3 Improvements Discussion

The deep learning models' capacity to autonomously learn hierarchical representations from the input data is a contributor to their excellent performance. The models reliably extracted features and identified faults by utilizing deep architectures and convolutions layers to capture local and global dependencies in the sensor data. The modeling of temporal dependencies and the identification of changing defects were made easier by the recurrent layers in RNNs.

The deep learning models are able to evaluate and categorize complicated patterns in sensor data for fault detection, as shown by their accuracy, f1-score, and performance. The models could detect slight irregularities and tell apart healthy from unhealthy vehicle behavior. This shows that deep learning techniques could improve car diagnostics and help with preventative maintenance plans, which would decrease downtime and improve safety.

Despite the positive outcomes, this thesis has several drawbacks that need to be taken into account. Firstly, the quantity and quality of training data have a significant impact on how well deep learning models perform. A little or unbalanced dataset had an impact on how generalizable the models were. The models' capacity to identify uncommon or previously undiscovered fault types was also constrained by the data's availability. Additionally, because the dataset was anonymised, we were unable to comprehend the features and draw information from the raw data.

Another limitation to consider is the dependency on sensor data for defect detection, particularly when the data does not encompass the full range of values. Although sensor data provides valuable insights, the inclusion of additional contextual information, such as maintenance history or driving habits, has the potential to enhance the precision and reliability of the defect detection system. By incorporating these supplementary data sources, a more comprehensive understanding of the vehicle's condition can be achieved, enabling more accurate and robust defect detection capabilities.

Initially, this thesis aimed to combine deep learning models with knowledge base graphs. However, implementing knowledge base graphs for vehicle fault detection can present certain challenges. One major hurdle is the availability and accessibility of comprehensive and reliable domain-specific knowledge. Constructing a knowledge base graph requires a substantial amount of data, including fault descriptions, symptoms, diagnostic procedures, and repair recommendations. However, obtaining such information for a wide range of vehicle faults can be time-consuming and challenging due to the complexity and diversity of vehicle systems. Additionally, keeping the knowledge base graph up-to-date with the constantly evolving vehicle technologies and diagnostic procedures requires continuous effort and resources. Therefore, the

lack of a robust and readily available knowledge base poses a significant obstacle in implementing knowledge base graphs for vehicle fault detection.

Additionally, using anonymous datasets may make it more difficult to create knowledge base graphs for car problem detection. While anonymized datasets are essential for protecting privacy and confidentiality, they frequently lack the exact and granular data required for building a thorough knowledge base. To build a solid and reliable knowledge base graph, one needs access to identifying vehicle data, such as vehicle make, model, and specific defect information. The fine-grained information required to build connections between problems, symptoms, and diagnostic techniques is removed during the anonymization process. As a result, the knowledge base might not have the breadth and depth necessary to accurately identify and diagnose vehicle defects. Although anonymization is necessary for compliance with privacy laws and ethical reasons, it restricts the possibility for leveraging detailed data in constructing knowledge base graphs.

9 Ethical Considerations

This section will go through the ethical aspects of this study and show that the researchers were aware of the ethical considerations about the data and the study.

Ethical considerations surrounding sensor data of vehicles are of significant importance in the era of connected cars and smart transportation systems. Sensor data collected from vehicles, such as GPS location, speed, acceleration, braking, and other driving behaviors, can provide valuable insights for various purposes, including traffic management, road safety, and vehicle performance optimization. However, it is crucial to handle this data with utmost care and adhere to ethical principles. First and foremost, data privacy and consent are paramount. Individuals must be informed about the types of data being collected, the purpose of its collection, and have the ability to provide informed consent for its use. Transparency and clear communication are crucial to maintaining trust and ensuring that individuals have control over their data. Furthermore, anonymization techniques should be applied rigorously to remove personally identifiable information, ensuring that individual drivers cannot be identified. Ethical considerations surrounding anonymized datasets are crucial in today's data-driven world. While anonymization is often employed as a means to protect privacy and ensure data confidentiality, it is important to recognize that complete anonymization is becoming increasingly challenging to achieve due to the growing availability of sophisticated re-identification techniques. Nevertheless, it is essential to recognize that sensor data, when combined with other datasets or analyzed comprehensively, can still reveal sensitive information or lead to the identification of individuals. As such, thorough risk assessments should be conducted to evaluate the potential for re-identification and the associated privacy risks.

Another ethical consideration is data bias. Anonymized datasets may still retain inherent

biases that were present in the original data. This can be a result of sampling bias, data collection methods, or the biases of individuals involved in data processing. Careful examination of potential biases should be undertaken to ensure that any analysis or decision-making based on anonymized data does not perpetuate discrimination or unfair treatment. Data governance frameworks and regulations should be established to ensure compliance with ethical standards, promote fair and unbiased data usage, and address potential societal implications. Responsible handling of sensor data of vehicles requires a balance between the benefits of data-driven insights and the protection of individual privacy, security, and public trust.

10 Conclusion

In conclusion, deep learning-based vehicle defect identification has enormous potential to revolutionize automobile maintenance and improve traffic safety and efficiency.

The capacity of deep learning algorithms to analyze complex sensor data and automatically identify patterns associated with various faults can significantly increase the efficiency and accuracy of fault detection systems. Notably, techniques such as 1D-CNN for feature extraction, along with Bi-LSTM and Bi-GRU for capturing time dependencies, demonstrate the capacity to effectively process complex automotive data. Deep learning-powered automotive systems can proactively identify and fix issues, resulting in better maintenance procedures, less downtime, and greater overall vehicle performance. These techniques allow for the creation of resilient and intelligent fault detection systems that are able to identify anomalies, foresee problems, and initiate prompt maintenance actions. However, ethical issues including data privacy, openness, and justice must be prioritized in its development. Respecting individual privacy rights requires making sure that the data used to train deep learning models is appropriately anonymized, with permission, and preserved. Furthermore, maintaining public confidence and preventing an overreliance on automated systems require open information about the strengths and weaknesses of deep learning algorithms.

Addressing possible biases in both data and algorithms is crucial to mitigate any biased effects. Furthermore, integrating deep learning techniques into vehicle defect detection systems has the potential to enhance security, reduce maintenance costs, and enhance overall vehicle reliability. This advancement paves the way for a smarter and more efficient transportation ecosystem while upholding ethical principles and ensuring the responsible deployment of deep learning technologies.

10.1 Future Work

There are still many opportunities for more study and improvement in car defect identification, despite the substantial advancements made utilizing deep learning techniques. Here, we identify

some prospective areas for more research and development:

1. Deep learning model generalization must be improved for use in practical applications. The development of methods that allow models to accommodate changes in vehicle makes and models, climatic circumstances, and sensor setups can be the subject of future study. To improve the models' capacity for fault detection in a variety of circumstances, this may employ transfer learning, domain adaptation, or multi-task learning techniques.

2. To increase the precision and effectiveness of deep learning algorithms for fault detection, additional research into model optimization methods can be explored. The efficiency and generalization of the models can be improved by investigating novel architectures, hyperparameter tuning, and regularization techniques.

3. The ability to capture temporal dependencies in vehicle sensor data can be useful for deep learning models. Recurrent neural networks (RNNs) or transformers can be combined in future work to more effectively describe long-term dependencies and temporal patterns, enabling more precise and dynamic defect detection.

4. Deep learning models frequently lack interpretability, making it difficult to comprehend how they make decisions. The development of methods to offer justifications or visual representations for the found defects should be the main focus of research. The incorporation of deep learning models into real-time diagnostic systems can be made easier and can increase trust.

5. Deep learning model training is generally difficult since labeled defect data is frequently scarce. Future research can investigate methods for data enhancement and synthesis to provide synthetic failure data, enabling larger and more varied training sets. Generative adversarial networks (GANs) or other data augmentation techniques tailored to defect detection tasks can be used in this regard.

6. For prompt maintenance interventions and probable failure prevention, real-time fault detection is essential. As additional sensor data becomes accessible, future research can concentrate on creating online learning algorithms that can continually update and change the model's parameters, enabling real-time issue identification without the need to retrain the entire model.

7. Integrating fault detection systems with autonomous driving systems is crucial as the use of driverless vehicles increases. Future research might investigate how to combine autonomous decision-making frameworks with deep learning-based fault detection algorithms to provide proactive defect detection and reaction in autonomous cars, improving safety and dependability.

8. To overcome this limitation, it is required to investigate several approaches for knowledge acquisition and inference, such as applying expert knowledge, integrating other databases, or using cutting-edge machine learning algorithms to make up for the absence of explicit data. Future research should focus on developing unique methods for locating vehicle flaws that strike a compromise between data privacy and effective knowledge base construction.

Aiming to improve model generalization and optimization, incorporate temporal information, enhance explainability, investigate data augmentation techniques, enable real-time detection, and integrate with autonomous systems are all objectives for future research in vehicle fault detection using deep learning methods. We can progress the field and aid in the creation of reliable, effective, and trustworthy defect detection systems for the automobile sector by tackling these issues.

References

[1] F. Arena, M. Collotta, L. Luca, M. Ruggieri, and F. G. Termine, "Predictive maintenance in the automotive sector: A literature review," *Mathematical and Computational Applications*, vol. 27, no. 1, p. 2, 2022.

[2] D. van Schrick, "Remarks on terminology in the field of supervision, fault detection and diagnosis," *IFAC Proceedings Volumes*, vol. 30, no. 18, pp. 959–964, 1997.

[3] F. Van Wyk, Y. Wang, A. Khojandi, and N. Masoud, "Real-time sensor anomaly detection and identification in automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1264–1276, 2019.

[4] C.-S. A. Gong, C.-H. S. Su, Y.-H. Chen, and D.-Y. Guu, "How to implement automotive fault diagnosis using artificial intelligence scheme," *Micromachines*, vol. 13, no. 9, p. 1380, 2022.

[5] J. Z. Varghese, R. G. Boone, *et al.*, "Overview of autonomous vehicle sensors and systems," in *International Conference on Operations Excellence and Service Engineering*, sn, 2015, pp. 178–191.

[6] R. Khoshkangini, P. Sheikholharam Mashhadi, P. Berck, *et al.*, "Early prediction of quality issues in automotive modern industry," *Information*, vol. 11, no. 7, p. 354, 2020.

[7] R. Khoshkangini, M. Tajgardan, J. Lundström, M. Rabbani, and D. Tegnered, "A snapshot-stacked ensemble and optimization approach for vehicle breakdown," *Available at SSRN 4342325*,

[8] Y. Fang, H. Min, W. Wang, Z. Xu, and X. Zhao, "A fault detection and diagnosis system for autonomous vehicles based on hybrid approaches," *IEEE Sensors Journal*, vol. 20, no. 16, pp. 9359–9371, 2020.

[9] T. Berghout and M. Benbouzid, "A systematic guide for predicting remaining useful life with machine learning," *Electronics*, vol. 11, no. 7, p. 1125, 2022.

[10] C. Ferreira and G. Gonçalves, "Remaining useful life prediction and challenges: A literature review on the use of machine learning methods," *Journal of Manufacturing Systems*, vol. 63, pp. 550–562, 2022.

[11] G. Chassagnon, M. Vakalopolou, N. Paragios, and M.-P. Revel, "Deep learning: Definition and perspectives for thoracic imaging," *European radiology*, vol. 30, pp. 2021–2030, 2020.

[12] L. Tian, X. Zhou, Y.-P. Wu, W.-T. Zhou, J.-H. Zhang, and T.-S. Zhang, "Knowledge graph and knowledge reasoning: A systematic review," *Journal of Electronic Science and Technology*, p. 100 159, 2022.

[13] J. P. Nieto González, "Vehicle fault detection and diagnosis combining an aann and multiclass svm," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 12, no. 1, pp. 273–279, 2018.

[14] O. Serradilla, E. Zugasti, J. Rodriguez, and U. Zurutuza, "Deep learning models for predictive maintenance: A survey, comparison, challenges and prospects," *Applied Intelligence*, vol. 52, no. 10, pp. 10 934–10 964, 2022.

[15] Z. Lai, M. Liu, Y. Pan, and D. Chen, "Multi-dimensional self attention based approach for remaining useful life estimation," *arXiv preprint arXiv:2212.05772*, 2022.

[16] J. Zhang, Y. Jiang, S. Wu, X. Li, H. Luo, and S. Yin, "Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism," *Reliability Engineering & System Safety*, vol. 221, p. 108 297, 2022.

[17] R. Jin, Z. Chen, K. Wu, M. Wu, X. Li, and R. Yan, "Bi-lstm-based two-stream network for machine remaining useful life prediction," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.

[18] K. G. Eknath and G. Diwakar, "Prediction of remaining useful life of rolling bearing using hybrid dcnn-bigru model," *Journal of Vibration Engineering & Technologies*, pp. 1–14, 2022.

[19] R. Gong, J. Li, and C. Wang, "Remaining useful life prediction based on multisensor fusion and attention tcn-bigru model," *IEEE Sensors Journal*, vol. 22, no. 21, pp. 21 101–21 110, 2022.

[20] T. Sun and Q. Wang, "Multi-source fault detection and diagnosis based on multi-level knowledge graph and bayesian theory reasoning (s).," in *SEKE*, 2019, pp. 177–232.

[21] W. Du, D. Côté, and Y. Liu, "Saits: Self-attention-based imputation for time series," *Expert Systems with Applications*, vol. 219, p. 119 619, 2023.

[22] "Https://github.com/drsasanbarak/metats," 2023.

[23] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[24] Z. Chen, J. Xu, C. Alippi, *et al.*, "Graph neural network-based fault diagnosis: A review," p. 17, 2021.

[25] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[26] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-d convolutional neural networks for signal processing applications," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8360–8364.

[27] S. Kiranyaz, A. Gastli, L. Ben-Brahim, N. Al-Emadi, and M. Gabbouj, "Real-time fault detection and identification for mmc using 1-d convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8760–8771, 2018.

[28] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-d convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7067–7075, 2016.

[29] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, pp. 5929–5955, 2020.

[30] O. Ö. Ersöz, A. F. İnal, A. Aktepe, A. K. Türker, and S. Ersöz, "A systematic literature review of the predictive maintenance from transportation systems aspect," *Sustainability*, vol. 14(21), p. 14 536, 2022.

[31] J. Lei, C. Liu, and D. Jiang, "Fault diagnosis of wind turbine based on long short-term memory networks," *Renewable energy*, vol. 133, pp. 422–432, 2019.

[32] H. Sun, M. Chen, J. Weng, Z. Liu, and G. Geng, "Anomaly detection for in-vehicle network using cnn-lstm with attention mechanism," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 880–10 893, 2021.

[33] Q. Li, R. Cheng, and H. Ge, "Short-term vehicle speed prediction based on bilstm-gru model considering driver heterogeneity," *Physica A: Statistical Mechanics and its Applications*, vol. 610, p. 128 410, 2023.

[34] Q. Wang, C. Xu, Y. Zhou, T. Ruan, D. Gao, and P. He, "An attention-based bi-gru-capsnet model for hypernymy detection between compound entities," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2018, pp. 1031–1035.

[35] X. Ran, Z. Shan, Y. Fang, and C. Lin, "An lstm-based method with attention mechanism for travel time prediction," *Sensors*, vol. 19, no. 4, p. 861, 2019.

[36] L. Xiang, P. Wang, X. Yang, A. Hu, and H. Su, "Fault detection of wind turbine based on scada data analysis using cnn and lstm with attention mechanism," *Measurement*, vol. 175, p. 109 094, 2021.

[37] H. Lv, J. Chen, T. Pan, T. Zhang, Y. Feng, and S. Liu, "Attention mechanism in intelligent fault diagnosis of machinery: A review of technique and application," *Measurement*, vol. 199, 2022.

[38] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[39] L. Shao, Y. Duan, X. Sun, H. Gao, D. Zhu, W. Miao, *et al.*, "Answering who/when, what, how, why through constructing data graph, information graph, knowledge graph and wisdom graph.," in *SEKE*, 2017, pp. 1–6.

[40] "Https://optuna.org/," 2023.

[41] S. Yang and G. Berdine, "The receiver operating characteristic (roc) curve," *The Southwest Respiratory and Critical Care Chronicles*, vol. 5, no. 19, pp. 34–36, 2017.

Acronyms

1D 1-Dimentional. i, 6, 14–16, 22, 24, 35, 39

2D 2-Dimentional. 14

AANN Auto-associative Neural Network. 4

AM Attention Mechanism. v, 18–20, 22

ANN Artificial Neural Network. 14

AUROC Area Under the Receiver Operating Characteristics. vi, 25, 26

Bi-GRU Bi-directional Gated Recurrent Unit. i, 5, 6, 14, 18, 22–25, 35, 39

Bi-LSTM Bi-Directional Long Short-term Memory. i, 6, 14, 17, 22–25, 35, 39

Bi-RNN Bi-Directional Recurrent Neural Networks. 35, 36

BP Back Propagation. 15, 16

CNN Convolutional Neural Networks. i, 4–6, 14–16, 21–26, 35, 36, 39

DCNN Deep Convolutional Neural Network. 5

FP Forward Propagation. 14, 16

GAE Graph Auto-Encoder. 14

GNN Graph Neural Network. 6, 14, 21, 22

GPS Global Positioning System. 2

GRU Gated Recurrent Unit. 4, 5, 18–20, 27, 33, 36

KNN K-Nearest Neighbor. 14

LIDAR Laser Range Finder System. 2

LSTM Long Short-term Memory. 4, 5, 16–20, 27, 33, 36

LVD Logged Vehicle Data. 2

MLP Multilayer Perceptron. 5, 14, 15

MSE Mean-Square-Error. 15

NaN Not a Number. 7, 10

NLPCA Non-Linear Principle Component Analysis. 4

RAdam Rectified Adam. 24

RADAR Radio Detection and Ranging. 2

RNN Recurrent Neural Networks. 4, 5, 16–18, 26, 30, 36, 37

RUL Remaining Useful Life. 5

SVM Support Vector Machine. 2, 4

TCN Temporal Convolutional Neural Network. 5, 13

V2V Vehicle to Vehicle. 2

WCD Warranty Claim Data. 2

A Appendix A

```
model to optimize:  NetModel(
  (att_layer1): AttentionLayer(
    (inner_attention): AttentionBlock(
      (dropout): Dropout(p=0.09241173230707932, inplace=False)
    )
    (query_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (key_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (value_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (out_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (activation_total): LeakyReLU(negative_slope=0.01)
  )
  (conv_layer1): ConvLayer(
    (dropout): Dropout(p=0.31607514729669495, inplace=False)
    (downConv): Conv1d(136, 136, kernel_size=(3,), stride=(2,),
    groups=2)
    (activation1): GELU(approximate='none')
    (actConv): Conv1d(136, 48, kernel_size=(3,), stride=(1,),
    groups=2)
    (activation2): GELU(approximate='none')
    (sampleConv): Conv1d(136, 48, kernel_size=(1,), stride=(1,)
    ,groups=2)
    (pool): MaxPool1d(kernel_size=3, stride=2, padding=1)
    (activation_total): LeakyReLU(negative_slope=0.01)
  )
  (bi_lstm): LSTM(48, 96, num_layers=2, batch_first=True,
  dropout=0.05058167360629957, bidirectional=True)
)

model to optimize:  NetModel(
  (att_layer1): AttentionLayer(
    (inner_attention): AttentionBlock(
      (dropout): Dropout(p=0.04586208255895992, inplace=False)
    )
    (query_projection): Conv1d(52, 52, kernel_size=(1,),
```

```
    stride=(1,))
    (key_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (value_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (out_projection): Conv1d(52, 52, kernel_size=(1,),
    stride=(1,))
    (activation_total): LeakyReLU(negative_slope=0.01)
  )
  (conv_layer1): ConvLayer(
    (dropout): Dropout(p=0.5585691604009321, inplace=False)
    (downConv): Conv1d(136, 136, kernel_size=(3,), stride=(2,))
    (activation1): GELU(approximate='none')
    (actConv): Conv1d(136, 35, kernel_size=(3,), stride=(1,))
    (activation2): GELU(approximate='none')
    (sampleConv): Conv1d(136, 35, kernel_size=(1,),
    stride=(1,))
    (pool): MaxPool1d(kernel_size=3, stride=2, padding=1,
    dilation=1)
    (activation_total): LeakyReLU(negative_slope=0.01)
  )
  (bi_gru): GRU(35, 70, num_layers=2, batch_first=True,
            dropout=0.24621989235577188, bidirectional=True)
```

B Appendix B

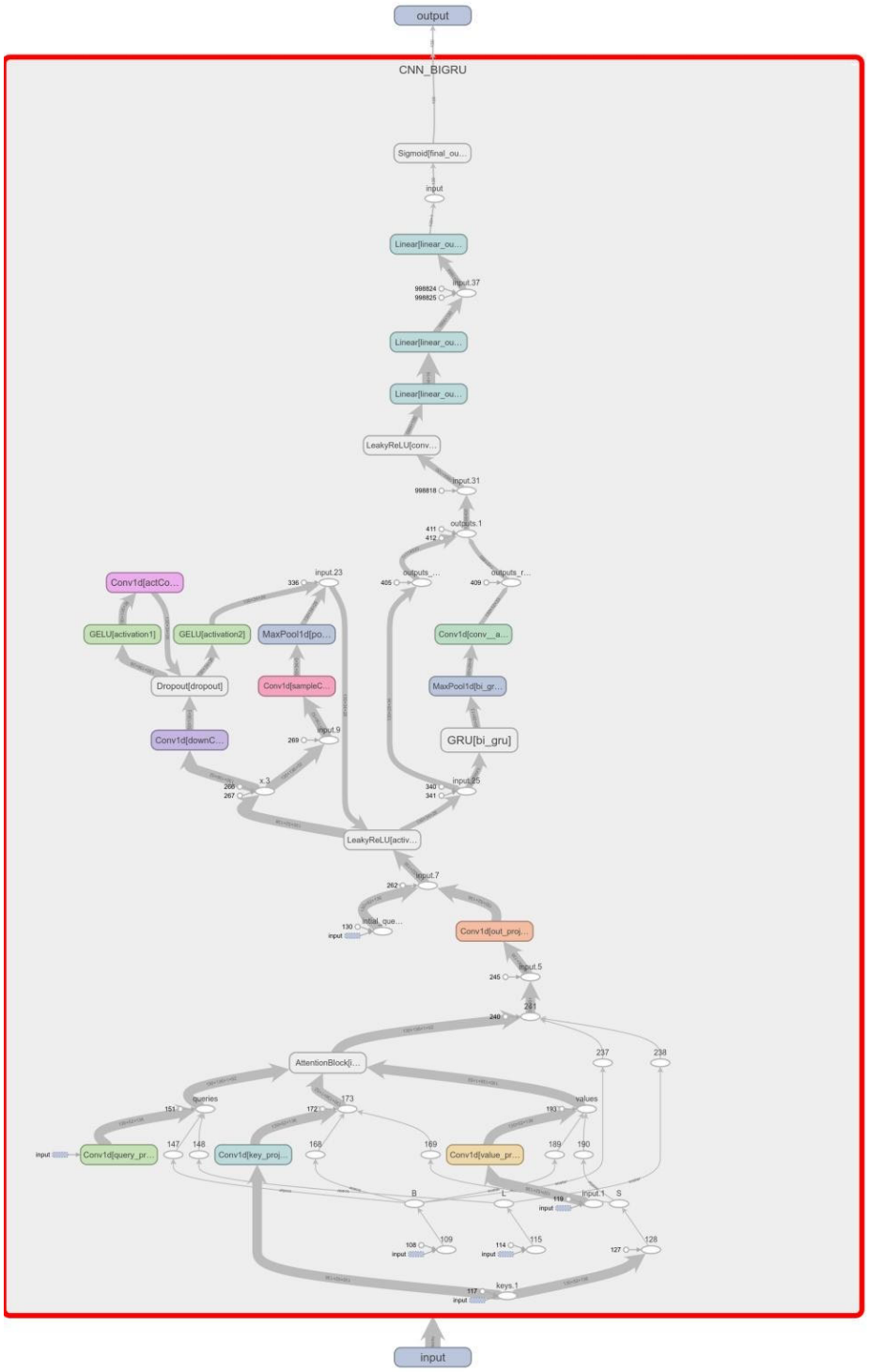


Figure 28: Process of CNN-Bi-GRU Model

Declaration of Authorship

A Deep learning Approach to vehicle fault detection based on vehicle behavior – Master Thesis

We hereby declare that this thesis is entirely our own work and without the use of any unauthorised assistance. Any content which has been taken verbatim or paraphrased from other sources has been identified as such. This paper has not been submitted in any form whatsoever to another examining body.

We agree that the present work may be verified with an anti-plagiarism software.

Malmö, Sweden, June 21, 2023
Rafi Khaliqi & Iulian Cozma