T.C.
ISTANBUL OKAN UNIVERSITY
INSTITUTE OF GRADUATE SCIENCES

THESIS

FOR THE DEGREE OF

MASTER OF SCIENCE

IN AUTOMOTIVE MECHATRONICS AND

INTELLIGENT VEHICLES PROGRAM

Yousif ALARAJI

(203005012)

AN INVESTIGATION INTO VIBRATION ANALYSIS FOR DETECTING FAULTS IN VEHICLE STEERING OUTER TIE-ROD

THESIS ADVISOR

Assist. Prof. Dr. Sina ALP

ISTANBUL, February 2024

ABSTRACT

This thesis investigates fault detection in a simulated car's gear steering system using MSC Adams and MATLAB simulations. It focuses on analyzing the angular acceleration signal from the outer tie rod, crucial for identifying faults. Through simulation, the car records the angular acceleration signal as a baseline for healthy signals and introduces simulated wear into the tie rod using MATLAB to mimic real-world faults. Various types of noise are added to the signals to assess the system's robustness. Two feature extraction methods, wavelet scattering and discrete wavelet transform, are evaluated for their effectiveness. Classification employs Support Vector Machines (SVM) and Neural Networks (NN) and aims to classify signals as normal or faulty and determine fault severity. Findings suggest wavelet scattering with Long Short-Term Memory (LSTM) Neural Networks as a stable approach. Techniques like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Recursive Feature Elimination (RFE) enhance classification accuracy. This research significantly advances fault detection in automotive systems, providing insights into signal processing, classification algorithms, optimization, and feature selection. The developed fault detection system promises real-world application, potentially enhancing steering system reliability and safety.

Keywords: Fault Detection; Steering Systems; Angular Acceleration; Simulation; Wavelet Analysis

ÖZET

Bu tez, MSC Adams ve MATLAB simülasyonlarını kullanarak bir simüle edilmiş aracın direksiyon sistemindeki hata tespitini araştırmaktadır. Dış bağlantı çubuğundan gelen açısal ivme sinyalini analiz etmeye odaklanarak hataları belirlemek için önemlidir. Simülasyon yoluyla, araç sağlıklı sinyaller için bir temel olarak açısal ivme sinyalini kaydeder ve MATLAB kullanarak gerçek dünya hatalarını taklit etmek için dış bağlantı çubuğuna simüle edilmiş aşınma ekler. Sistemin dayanıklılığını değerlendirmek için sinyallere çeşitli tiplerde gürültü eklenir. Dalgalet dağılması ve kesikli dalgalet dönüşümü olmak üzere iki özellik çıkarma yöntemi etkinlikleri açısından değerlendirilir. Sınıflandırma, Destek Vektör Makineleri (SVM) ve Sinir Ağları (NN) kullanır ve sinyalleri normal veya hatalı olarak sınıflandırmayı ve hata ciddiyetini belirlemeyi amaçlar. Bulgular, dalgakıran dağılmasıyla Uzun Kısa Vadeli Hafıza (LSTM) Sinir Ağları'nın istikrarlı bir yaklaşım olduğunu öne sürmektedir. Temel Bileşen Analizi (PCA), Doğrusal Ayırıcı Analiz (LDA) ve Tekrarlanan Özellik Eleme (RFE) gibi teknikler, sınıflandırma doğruluğunu artırır. Bu araştırma, otomotiv sistemlerinde hata tespitini önemli ölçüde ilerletmekte olup, sinyal işleme, sınıflandırma algoritmaları, optimizasyon ve özellik seçimi konularında içgörüler sunmaktadır. Geliştirilen hata tespit sistemi, direksiyon sistemi güvenilirliğini ve güvenliğini potansiyel olarak artırarak gerçek dünya uygulaması vadetmektedir.

Anahtar kelimeler: Hata Tespiti; Direksiyon Sistemleri; Açısal İvme; Simülasyon; Dalgalet Analizi

ACKNOWLEDGMENT

I am immensely grateful to Dr. Sina ALP for his unwavering guidance, support, and encouragement throughout the entirety of this research. His expertise, profound knowledge, and insightful feedback have been pivotal in shaping the direction and quality of this study. I am truly fortunate to have had the opportunity to work under his supervision.

I would also like to extend my heartfelt appreciation to Dr. Rami KHUSHABA for his valuable contributions and assistance during this research endeavor. His expertise and thoughtful input have significantly enriched the outcomes of this study, and I am deeply grateful for his guidance and support.

Furthermore, I would like to express my sincere thanks to the staff of the Mechatronics Engineering Department for their assistance and cooperation. Their continuous support and dedication have been instrumental in facilitating the smooth progress of this project.

I am indebted to the library staff of Okan University and Baghdad University for their kind assistance and access to invaluable resources. Their efforts have been instrumental in expanding the breadth of my research and enhancing the overall quality of this work.

Last but not least, I would like to extend my gratitude to all the individuals who have contributed, in any way, to the success of this research. Your support, whether it was through providing valuable insights, sharing resources, or offering encouragement, has been greatly appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

Yt	Time series variable at time t
ε(t)	Error term at time t in the time series model
P(θ|y)	Posterior probability given the data
P(y|θ)	Likelihood function
f(x; μ,γ)	Probability density function of a Cauchy distribution
θ̂α(X)i	Threshold estimate of θ for a given α and data point i
Ø	Wavelet coefficient in the discrete wavelet transform
λ	Lambda (threshold parameter in denoising scaling or shaping parameter)
N	Number of observations or data points
DWTφ	Discrete wavelet transform with wavelet function φ
φ	Wavelet function in the wavelet transform
yhigh(k)	High-pass filtered signal
ylow(k)	Low-pass filtered signal
Φ	Continuous wavelet transform with a father wavelet Φ
ψ	Mother wavelet in wavelet analysis
s	Scaling factor in wavelet equations
w	Weight vector
b	Bias term associated with the wavelet coefficient in DWT
C	Regularization parameter (Penalty parameter)
‖w‖	Euclidean norm of the weight vector
α	Alpha Lagrange multiplier associated with training samples
K(xi,xj)	Kernel function measuring similarity between samples
Ot	Output gate
σ(x)	Logistic sigmoid function
g(x)	Hyperbolic tangent function
ΔW*	Update term in the weight matrix W in neural networks

Δpi	Update term for the input-to-hidden weights in neural networks
ΔR*	Update term for the recurrent weights in neural networks
α	Alpha (Learning rate)
β̂	Estimated coefficients in a linear model
ŷ	Predicted class in a support vector machine
di	Margin distance in a support vector machine
Sm	Multiscale wavelet scalogram with multiple scales

ABBREVIATIONS

NHTSA	National Highway Traffic Safety Administration
SVM	Support Vector Machine
DWT	Discrete Wavelet Transform
WPT	Wavelet Packet Transform
LCWPE	Local Characteristics-scale Wavelet Packet Energy
WEK	Wavelet Energy and Kurtosis
TEOWT	Teager Energy Operator with Wavelet Transform
ASR	Automatic Speech Recognition
ECG	Electrocardiogram
PCA	Principal Component Analysis
NLM	Non-Local Means
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GBM	Gradient Boosting Machines
k-NN	k-Nearest Neighbors
GPU	Graphics Processing Unit
CPU	Central Processing Unit
FPGA	Field-Programmable Gate Array
DSR	Design Science Research
FEA	Finite Element Analysis
PSD	Power Spectral Density
CWT	Continuous Wavelet Transform
FEM	Finite Element Method
ARX	Auto Regressive model with exogenous input
KYP	Kalman-Yakubovich-Popov

GMF	Gear Mesh Frequency
MCSA	Motor Current Signature Analysis
ERA	Eigensystem Realization Algorithm
HVAC	Heating, Ventilation, and Air Conditioning
VAE	Variational Autoencoder
LDA	Linear Discriminant Analysis
STFT	Short-Time Fourier Transform
WPE	Wavelet Packet Energy
MSB	Modulation Signal Bi-spectrum
PMSM	Permanent Magnet Synchronous Motors
REB	Rolling Element Bearings
VFD	Vibration-Based Fault Diagnosis
ANN	Artificial Neural Networks
AI	Artificial Intelligence
ML	Machine Learning
MSC	Software for Multibody Dynamics Simulation
Matlab	Matrix Laboratory
PDF	Probability Density Function
Sym4	Symlet
LPF	Low-Pass Filter
HPF	High-Pass Filter
MRA	Multiresolution Analysis
IEEE	Institute of Electrical and Electronics Engineers
RFE	Recursive Feature Elimination
SRE	Sequential Feature Selection
NN	Neural Network
RBF	Radial Basis Function
C	Regularization Parameter (Penalty parameter in SVM)

Hinge	function used SVM for penalizing misclassifications
Kernel	Function used for transforming data into higher dimensions
Dual	The dual form of the optimization problem in SVM
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
SGD	Stochastic Gradient Descent
MSE	Mean Squared Error
Cross-Entropy	A loss function used in classification tasks
ReLU	Rectified Linear Unit
BPTT	Back-Propagation Through Time
Adam	Adaptive Moment Estimation algorithm
RMSprop	Root Mean Square Propagation
numHiddenUnits	Number of hidden units in the LSTM layer
numClasses	Number of output classes
YTrain/YTest	Training and Testing datasets respectively
YPred	Predicted output
YPredAll	Matrix storing predictions from multiple models
YPredEnsemble	prediction aggregated from multiple models
cm	Confusion matrix
net	LSTM model

I. INTRODUCTION

1.1. Background

Suspension failure in vehicles is a critical issue that poses a significant risk on the roads, leading to car accidents and potentially fatal outcomes. According to data from the National Highway Traffic Safety Administration (NHTSA) [1], suspension-related problems rank as the third most habitual reason for car accidents, often occurring in combination with steering, transmission, or engine issues. It has been reported that this combination accounts for approximately 3% of all traffic accidents. Sadly, the motivation for my thesis stems from a personal tragedy - the loss of a dear friend due to suspension failure.

Suspension failure is a serious concern that can have devastating consequences. When driving a car, drivers may experience clicking or popping sensations in the suspension, which, unfortunately, some drivers may ignore or not recognize as significant. However, it is crucial to address these issues promptly, as they can lead to fatal accidents if left unattended.

One specific danger associated with suspension failure is the potential loss of steering control, particularly in cases of ball joint or end-spindle failure. Such failures can result in a vehicle rollover and a catastrophic accident. The severity of these accidents is further highlighted by real-life incidents, such as the case involving Mena Massoud, a star of Disney's live-action films [2]. Mr. Massoud sued Tesla, alleging that a suspension failure was the cause of the accident that resulted in his damaged car (Figure I-1). Despite the contradictory statements from a Tesla spokesperson, the accident occurred, underscoring the need to uncover the truth behind such incidents.

These instances of suspension failure and their devastating consequences demand action. It is imperative to develop ideas and preventive measures to mitigate these accidents and enhance vehicle safety. Additionally, the creation of a fault recorder for suspension systems could provide valuable insights into the causes and potential warning signs of failures, contributing to improved preventive measures.

By focusing on the prevention of suspension-related accidents and the development of a fault recorder, this thesis aims to address this critical issue and promote safer driving conditions.

Figure I-1 Damaged Car Resulting from Suspension Failure. (Source Forbes)

1.2. Gear Steering outer tie rode

The outer tie rod is a crucial component of the steering system, responsible for connecting the steering mechanism to the wheel knuckle and enabling the wheels to turn. It comprises various parts, including the body, lower cup, lower part of the bearing, lower ring, dust boot, dust boot skirt, upper ring, ball stud, and castle nut with a cotter pin (Figure I-2).

Due to exposure to different forces such as longitudinal, lateral, and vertical forces, the tie rod experiences various types of stresses, leading to gradual wear and tear. If left unattended, this deterioration can pose significant risks and increase the likelihood of accidents. Therefore, it is essential to periodically replace the tie rod based on its individual life cycle, which is influenced by factors like material composition and design [3].

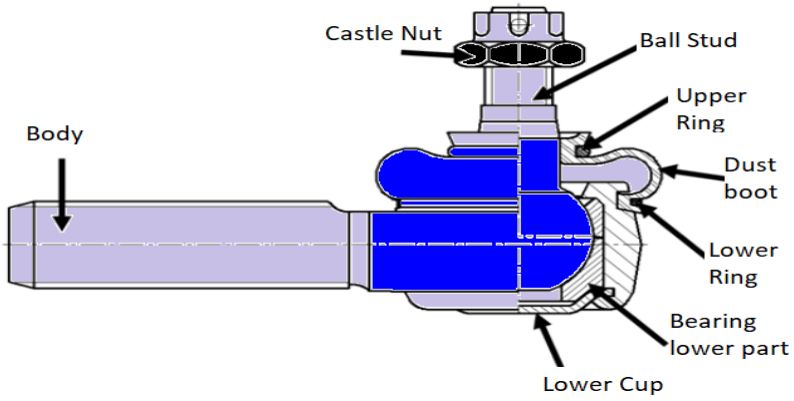

Figure I-2 Outer Tie Rod Subdivisions (Source Wozniak, 2022)

The ball joint in the outer tie rod is particularly vulnerable as it encounters substantial longitudinal, lateral, and vertical stresses. As a result, this specific component is highly susceptible to wear, leading to vibrations and potential failure. The failure typically occurs at the spherical contact interface between the ball pin and the ball socket [4].

Finite element analysis serves as a powerful tool for understanding the tie rod's life cycle, load-bearing capacity, and optimal dimensions. By employing this analysis technique, we can gain insights into the tie rod's performance limitations and expected lifespan, ultimately enhancing vehicle safety and maintenance practices [5].

Additionally, vibrations can occur when there is increased wear or improper tightening of the outer tie rod. These vibrations can serve as an indicator of the extent of damage. By monitoring these vibrations, we can evaluate the level of wear and potential damage to the tie rod.

Our research focuses on utilizing vibration analysis as a tool to assess wear and potential damage in the outer tie rod of the steering system. We aim to monitor tie rod vibrations to determine the level of fault and extent of wear, providing valuable insights into its performance limitations and expected lifespan. This knowledge will contribute to enhancing vehicle safety and maintenance measures, ultimately reducing the risk of accidents caused by tie rod failure.

Figure I-3 Tie-Rod. (Source International Journal of Application or Innovation in Engineering & Management (I JAI EM)

1.3. Vibration Fault Objective

Vibrations in tie rods within vehicle suspension systems can lead to potential dangers and safety concerns. These vibrations can be categorized into noise vibrations and movement vibrations, becoming more noticeable as the tie rod experiences wear and-

tear over time [6]. Incorrect tightening of the tie rod can further exacerbate vibration-related issues. It is crucial to measure and analyze these vibrations to assess the level of danger and develop effective mitigation strategies.

Various methods are available to measure vibrations, including accelerometers, velocity sensors, displacement sensors, strain gauge sensors, microphone sensors, and fiber optic sensors. One commonly used sensor is the piezoelectric angular acceleration sensor, which is battery-free and provides detailed information about angular acceleration [7]. The choice to measure angular acceleration instead of angular displacement or angular velocity is because angular acceleration provides more features about the signals [8]. It is important to note that angular acceleration (α) is crucial in understanding the impact of wear on the tie rod. Angular acceleration is measured in degrees per second squared (deg/s^2), and as wear increases, vibration intensity also increases until failure eventually occurs. Analyzing these vibrations thoroughly allows us to classify the level of danger based on their intensity, providing valuable insights for effective risk assessment and developing mitigation strategies.

One potential method for identifying faults in the tie rod and categorizing them into various levels involves the implementation of Automatic Speech Recognition (ASR) using wavelet-based feature extraction and classifiers, as proposed by [9]. This technique has also been used in fault detection for other devices such as gearboxes [10] and has been used for fault detection in heart signal ECG [11]. By employing this approach, vibrations in the vehicle suspension system, specifically in the tie rod, can be detected and analyzed, thereby facilitating fault diagnosis.

Using vibration analysis techniques allows us to gain comprehensive knowledge about the tie rod's condition. This knowledge is vital for enhancing vehicle safety, optimizing maintenance practices, and ensuring the reliability of the steering gear system.

1.4. Vibration fault identification

Vibration fault detection involves identifying the specific fault source by comparing test vibration data with fault models. The process consists of training (modeling) and identification (matching) units [12] Training builds fault models based on vibration features extracted from known fault samples, while identification calculates the correspondence between input features and fault models. Success in fault identification relies on effective training and identification. Vibration data is used to detect and pinpoint the source of the fault [13]. process can be further simplified and visualized using the provided flowchart (Figure I-4).

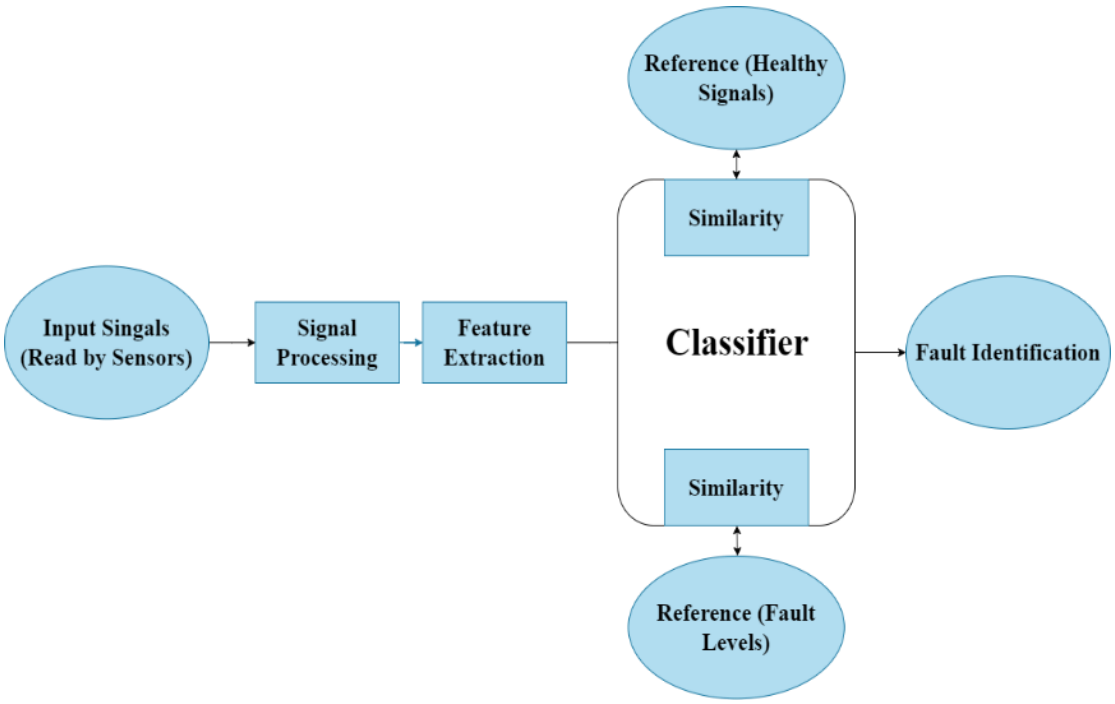

Figure I-4 Fault Identification Flowchart

The process begins with the signal undergoing preprocessing, which is essential for preparing the signal for fault identification. During this stage, the signal undergoes several key steps.

Firstly, the signal is labeled, categorizing it based on known fault types. This labeling step establishes a reference for the subsequent fault identification process. Additionally, denoising techniques are applied to remove unwanted noise and artifacts from the signal. This ensures that the subsequent analysis focuses on relevant information by eliminating unnecessary disturbances [14].

Following preprocessing, the signal enters the feature extraction stage. Various methods are utilized to extract meaningful information from the vibration data. Standard techniques include Fourier analysis, wavelet transforms, and statistical measures like mean and standard deviation. These methods enable the extraction of relevant features that can effectively differentiate between different fault types [15].

Once the features have been extracted, the signal proceeds to the classification stage. In this stage, a classifier is employed to match the extracted features with fault models, facilitating the identification of the specific fault type. Classification algorithms, such as support vector machines (SVM), decision trees, or neural networks, are commonly utilized for this purpose. The classifier compares the input features with the fault models generated during the training stage and determines the closest match, thereby identifying the fault [16].

In conclusion, vibration fault detection plays a vital role in detecting and localizing faults in machinery. Through effective training in fault models and the use of accurate

identification techniques, maintenance personnel can efficiently address issues, minimize downtime, and optimize overall machine performance.

1.4.1. Signal preprocessing

Signal preprocessing in vibration fault identification involves signal labeling and signal denoising. Signal labeling is the process of assigning appropriate labels or categories to the vibration signals based on the fault or condition being analyzed [17]. This step helps in organizing the data and facilitating subsequent analysis and classification. Signal denoising aims to remove unwanted noise from the vibration signals to improve the accuracy of fault detection and diagnosis. Various methods have been used for denoising, such

1.4.1.1. Wavelet Denoising

Wavelet denoising decomposes a signal into different scales using wavelet transform. Thresholding is applied to remove noise in certain scales while preserving essential signal details [18].

1.4.1.2. Median Filtering

This non-linear method replaces each data point with the median of its neighboring points. It effectively removes impulse noise while preserving signal edges [19].

1.4.1.3. Low-Pass Filtering

Low-pass filters attenuate high-frequency noise while keeping the lower-frequency components of the signal intact. Common examples include Gaussian filters and moving average filters [20].

1.4.1.4. Total Variation Denoising

Total variation denoising minimizes the total variation of the signal while ensuring that the denoised signal remains close to the observed noisy signal. It is useful for signals with sharp edges and discontinuities [21].

1.4.1.5. non-Local Means (NLM)

NLM is a powerful technique that exploits redundancy in signals. It averages similar patches in the signal to reduce noise while preserving key details [22].

1.4.1.6. Principal Component Analysis (PCA)

PCA transforms the noisy signal into a new coordinate system where the first few principal components capture the main signal information while the remaining components represent noise. Removing the noise components yields a denoised signal [23].

1.4.1.7. Sparse Representations

Sparse representations represent signals as linear combinations of few basic elements (atoms) from a learned dictionary. Promoting sparsity helps to effectively suppress noise [24].

1.4.1.8. Kalman Filtering

Kalman filters are optimal for estimating the true signal state from noisy measurements in systems with known or predictable dynamics [25].

1.4.1.9. Machine Learning-based Denoising

Machine learning techniques, such as deep learning and support vector machines, can be applied for signal denoising. These methods learn complex noise patterns and denoise signals effectively [14].

The choice of denoising method depends on the specific characteristics of the signal and the nature of the noise. Combining multiple denoising methods can often yield better results than using a single approach alone.

1.4.2. Features Extraction

Feature extraction is a crucial step in vibration fault identification as it involves extracting relevant information or characteristics from the preprocessed vibration signals. The extracted features serve as inputs for the subsequent classification algorithms. Several methods have been used for feature extraction in vibration fault identification. These methods include

1.4.2.1. Continuous Wavelet Transform (CWT)

The continuous wavelet transform is widely utilized in vibration analysis to create a time-frequency representation of vibration signals. It allows for the detection of transient events and the analysis of non-stationary signals [26].

1.4.2.2. Discrete Wavelet Transform (DWT)

DWT is commonly employed for feature extraction in vibration analysis. By decomposing the vibration signal into different frequency bands, it enables the identification of specific frequency components and energy distribution [27].

1.4.2.3. Wavelet Packet Transform (WPT)

WPT is an extension of DWT, providing a more detailed decomposition of vibration signals. It facilitates a comprehensive analysis of different frequency components and proves useful for fault diagnosis and feature extraction [28].

1.4.3. Wavelet Scattering Transform

Wavelet scattering is an advanced feature extraction method, that offers a stable and translation-invariant representation of data. It effectively captures both low-frequency and high-frequency information, making it robust against variations and deformations. Wavelet scattering has gained popularity in various signal analysis tasks, including vibration analysis [29].

1.4.3.1. Local Characteristics-scale Wavelet Packet Energy (LCWPE)

LCWPE is a wavelet-based feature extraction method that quantifies the energy of signals in different frequency bands. It is valuable for identifying localized faults in rotating machinery [30].

1.4.3.2. Wavelet Energy and Kurtosis (WEK)

WEK combines wavelet energy and kurtosis to extract features relevant for identifying fault-related components in vibration signals. Kurtosis, measuring the non-Gaussianity of the signal, aids in detecting impulsive faults [31].

Teager Energy Operator with Wavelet Transform (TEOWT) TEOWT combines the Teager energy operator with wavelet transform, enhancing the detection of transient signals in vibration data [32].

These methods are widely applied in vibration analysis for various purposes, such as fault detection, condition monitoring, and predictive maintenance of mechanical

systems. Researchers and practitioners often choose specific methods or combinations of methods based on the unique characteristics of vibration data and their analysis objectives. Staying up-to-date with the latest research and literature is essential for accessing the most current information on wavelet-based feature extraction methods in vibration analysis.

1.4.4. Feature Selection, Training, Testing, and Classification

After feature extraction, the next steps involve feature selection, training, testing, and classification. Feature selection aims to identify the most relevant and informative features from the extracted set of features. This step helps in reducing the dimensionality of the data and improving the efficiency and accuracy of the classification algorithms. Various classification methods have been used in vibration fault identification, including

1.4.4.1. Support Vector Machine (SVM)

SVM is a powerful and versatile classifier that works well for both linear and non-linear data. It is particularly effective when dealing with high-dimensional feature spaces, making it suitable for vibration data analysis. SVM aims to find an optimal hyperplane that separates different classes, making it useful for binary classification tasks where you want to distinguish between normal and faulty states [33].

1.4.4.2. Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve classification accuracy and robustness. It can handle large datasets with high dimensionality and is less prone to overfitting. Random Forest can be effective in

detecting various fault conditions by analyzing the patterns present in the vibration data Gradient [34].

1.4.4.3. Boosting Machines (GBM)

GBM is another ensemble learning technique that sequentially builds multiple weak learners to create a robust predictive model. It is known for its high accuracy and ability to handle complex relationships in data. GBM can be well-suited for vibration fault detection tasks where there might be intricate interactions between distinctive features [35].

1.4.4.4. Neural Networks

Deep learning-based approaches, particularly neural networks, have shown promising results in various fault detection tasks, including vibration analysis. Convolutional Neural Networks (CNNs) can effectively extract features from vibration signals, while Recurrent Neural Networks (RNNs) can capture temporal dependencies. Long Short-Term Memory (LSTM) networks, a type of RNN, are well-suited for sequential data like vibration time series [36].

1.4.4.5. k-Nearest Neighbors (k-NN)

k-NN is a simple but effective non-parametric classifier for vibration fault detection. It classifies data points based on the majority class among their k-nearest neighbors in the feature space. k-NN can be valuable when the underlying data distribution is not well-defined or when dealing with imbalanced datasets [37].

1.4.4.6. Decision Trees

Decision trees are easy to interpret and visualize, making them valuable for understanding the reasoning behind the classifier's decisions. They can be useful for fault detection when the features have clear decision boundaries [38].

1.4.4.7. Ensemble Methods

Ensemble methods like AdaBoost and XGBoost combine multiple weak learners (e.g., decision trees) to create a strong, accurate classifier. These methods can handle noisy or imbalanced data and are effective in capturing complex patterns in vibration signals [39].

Selecting the most appropriate classifier for vibration fault detection may require experimentation and tuning based on the specific dataset and problem requirements. Other crucial factors to consider include data size, dimensionality, class distribution, and available computational resources. These methods utilize the extracted and selected features to classify the vibration signals into different fault categories. The training and testing phases involve training the classification model using a labeled dataset and evaluating its performance on unseen data. This helps in assessing the accuracy and effectiveness of the classification algorithm in identifying and diagnosing faults in vibration signals. In summary, signal preprocessing in vibration fault identification involves signal labeling and denoising. Feature extraction methods include improved deep learning algorithms, hierarchical fuzzy entropy, wavelet packet energy entropy, WCFSE, and histogram features. Feature selection, training, testing, and classification methods such as SVM, CNN, fuzzy logic, and mathematical statistics are used for fault identification.

1.5. Best Model for fault Identification

Choosing the most suitable model for fault identification depends on the characteristics of the signals being analyzed. Diverse types of signals, such as suspension vibrations, heart signals, human sounds, AC machine signals, engines, and other reciprocating signals, exhibit unique forms and timings. Consequently, each signal type requires specific approaches for feature extraction, filtering, and classification [40].

The selection of an optimal fault identification method depends on the designer's primary objectives. If accuracy is paramount, the chosen approach should prioritize precise fault detection. This means selecting a method or model that can accurately identify faults with minimal false positives or false negatives. Various techniques, such as machine learning algorithms or statistical methods, can be employed to achieve high accuracy in fault identification [41].

Conversely, if the time required for accomplishing the task is crucial, a method that offers faster results may be more appropriate. In time-sensitive scenarios, such as real-time fault detection in critical systems, the speed of the identification process becomes a priority. In such cases, simpler algorithms or rule-based approaches that can quickly process the signals and provide prompt fault identification may be preferred [42].

The hardware employed for fault identification, such as GPUs, CPUs and FPGA, also influences the decision-making process. Certain models or algorithms may be better suited for specific hardware configurations. GPUs, with their parallel processing capabilities, are known to excel in tasks that can be parallelized, such as certain machine learning algorithms. On the other hand, CPUs may be more suitable for algorithms that

require sequential processing or have lower computational requirements, and FPGAs offer customization and optimization possibilities [43].

Cost can also be a factor in selecting a fault identification method. Some methods may require expensive hardware or extensive computational resources, which may not be feasible in certain situations. In such cases, cost-effective approaches that can still achieve the desired level of accuracy and speed may be preferred.

When it comes to fault identification, numerous scientific techniques and algorithms can be employed. These approaches are typically categorized into two main groups model-based and data-driven methods.

Model-based methods These techniques involve developing mathematical models that represent the system under analysis. Fault identification is performed by comparing the behavior of the actual system with the predicted behavior based on the model. Model-based methods are particularly useful when a deep understanding of the system's dynamics is available. However, they may require extensive knowledge and expertise in system modeling.

Data-driven methods These approaches rely on analyzing the measured data directly without explicitly modeling the system. Data-driven methods are advantageous when the underlying system dynamics are complex or not well understood. They often utilize machine learning algorithms to automatically learn patterns and correlations in the data. These methods can be further divided into supervised learning, unsupervised learning, and semi-supervised learning, depending on the availability of labeled training data [44].

Supervised learning algorithms, such as support vector machines (SVMs) or random forests, require labeled examples of both normal and faulty behavior to train the model. Unsupervised learning algorithms, such as clustering or anomaly detection methods, can identify abnormal patterns in data without relying on labeled examples. Semi-supervised learning combines aspects of both supervised and unsupervised learning, utilizing a limited amount of labeled data along with a larger amount of unlabeled data [45].

To enhance fault identification accuracy, feature extraction techniques play a crucial role. Feature extraction involves transforming the raw signals into a set of representative features that capture relevant information for fault identification. Common techniques include Fourier transforms, wavelet transforms, time-frequency analysis, and statistical measures [46].

In conclusion, selecting the best model for fault identification involves considering the specific characteristics of the signals, the desired objectives (accuracy vs. time efficiency), and the available hardware resources. Both model-based and data-driven methods offer effective approaches, with data-driven methods, particularly machine learning algorithms, being popular choices due to their ability to handle complex and unmodeled systems. Feature extraction techniques further aid in improving fault identification accuracy by extracting pertinent information from the signals.

1.6. Thesis Scope

The scope of this thesis encompasses the prevention and detection of suspension-related accidents in vehicles, with a specific focus on the outer tie rod component of the steering system. The research will primarily involve the analysis of vibrations in the tie rod to

assess wear and potential damage. The thesis will explore the use of vibration analysis techniques, such as Fourier analysis, wavelet transforms, and statistical measures, to extract meaningful information from the vibration data. The objective is to monitor tie rod vibrations and determine the level of fault and extent of wear, providing valuable insights into the tie rod's performance limitations and expected lifespan.

Additionally, the thesis will investigate fault identification methods, including Automatic Speech Recognition (ASR) and wavelet-based feature extraction, to detect and categorize faults in the tie rod. The aim is to develop effective fault identification techniques that can accurately diagnose faults and facilitate the implementation of appropriate mitigation strategies.

The thesis will focus on the outer tie rod component and its associated vibrations, considering factors such as material composition, design, and the specific failure modes of the tie rod. The research will involve experimental data collection and analysis to validate the effectiveness of the proposed vibration analysis and fault identification methods.

The scope of the thesis does not extend to other suspension components or systems, such as shocks, struts, or control arms. The research will primarily concentrate on the outer tie rod and its role in suspension failure and accident prevention. However, the findings and methodologies developed in this thesis may have broader implications for vehicle suspension safety and maintenance.

Overall, the thesis aims to contribute to the improvement of vehicle safety by developing preventive measures and utilizing advanced fault detection technologies specifically tailored to the outer tie rod component.

1.7. Research Objectives

1. To investigate the causes and factors contributing to suspension failure in vehicles, with a specific focus on the outer tie rod component of the steering system.
2. To analyze and assess the vibrations in the outer tie rod as an indicator of wear and potential damage, aiming to develop a comprehensive understanding of the tie rod's performance limitations and expected lifespan.
3. To develop effective fault identification methods, such as Automatic Speech Recognition (ASR) and wavelet-based feature extraction, to detect and categorize faults in the tie rod.
4. To explore preventive measures and maintenance practices that can enhance vehicle safety and reduce the risk of accidents caused by tie rod failure.
5. To validate the proposed vibration analysis and fault identification methods through experimental data collection and analysis.
6. To contribute to the improvement of vehicle safety by providing insights and recommendations for the design, maintenance, and monitoring of the outer tie rod component.

1.8. Methodology

The research will follow a Design Science Research (DSR) methodology, as proposed by [47]. The six steps of DSR, including programming, data collection and analysis, synthesis of objectives and analysis results, development, prototyping, and documentation, will guide the research process. This systematic approach will ensure that the research objectives are addressed in a rigorous and structured manner.

The research design will involve both quantitative and qualitative methods. Quantitative data will be collected through vibration measurements and analysis, utilizing techniques such as Fourier analysis, wavelet transforms, and statistical measures. Qualitative data will be gathered through interviews and surveys to gain

insights into maintenance practices and the effectiveness of fault identification methods.

The research instruments will include sensors for vibration measurement, questionnaires for data collection, and interview protocols for qualitative data gathering. The research will employ a mixed-methods approach, combining the strengths of quantitative and qualitative analysis to provide a comprehensive understanding of suspension failure and fault detection in the outer tie rod.

Data analysis will involve statistical techniques for quantitative data, such as correlation analysis and regression analysis, to identify patterns and relationships. Qualitative data will be analyzed using thematic analysis to identify key themes and patterns in the responses.

The research will also include experimental testing to validate the proposed vibration analysis and fault identification methods. Prototypes will be developed and tested under controlled conditions to assess their effectiveness in detecting and categorizing faults in the tie rod.

Overall, the research methodology will ensure a systematic and rigorous approach to address the research objectives, combining quantitative and qualitative methods, and validating the proposed methods through experimental testing. The research findings will contribute to vehicle safety and provide insights into the design and maintenance of suspension systems.

1.9. Thesis Overview

There will be six chapters accordingly

Chapter 1 Introduction: The first chapter of the thesis introduces the topic of suspension failure in vehicles and its impact on road safety. It highlights the importance of addressing this issue and provides the motivation behind the research. The chapter outlines the objectives and scope of the thesis, setting the foundation for the subsequent chapters.

Chapter 2 Literature Review: The literature review chapter provides a comprehensive overview of existing research and knowledge related to suspension failure and fault detection in vehicles. It examines relevant studies, theories, and methodologies employed in the field. The chapter critically analyzes the strengths and limitations of previous research, identifies gaps in the literature, and establishes the theoretical framework for the current study.

Chapter 3 Methodology: In this chapter, the thesis describes the research methodology employed to achieve the stated objectives. It outlines the research design, data collection methods, and analysis techniques utilized in the study. The chapter also discusses any limitations and constraints encountered during the research process and explains how they were addressed. The methodology chapter provides a clear and detailed explanation of the steps taken to ensure the validity and reliability of the findings.

Chapter 4 Data Results: This chapter presents the findings of the data analysis conducted in the study, along with their interpretation and discussion in relation to the research objectives. The results are presented in a clear and organized manner, utilizing

tables, graphs, and other visual aids to enhance understanding. Additionally, the implications of the findings are examined, compared to existing literature, and supported by appropriate references. Practical implications for suspension failure prevention and fault detection are explored, along with recommendations for future research and practical applications.

Chapter 5 Conclusion: The conclusion chapter summarizes the main findings of the study and restates the research objectives. It highlights the contributions of the research, discusses its limitations, and suggests areas for further investigation. The chapter concludes with a final reflection on the significance of the research and its potential impact on improving vehicle safety and preventing suspension-related accidents.

Overall, the thesis aims to provide a comprehensive analysis of suspension failure in vehicles, focusing on the outer tie rod component. It utilizes vibration analysis techniques and fault identification methods to enhance understanding, detection, and prevention of suspension failures. The subsequent chapters will delve into the research process, analysis, and interpretation of the findings.

II. LITERATURE REVIEW

The literature review chapter examines the existing knowledge and research on suspension failure in vehicles, with a focus on the outer tie rod and its role in the suspension system. It aims to understand the causes and consequences of suspension failure and explore the potential application of vibration analysis and fault identification techniques in improving vehicle safety. By reviewing relevant literature, this chapter contributes to the understanding of suspension failure and provides insights for preventive measures. It also discusses the importance of vibration analysis and fault identification in detecting faults in the outer tie rod and suspension system.

2.1. Overview of suspension systems problem

Suspension systems are crucial components of vehicles, contributing significantly to driving comfort, steering control, and road friction. Ahmed Aboazoum underscores the prevalence of common suspension problems such as poor wheel alignment, faulty shocks or struts, damaged springs, failing ball joints, and faulty control arms. To uphold optimal suspension performance, Aboazoum advocates for regular inspections and timely repairs [48]. Meanwhile, Ravi Kumar et al. delve into investigating the failure of primary suspension systems within FIAT-type LHB bogies. Their study focuses specifically on the fatigue failure of primary helical springs and proposes design modifications aimed at enhancing fatigue life. Employing a flexible finite element model, the researchers conduct multi-body dynamic analysis and stress analysis. They estimate fatigue life using the Modified Goodman diagram and validate design

modifications through comparative fatigue life estimation [49]. In parallel, Saurabh D. Shinde, Shruti Maheshwari, and Satish Kumar's literature review delves into the analysis of McPherson suspension system components, with a particular emphasis on the strut mount. They highlight the critical role of suspension systems for safety and comfort while underscoring the risks associated with poor design. Previous research endeavors have leveraged computer-aided engineering techniques, encompassing static and dynamic simulations, finite element analysis, and fatigue analysis, to study various components. However, the authors identify a research gap concerning strut mount failure and propose design modifications to mitigate this issue [50].

2.2. Previous Research on Suspension Failure and analysis

Suspension systems play a crucial role in ensuring the safety, performance, and comfort of automotive and railway vehicles. Detecting and diagnosing faults in suspension systems is essential for maintaining their reliability and preventing potential failures. Over the years, researchers have conducted numerous studies to develop effective techniques for detecting and diagnosing suspension failures. This topic aims to provide an overview of previous research on suspension failure detection techniques in automotive and railway systems. For instance, Hamed, Tesfa, Belachew, Fengshou, Gu, & Ball (2015) Focus on developing a mathematical model using a seven-degree-of-freedom full car to analyze suspension performance. The study conducts simulations to predict the vehicle's response when driven over speed bumps of different shapes and speeds. The model is validated using experimental data collected from driving the vehicle over a specific bump at a speed of 8 km/hr. The research analyzes suspension

performance in terms of ride comfort, road handling, and stability, considering parameters such as wheel deflection, suspension travel, and vehicle body acceleration. The study also explores the effects of speed and changes in suspension specifications, including tire pressure. The developed model is used for fault detection of under-inflated tires and predicting potential suspension faults [51]. Similarly, Patil & Darade (2018) Conducted a comprehensive study to examine the fatigue life and vibration behavior of the pitman arm in a steering system. Utilized Finite Element Analysis (FEA) using Ansys software and CATIA software for structural analysis and equipment design, respectively. By subjecting the Pitman arm to varying frequencies in Ansys, the researchers successfully identified critical areas prone to structural weaknesses. This research significantly contributes to the understanding of the pitman arm's durability and vibration characteristics within steering systems. The integration of FEA and advanced software tools allowed for a thorough evaluation of the arm's structural integrity and design considerations. The findings have substantial implications for enhancing the performance and reliability of steering systems in automotive applications [52]. In another Study, Reza Kashyzadeh et al. (2015) Conducted a study on predicting fatigue life in suspensions exposed to random vibrations from road roughness. Employed Catia, Ansys, and MATLAB for suspension modeling, fatigue life analysis, and simulation, respectively. By calculating fatigue life using FEM Ansys and comparing it with MATLAB’s PSD function, component life cycles and potential failures were determined. To enhance this approach, integration with vibration fault detection methods was proposed. This integration would improve the accuracy of identifying potential suspension failures, enabling proactive maintenance and fault

mitigation. Kashyzadeh's research contributes to the field by highlighting the potential for further advancements in fatigue life prediction and fault detection in suspensions. Future research can focus on developing advanced algorithms and methodologies to integrate vibration fault detection with fatigue life prediction, enhancing suspension reliability and performance [53]. Xiukun et al. (2013) Propose a novel approach for fault isolation in Light Rail Vehicles (LRVs) suspension systems using the Dempster-Shafer (D-S) evidence theory. The authors address the importance of fault isolation for ensuring train safety and reliability, specifically focusing on the suspension system. They introduce a fault isolation algorithm that incorporates a fault feature database and multi-sensor information fusion. The algorithm utilizes a Kalman filter to generate residuals for fault diagnosis and employs the Eros and norm distance to measure the similarity between new and existing fault features. The obtained similarities are converted into basic belief assignments and fused using the D-S evidence theory to enhance isolation accuracy. The effectiveness of the proposed method is demonstrated through two case studies, highlighting its potential in improving the accuracy of fault recognition and the safety of LRVs [54]. Further, Jin et al. (2019) Introduce a method for estimating actuator faults in active suspension systems. The proposed approach combines an adaptive observer with genetic algorithm optimization to accurately track actuator faults under various conditions. Simulation results demonstrate the effectiveness and robustness of the method, outperforming other approaches. The paper also applies the fault estimation method in fault-tolerant control of the active suspension system, improving ride comfort and ensuring important constraints. Overall, the paper contributes to the field by addressing actuator fault estimation and providing a practical

solution for active suspension systems [55]. Meanwhile, X. Zhu et al., 2019 Addressed the problem of fault detection in vehicle active suspension systems. Proposed a fault detection filter design in the finite-frequency domain to enhance suspension system performance. Utilized the generalized Kalman-Yakubovich-Popov (KYP) lemma to derive a sufficient condition for the residual system with the prescribed H∞ performance index. The authors emphasized the importance of active suspension systems in improving ride comfort and vehicle safety. The proposed approach aimed to enhance suspension performance in the finite frequency range, targeting the frequency band of 4-8Hz known for its sensitivity to human body vibrations. The fault detection filter was formulated as a set of linear matrix inequalities, ensuring stability and performance. The authors demonstrated the effectiveness of their approach through a numerical example and compared it with existing methods. The paper contributes to the field of fault detection in vehicle active suspension systems by providing a finite-frequency domain approach to improve performance and reliability [56]. Addressing fault detection in rail vehicle suspension systems, Mao et al. (2017) propose a fault detection scheme that uses a fault detection observer, considering uncertain track regularity and stochastic noises. The authors introduce disturbances to the suspension system states and design an observer to estimate both the system states and disturbances. They analyze the existence conditions for observer design and develop a systematic detection algorithm based on the residual signal. The paper includes simulation results to demonstrate the observer's behavior and performance. This study contributes to the field of fault detection in suspension systems by considering the impact of disturbances and stochastic noises. The proposed observer-based approach

shows promise in effectively detecting sensor faults in rail vehicle suspension systems. However, further research is necessary to validate the approach using real-world systems and explore its applicability to other types of suspension systems [57]. Sugahara & Researcher (2013) Proposed a fault detection technique for vertical dampers in railway vehicles. The method focuses on analyzing the phase difference between the bounce and pitch motions of bogie frames or the car body using inertial sensors. Through vibration excitation tests and running tests on a meter-gauge line, the authors validate the effectiveness of the proposed technique in detecting faults in both primary and secondary dampers. This approach offers a practical and cost-effective solution compared to traditional methods that rely on strain gauges. By evaluating the phase difference, damper faults can be detected during routine train operations, contributing to the field of fault detection in railway vehicle suspension systems [58]. In the realm of vehicle suspension systems, Yin & Huang (2015) Present a novel fault diagnosis method for vehicle suspension systems. The proposed approach leverages accelerometer measurements and employs a three-step process. Firstly, the method utilizes principal component analysis to determine the number of clusters. Secondly, it detects faults through fuzzy positivistic C-means clustering and fault lines. Finally, the root causes of faults are isolated using Fisher discriminant analysis. Notably, the method offers a practical and efficient solution by relying solely on accelerometer data from the four corners of the suspension. The authors demonstrate the effectiveness of their approach through a comprehensive simulation using a full vehicle benchmark [59]. Wei et al. (2013) Propose a data-driven approach for fault detection in vertical rail vehicle suspension systems. The authors use accelerometer sensors placed in the car

body and bogies to collect data. They investigate PCA-based and CVA-based fault detection methods. The simulation results show their approach's effectiveness for detecting vertical damper and spring faults. The paper highlights the importance of on-line fault detection and condition monitoring for railway vehicle suspension systems. It also discusses previous studies on condition monitoring and fault detection in railway systems. Overall, the paper contributes to the field of condition monitoring and fault detection in railway vehicle suspension systems [60]. Moreover, Sakellariou et al. (2015) Present a feasibility study on vibration-based fault diagnosis in railway vehicle suspensions using a functional model-based method. The authors propose a fault detection and isolation (FDI) unit that is trained using data from a physics-based model of the suspension. The unit achieves fault diagnosis using a data-based method called the functional model-based method (FMBM). The FMBM utilizes a novel class of stochastic ARX-type models to accurately represent the system in a faulty state. The study demonstrates the feasibility of fault diagnosis in railway vehicle suspensions through Monte Carlo simulations. The FDI unit is shown to exhibit high sensitivity and accurate estimation of fault magnitudes, and it is robust to measure noise and other uncertainties. The authors conclude that the FDI unit has the potential to improve safety and performance in railway vehicles [61]. Aravanis et al. (2020) Investigated vibration-based faults in railway vehicle suspension. Analyzed vibrations in a 9-degree-of-freedom railway vehicle setup and compared two fault detection methods physics-based models and data-based models using Principal Component Analysis (PCA). Simulations were conducted in MATLAB using the Isim Function and Monte Carlo simulations. The study utilized baseline and inspection phases for active and faulty

systems, respectively. Fault detection was achieved through FM and PCA methods, analyzing curves and evaluating ROC curves. The experiment faced challenges due to limited vibration signals and the presence of non-measurable system components. The findings contribute to understanding fault detection in railway vehicle suspension systems. Further research is needed to address the challenges and enhance fault detection techniques in practical applications [62]. Additionally, Rahim et al. (2020) Assessed suspension fatigue in automobile systems. Utilized the discrete wavelet transform (DWT) and wavelet energy analysis to classify the fatigue level of suspensions. The study focused on measuring strain signals obtained from the coil spring component while driving on both smooth and bumpy roads. The findings provide valuable insights into the use of DWT and wavelet energy analysis for evaluating suspension fatigue. Future research can explore the refinement of these techniques, the inclusion of additional parameters, and the development of real-time fatigue monitoring systems to enhance suspension performance and reliability [63]. Azadi & Soltani (2009) Investigated fault detection in vehicle suspension systems using continuous wavelet transform (CWT) analysis. Focused on identifying faults in the damper and upper damper bushing (UDB) components by approximating system natural frequencies and frequency components with maximum energy using Morlet wavelet functions. The study involved simulation in ADAMS/CAR software and validation through laboratory tests. The authors suggested enhancing their method by integrating wavelet scattering or discrete wavelet transform along with modified classifiers for improved results and accuracy. Comparisons between these approaches and CWT analysis were proposed to assess their effectiveness in fault detection. The research

provides valuable insights into fault detection in vehicle suspension systems and suggests avenues for further improvement in fault detection capabilities [64]. In conclusion Previous research on suspension failure detection techniques in automotive and railway systems has contributed significantly to the understanding and improvement of fault detection methods. These studies have utilized various approaches, including physics-based models, data-driven models, wavelet analysis, fault detection filters, and fault isolation algorithms. The findings have provided valuable insights into the durability, vibration characteristics, fatigue life, and fault detection capabilities of suspension systems. Further research is needed to address the challenges and enhance fault detection techniques in practical applications, optimize design and material selection, and develop advanced algorithms and methodologies for integrating vibration fault detection with fatigue life prediction. These advancements will contribute to improving suspension reliability, performance, and safety in automotive and railway systems

2.3. Vibration Analysis for Fault Detection

Vibration analysis is a critical technique for detecting faults and predicting failures in rotating equipment, offering a proactive approach to minimize unforeseen failures and plant shutdowns. This literature review presents a compilation of studies that explore various methodologies and applications of vibration analysis for fault detection in rotating machinery. For instance, Nouman Khan and Ajit Prasad S L (2019) explore the vibration characteristics of a gearbox with cracked gear teeth to facilitate early fault detection. The authors employ a back-to-back gear test rig for vibration analysis and

utilize frequency spectrum and order tracking analysis for fault diagnosis. Results demonstrate that faulty gears exhibit higher amplitude vibrations at the gear mesh frequency (GMF), with amplitude increasing in proportion to the number of cracked teeth. Additionally, frequency domain analysis reveals higher-order sidebands at the GMF, further confirming the presence of defects. The study emphasizes the effectiveness of order tracking analysis for fault diagnosis and underscores the importance of timely detection of tooth fractures to avert catastrophic failures [65]. Building upon this, Nandi et al. (2005) provided a comprehensive overview of fault diagnosis techniques for electrical motors. The authors cover various types of faults including stator faults, bearing faults, broken rotor bar faults, and eccentricity-related faults. They highlight vibration frequencies as a key indicator for detecting bearing faults, along with thermal measurements and chemical analysis. For stator faults, techniques such as online partial-discharge tests, axial flux-based detection, and monitoring negative-sequence currents are discussed. Broken rotor bar faults can be detected using motor current signature analysis (MCSA), as well as through voltage and vibration analysis. Eccentricity-related faults can also be detected using MCSA and vibration analysis, with frequency components depending on the eccentricity type. The authors additionally explore the use of artificial intelligence techniques, such as neural networks and fuzzy logic systems, for machine condition monitoring and fault diagnosis, enabling the classification of fault signatures and diagnostic decision-making. This article serves as a valuable resource for researchers and practitioners in the field of electrical motor fault diagnosis [66]. Plante et al. (2015) further extend this understanding by showcasing distinctive use of vibration analysis in detecting faults

and predicting failures in rotating equipment. The authors emphasize the significance of equipment monitoring to mitigate the risks of unforeseen failures and plant shutdowns. Vibration analysis is identified as the primary method for assessing equipment conditions and predicting failures. The study conducts a motor condition monitoring experiment, employing spectrum analysis software and MATLAB to analyze measured vibration data. The severity of vibration and specific natural frequencies are used to determine the motor's condition and identify fault types. The article presents results for three fault conditions (unbalance, mechanical looseness, and bearing fault), showcasing distinctive patterns and peaks in the frequency spectrum associated with each condition. The study concludes by highlighting the efficacy of vibration analysis and proposes further research on analyzing vibration trends for accurate failure prediction and reduced maintenance costs [67]. Expanding the scope, Sheng Fu et al. (2016) introduced a new method for mechanical fault diagnosis using time domain analysis and adaptive fuzzy C-means clustering. It addresses the challenge of identifying defects early due to component noise by calculating nine-time domain parameters as characteristic vectors for fault detection. The proposed approach demonstrates effectiveness in classifying different fault types, including micro-sized faults, based on vibration signals. It outperforms other methods like Hilbert transformation and wavelet denoising in detecting faults in rolling bearings. The simplicity and direct signal processing of time domain analysis make it advantageous in terms of processing time. Overall, the study validates the method's effectiveness and robustness through experiments using bearing data [68]. Dong et al. (2021) address prevalent challenges in existing methods by proposing a method to monitor bolt

looseness using vibration transmissibility analysis, addressing challenges in existing methods. The approach utilizes accelerometers to measure vibrations above and below cargo bolts. The spectral moment factor assesses torque level variations of the bolt group, while the eigensystem realization algorithm (ERA) identifies subtle eigenvalue changes to detect local bolt looseness. Experimental results demonstrate the method's effectiveness in detecting both global and local bolt looseness, making it a practical and cost-effective solution for ensuring safety in bolted joints across industries [69]. Parzinger et al. (2020) shift focus towards automated fault detection in building HVAC systems using machine learning models and statistical tests on residuals. The study emphasizes energy efficiency and the lack of cost-efficient methods for fault detection. They use detailed simulation data from a residential case study house to compare fault-free and faulty operations. Nine statistical tests are applied to analyze residuals for fault detection. Results indicate accuracy is affected by data amount, fault type, and density. Finding the best combination of tests is crucial for accurate fault prediction in building HVAC systems [70]. Huang et al. (2019) leverage a two-stage machine-learning architecture for motor fault detection and feature extraction. The method avoids complex preprocessing by using motor vibration time-domain signals. The first stage utilizes an RNN-based VAE to reduce the dimension of sequential data and improve prediction accuracy. In the second stage, PCA and LDA further reduce dimensionality, enabling visualization and detection of different fault modes. Experimental results demonstrate over 99% accuracy in motor fault detection using a simple neural network. The proposed method has significant advantages over other dimension reduction techniques and is valuable for smart manufacturing and preventive maintenance

decision-making. Overall, the RNN-based VAE approach provides an effective and efficient solution for motor fault detection in various industries [71]. Concurrently, Belkacemi et al. (2020) investigate the detection of induction motor bearing lubrication issues using Discrete Wavelet Transform (DWT) analysis with MATLAB/Wavelets toolbox. Traditional time and frequency domain methods face challenges with non-stationary vibration signals, making DWT a suitable alternative for accurate fault identification. Experimental validation using vibration signals from healthy and improperly lubricated bearings reveals that the DWT enhanced by MATLAB/Wavelets toolbox is effective in diagnosing lubrication defects. The healthy bearing signal exhibits lower magnitude peaks and lacks periodicity compared to the improper lubricated bearing signal. The DWT decomposition process, analyzing magnitude ranges and histogram distributions, supports the procedure's efficiency. The paper suggests future research in intelligent techniques for bearing fault detection and monitoring [27]. Hashemi et al. (2013) proposed a fuzzy model for auto-detecting gear faults based on vibration signal analysis. The model combines conventional methods and uses wavelet transform and statistical indexes as fault criteria. It simplifies the decision-making process by employing fuzzy systems, considering gear signals and fault effects. The model is validated through an empirical setup and performs well in detecting gear faults, even for different setups. It can estimate gear health and status using fuzzy logic, offering a simplified approach to gear fault diagnosis despite limited data availability and manufacturing challenges. Overall, the paper's contribution lies in proposing an effective fuzzy-based method for gear fault detection and health assessment [72]. Wang et al. (2019) introduced an enhanced cyclic modulation

spectrum (CMS) algorithm for detecting broken rotor bar (BRB) faults in induction motors (IMs). The CMS algorithm, based on vibration signature analysis, handles non-stationary and non-linear signals characteristic of IMs with BRB faults. It optimizes window function, length, and step size for short-time Fourier transform (STFT) to improve accuracy and computational efficiency. Compared to motor current and vibration signature analyses, the improved CMS algorithm offers better fault detection and noise immunity. Simulation and experimental studies validate its efficacy in accurately diagnosing healthy and faulty motors with BRBs, making it a promising tool for online fault diagnosis [73]. Guo et al. (2018) contribute to early fault diagnosis in planetary gearboxes based on wavelet packet energy (WPE) and modulation signal bi-spectrum (MSB) analysis. Vibration-based analysis was used to extract fault features by decomposing vibration signals into time-frequency subspaces using WPD. The method accurately diagnosed faults in experimental tests, including chipped sun gear tooth and inner-race fault cases. It offers advantages over existing methods by being data-driven, not requiring extensive system knowledge, and overcoming the limitations of other analysis techniques. The combination of WPE and MSB enables effective fault feature extraction and noise suppression, promising more accurate fault diagnosis in planetary gearboxes, and ensuring machinery safety and reliability. Future research can focus on optimization and addressing method limitations [74]. Meanwhile, Y. Li et al., (2020) propose a novel methodology for wheelset bearing fault detection in railway vehicles. The method comprises two stages morphological signal processing and morphological image processing. It utilizes a double cross-correlation operation to reduce noise and emphasize fault features in the signal. The filtered signal is

transformed into a time-frequency domain image using wavelet transform, and morphological image processing techniques are applied to enhance fault features while eliminating noise. The proposed amplitude-sum-based peak search algorithm extract's fault features from the time-frequency plane. Real vibration signals from a wheelset bearing test rig were used for testing, showing superior performance compared to other methods in detecting various bearing faults, promising safer and more reliable railway operations [75]. Popescu and Aiordachioaie (2018) further expand fault detection to rolling element bearings (REB) using change detection and optimal segmentation of vibrating signals. The authors highlight the importance of fault modeling and predictive health monitoring for REB to prevent machine failure and economic losses. They provide an overview of existing condition-monitoring and fault diagnosis techniques for REB. Their proposed method employs an optimal segmentation algorithm based on a linear regression model with piecewise constant parameters, implemented in a MATLAB toolbox called VIBROTOOL. Experimental evaluations using data sets from Case Western Reserve University demonstrate the method's effectiveness in detecting faults in different components of REB. The method contributes to the field of condition monitoring and fault diagnosis in rotating machines, particularly in the context of rolling element bearings. Overall, the paper offers a promising approach for fault detection in REB through change detection and optimal signal segmentation [76]. Zhu et al. (2021) propose a comprehensive method for diagnosing bearing faults in rotating machinery. Their approach involves time-frequency feature extraction using Wavelet Packet Transform (WPT), followed by Multi-Weight Singular Value Decomposition (MWSVD) for relevant feature extraction and dimensionality reduction. A Support

Vector Machine (SVM) classifier is then used for fault diagnosis. The method is validated with data sets from different sources and outperforms traditional techniques like PCA and SVD in fault diagnosis and feature extraction. The proposed approach shows promise for practical applications in rotating machinery maintenance and fault diagnosis [77]. Liang et al. (2018) contribute to the field by introducing a fault detection method for stator inter-turn short-circuit in Permanent Magnet Synchronous Motors (PMSMs) using stator current and vibration signals. The authors introduce a time-frequency method based on an improved wavelet packet transform to analyze the signals and detect short circuit faults. The feasibility of the approach is demonstrated through experimental tests on a three-phase PMSM. Signal-based methods, which analyze the signals collected from the motors, are preferred for PMSM fault diagnosis due to their speed and independence from specific models. The study contributes to the field of motor fault diagnosis and highlights the importance of signal-based methods for detecting faults in PMSMs [78]. In 2019, Rahnama et al. introduced a novel fault detection method for diode rectifiers in brushless synchronous generators using vibration signals. The approach involves wavelet transform-based feature extraction and multiclass support vector machines for classification. A modified sequential forward subset selection approach is employed for enhanced accuracy. The study includes an extensive literature review on fault detection using vibration signals and explores various fault types and their impact on machine vibration behavior. Experimental results demonstrate the method's effectiveness in detecting rectifier faults, outperforming conventional techniques [79]. The reviewed studies demonstrated the efficacy of diverse fault detection techniques. Integration of wavelet and machine

learning methods holds promise for accurate fault diagnosis. Future research should explore deep learning and big data applications to further enhance fault detection in rotating machinery, benefiting industries with improved reliability and reduced maintenance costs.

2.4. A Comparative Study of Deep Learning Models for Vibration-Based Fault Identification in Rotating Machinery

The field of fault identification in rotating machinery has witnessed significant progress with the integration of deep learning techniques. This paper presents a comprehensive comparative study of two state-of-the-art deep learning models for vibration-based fault diagnosis, each offering distinct advantages over traditional methods. In a 2017 study, Zhang Wei, Peng Gaoliang, and Li Chuanhao proposed a novel approach that utilizes Convolutional Neural Networks (CNNs) with a 2D representation of vibration signals as input. This approach eliminates the need for time-consuming data preprocessing, a common requirement in conventional methods like Fast Fourier Transform (FFT) and Artificial Neural Networks (ANN). Experimental results showcase the method's effectiveness, demonstrating improved fault diagnosis accuracy and stability when compared to a baseline system using FFT paper's significant contribution to intelligent fault diagnosis highlights its potential for further research and development [80]. In a parallel study, Shaheryar et al. (2017) introduces and ANN. Additionally, real-world datasets from the Case Western Reserve University Bearing Data center validate the model's performance. Although specific CNN hyperparameters and computational efficiency details are not fully disclosed, the MCNN-SDAE, a deep-learning framework for vibration-based fault identification in rotating machinery. This model combines

Convolutional Neural Networks (CNNs) with Denoising Autoencoders to facilitate unsupervised feature learning from raw vibration signals. MCNN-SDAE surpasses traditional methods in fault identification on a benchmark dataset of bearing-related faults. By effectively capturing complex fault dynamics and reducing the need for manual feature engineering, MCNN-SDAE exemplifies the capacity of deep neural architectures for vibration-based fault diagnosis in mechanical systems [81]. both proposed deep learning models offer valuable insights and advancements in the field of fault identification in rotating machinery. While the first model focuses on exploiting 2D representations of vibration signals and simplifying data preprocessing, the second model emphasizes the strength of unsupervised feature learning through CNNs and Denoising Autoencoders. The findings from this comparative study can guide researchers and practitioners to explore hybrid approaches that leverage the strengths of both models, ultimately enhancing the accuracy and efficiency of vibration-based fault diagnosis.

In conclusion, the application of artificial intelligence (AI) for fault detection in rotating machinery shows great potential for improving accuracy and reliability. The integration of AI-based machine learning (ML) models with vibration analysis techniques offers a promising approach to detecting faults in rotating machinery. The reviewed studies highlight the importance of utilizing AI and ML methods in vibration-based fault diagnosis (VFD) to enhance the accuracy and efficiency of fault detection. One area of further research is the optimization of parameters in vibration-based ML models to ensure accurate and reliable fault diagnosis. By considering the dynamics of the machine and optimizing the vibration parameters, the developed ML models can be

more effective in predicting faults accurately, even when applied to different machines or under different operating conditions. Another avenue for further research is the comparison and exploration of different AI techniques, such as artificial neural networks (ANNs) and adaptive neuro-fuzzy inference systems (ANFIS), for fault detection in rotating machinery. Comparative studies between ML algorithms and AI neural networks can provide insights into the effectiveness and suitability of different approaches for detecting minor faults in induction motors. To achieve better accuracy and efficiency, future research should focus on utilizing larger and more diverse datasets, integrating multiple AI and ML methods, and optimizing the performance of the developed models. The use of advanced AI techniques, such as deep learning models, can further enhance fault detection capabilities by capturing complex fault dynamics and reducing the need for manual feature engineering. Additionally, the development of efficient algorithms and the utilization of parallel computing techniques can enable real-time fault detection, reducing the computational time required for analyzing large amounts of data [82]. The integration of edge computing and cloud-based platforms can also facilitate real-time monitoring and remote fault diagnosis. In conclusion, further research should focus on utilizing AI for fault detection in rotating machinery, exploring different AI techniques, optimizing parameters, and developing efficient algorithms for real-time fault detection. By addressing these areas, researchers can enhance the accuracy, efficiency, and timeliness of fault diagnosis, leading to improved reliability, performance, and safety in various industries.

III. METHODOLOGY

Our research methodology forms the bedrock of our quest for efficient fault detection in diverse road obstacles. We harness the power of Long Short-Term Memory (LSTM) networks, Neural Networks (NN), and Support Vector Machines (SVM) to address our objectives.

LSTM networks, designed for sequential data, are first elucidated for their role in decoding intricate dependencies. NN's mathematical foundations are outlined to reinforce our understanding. Comparatively, LSTM networks and SVM models are dissected regarding complexity, feature engineering, scalability, interpretability, and performance.

Hardware is pivotal. The Intel(R) Core(TM) i7-7820HK CPU at 2.90GHz, paired with the NVIDIA GeForce GTX 1080 GPU, drives our experiments.

Feature extraction techniques such as Discrete Wavelet Transform and wavelet scattering in Matlab enhance our data analysis.

This holistic approach equips us with the tools, techniques, and computational power to explore fault detection across various road obstacles, making our research a beacon of innovation in the field.

3.1. Simulation Setup

The simulation setup of the study employed two software tools MSC Adams and MATLAB. The car model represented a front-wheel-drive salon car and comprised ten interconnected systems, including the steering system, front and rear tires, brake system, sedan body system, power system (consisting of engine, transmission, and

transverse components), stabilizer bar, rear suspension system, front Macpherson suspension system, and the drive line (Figure III-1).

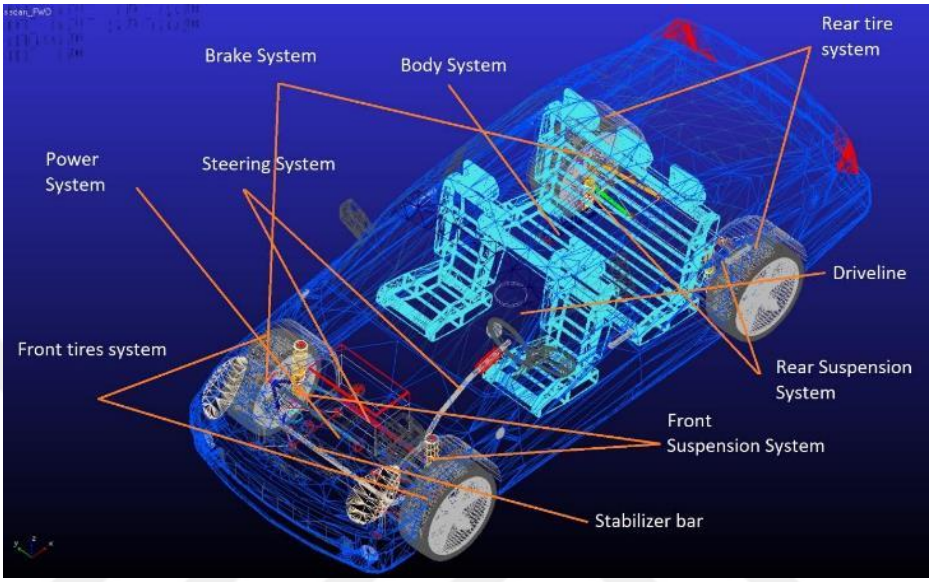

Figure III-1 Car Model in MSC Adams

Central to the analysis was the intensive examination of the front Macpherson suspension system, with particular emphasis on the left outer tie rod (Figure III-2). Signals of pivotal importance were exclusively extracted from this component for in-depth scrutiny.

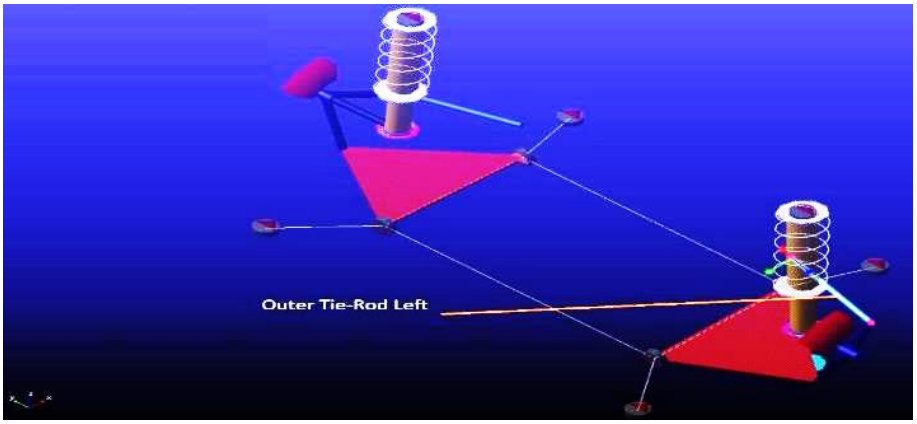

Figure III-2 Left Outer Tie Rod

Upon successfully integrating the car model into MSC Adams, meticulous simulations ensued, employing a straight-line and maintained road scenario. The simulation parameters included 20 seconds, a step frequency of 50, a velocity of 20 km/hr, and a gear position of 20 (Figure III-3).

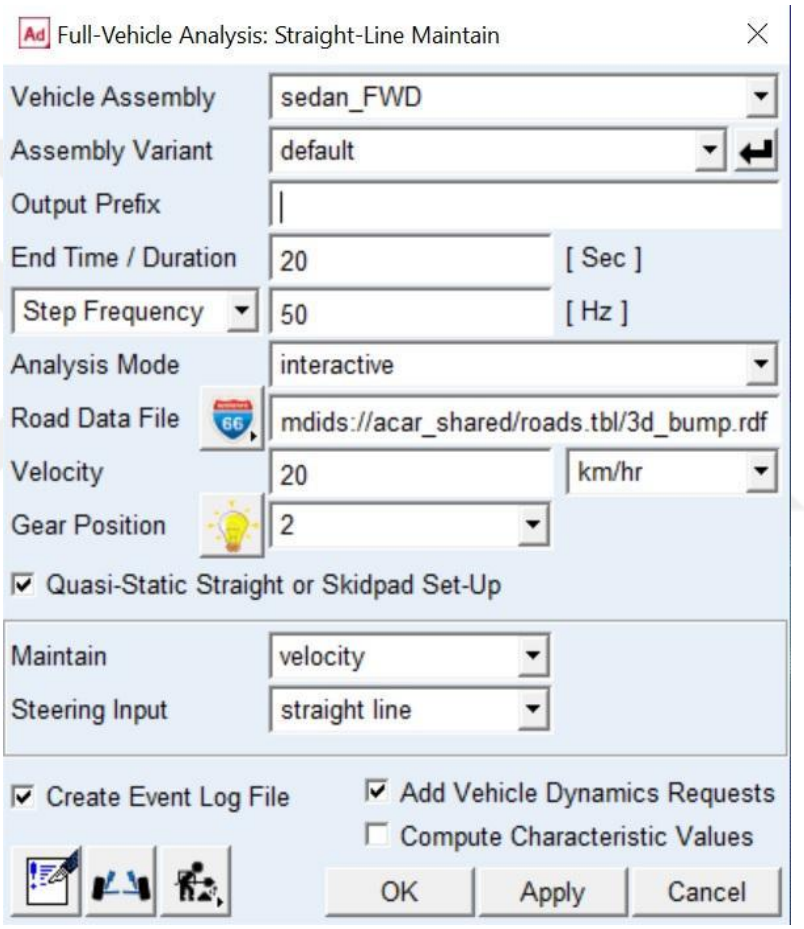

Figure III-3 Simulation Parameters in MSC Adams

Four discrete and meticulously crafted tests (scenarios) were executed on the car model, delving into the realm of the angular acceleration of the left outer tie rod on the X-axis. The outcomes derived from these tests offer profound insights into the car dynamic behavior under diverse road conditions, akin to the interpretive potential of real-life accelerometer readings.

Sine Road Scenario This scenario ingeniously replicated a Sinewave-like obstacle, spanning an expanse of 80 meters (Figure III-4). The resultant angular acceleration is vividly illustrated in (Figure III-5).

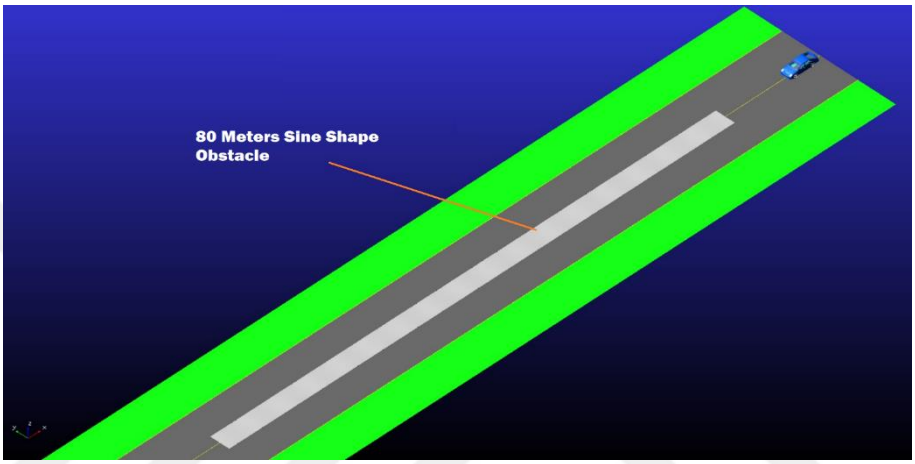

Figure III-4 Sine Road Scenario

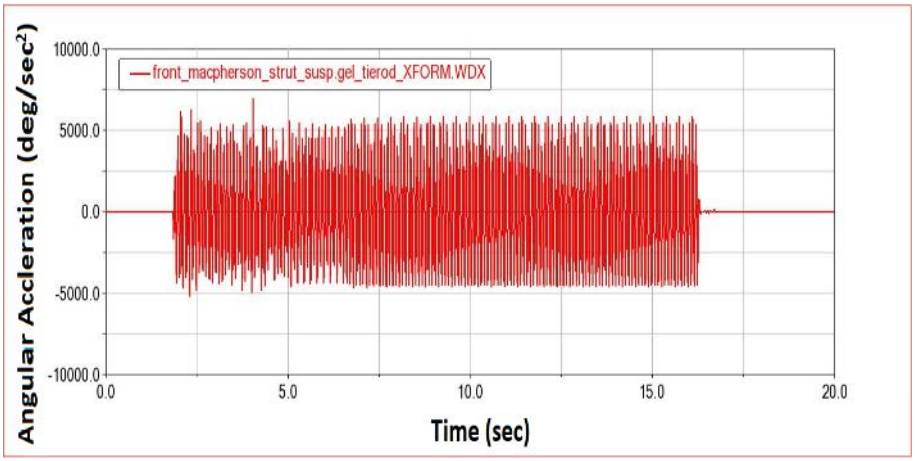

Figure III-5 Angular Acceleration - Sine Road Scenario

Roughness Obstacle Scenario A simulation introduced a distinctly uneven road surface, incorporating a roughness obstacle stretching across 85 meters (Figure III-6). The ensuing angular acceleration is perceptibly displayed in (Figure III-7).

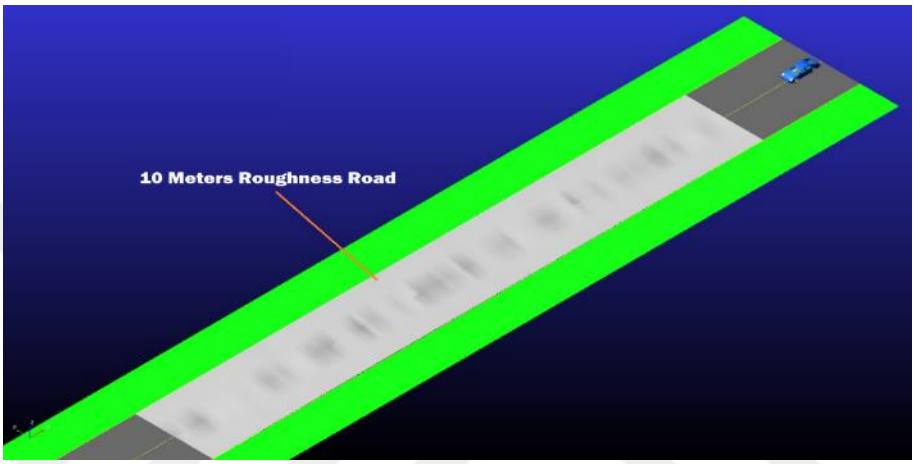

Figure III-6 Roughness Obstacle Scenario

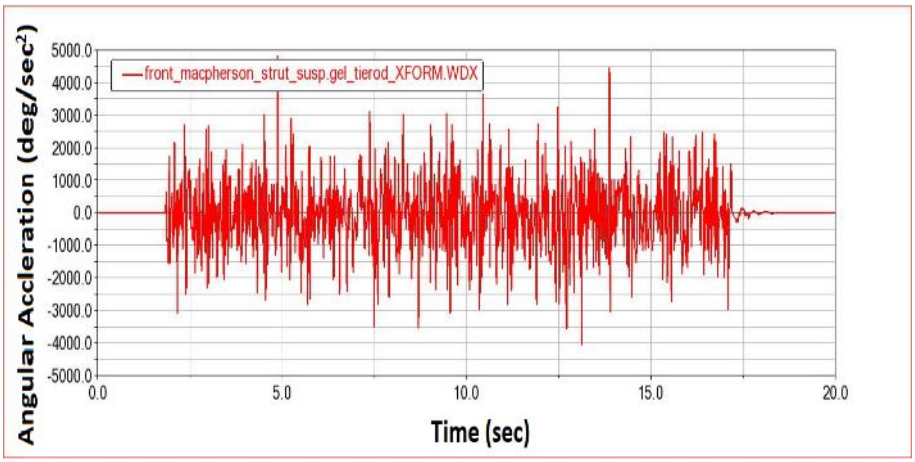

Figure III-7 Angular Acceleration - Roughness Road Scenario

Pothole Obstacle Scenario This scenario emulated a road with a pothole obstacle spanning 10 meters (Figure III-8). The consequential angular acceleration is vividly portrayed in (Figure III-9).

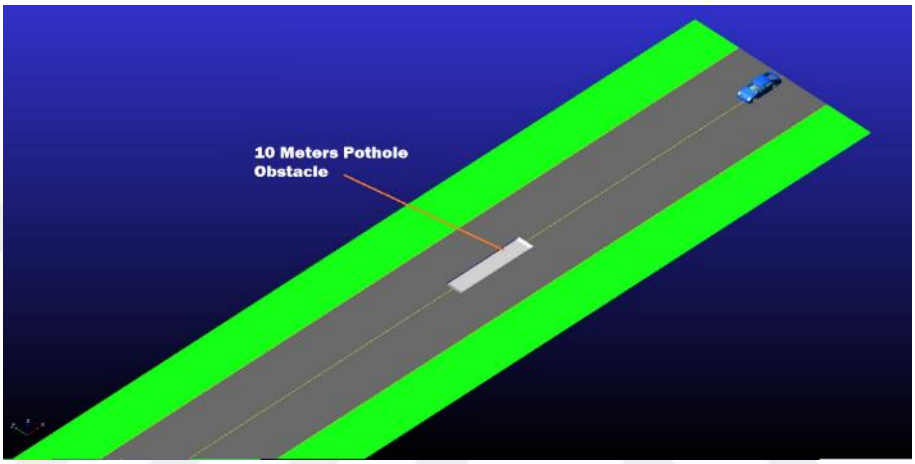

Figure III-8 Pothole Obstacle Scenario

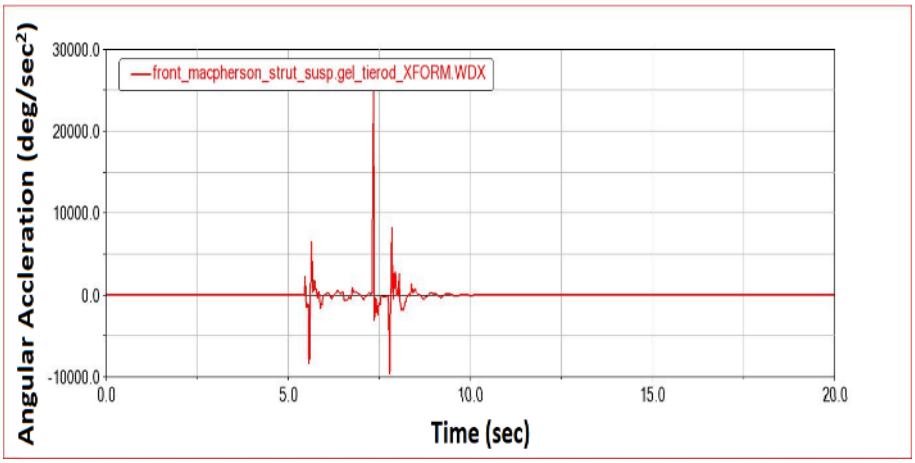

Figure III-9 Angular Acceleration - Pothole Road Scenario

Bump Obstacle Scenario Intriguingly, a bump obstacle measuring 2 meters in length and peaking at a maximum height of 7 cm was tactfully introduced (Figure III-10). The subsequent angular acceleration findings are depicted in (Figure III-11).

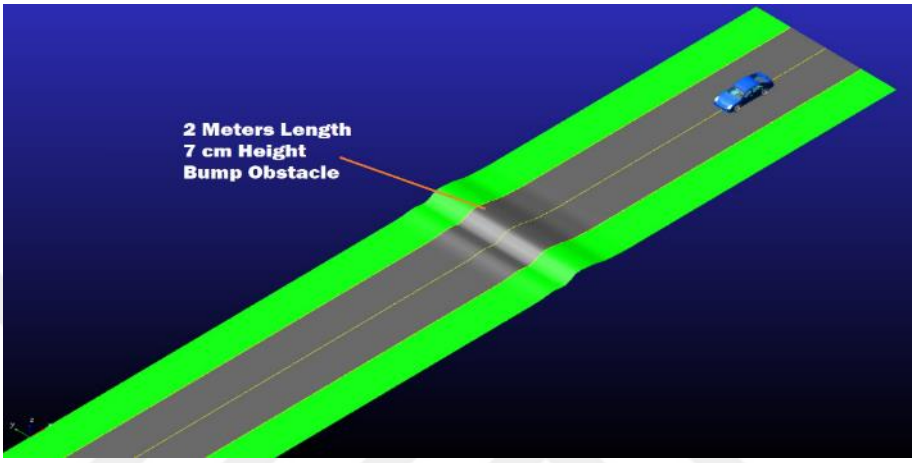

Figure III-10 Bump Obstacle Scenario

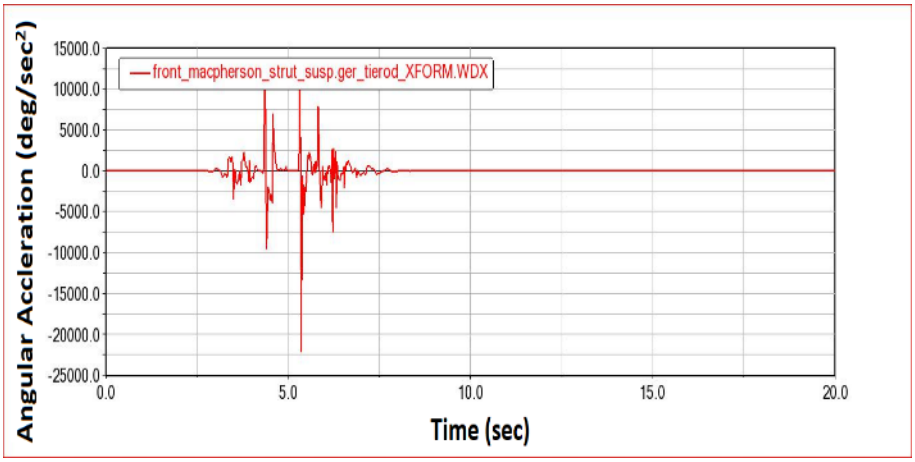

Figure III-11 Angular Acceleration - Bump Road Scenario

The acquisition and analysis of angular acceleration signals underpinned meticulous scrutiny for each distinct scenario. This rigorous data collection procedure echoed the methodologies applied to real-life accelerometers, ultimately generating four

distinctive signals synonymous with the four individual scenarios (Figures I-5, I-7, I-9, and I-11).

Post-simulation (Figure III-12), the ensuing results were seamlessly transferred as .tab files to the MATLAB platform for a profound analytical phase. This meticulous MATLAB analysis encompassed an array of procedures, including signal preprocessing, comprehensive data labeling, intricate feature extraction, and precise training and testing signal classification.

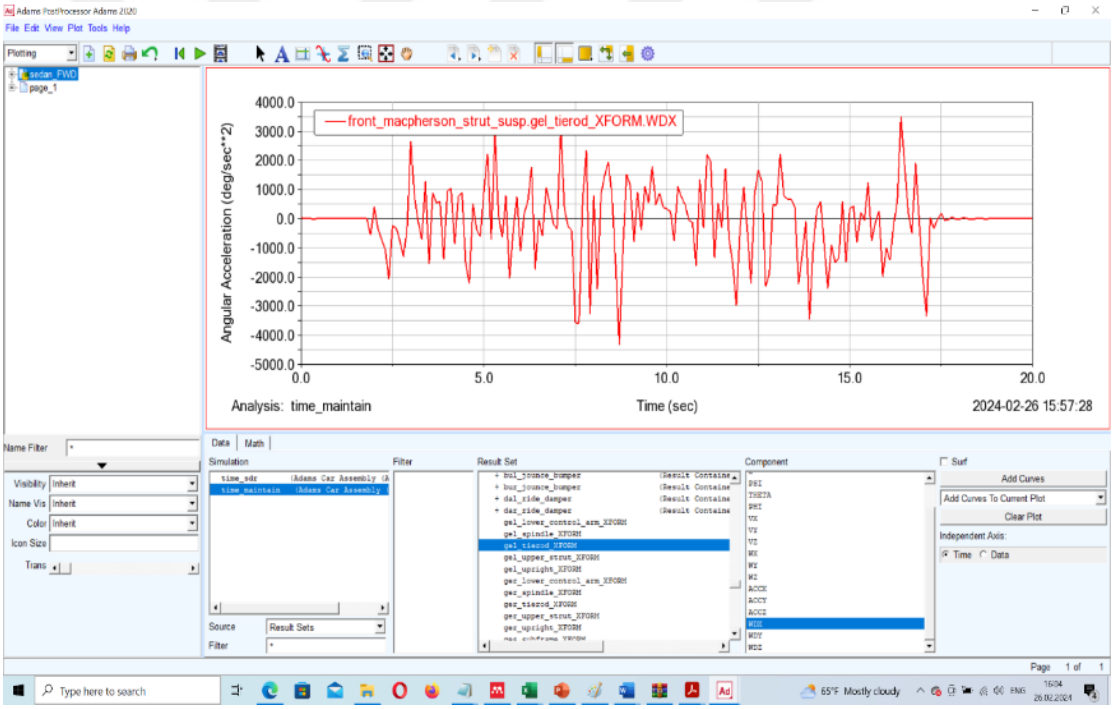

Figure III-12 MSC ADAMS Post-Simulation

The sophisticated array of models implemented within the MATLAB environment facilitated effective interpretation and astute visualization of the outcomes. This holistic approach contributed substantially to our grasp of the multifaceted implications of diverse road obstacles on the angular acceleration patterns of the left outer tie rod within the context of the front Macpherson suspension system.

3.2. Signal Processing Procedure

The raw data obtained from MSC Adams, saved as .tab files, is efficiently imported into MATLAB using the "importdata" function, resulting in the following assignments

MATLAB

```
SineWave = importdata('Sinewave.tab', '\t').data;
Roughness = importdata('Roughness.tab', '\t').data;
Pothole = importdata('Pothole.tab', '\t').data;
Bump = importdata('bump.tab', '\t').data;
```

Subsequent to data import, the temporal and angular acceleration data from each scenario file are meticulously extracted (Figure III-13).

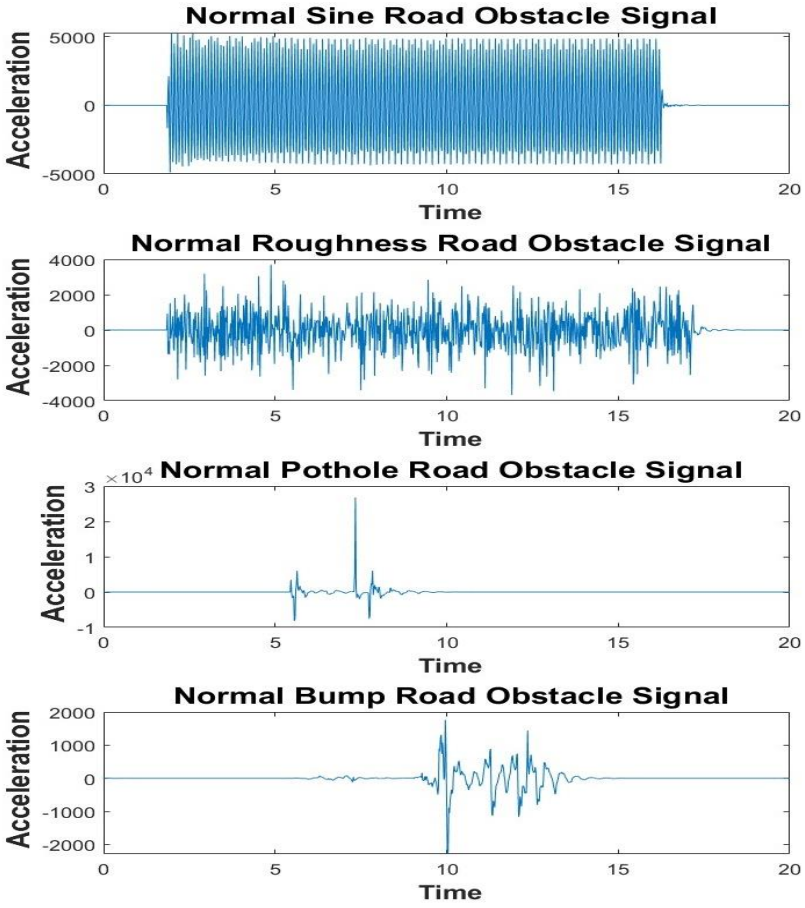

Figure III-13 Signals Extracted by MATLAB For Four Road Obstacles

. This process yields the essential signals represented in (Figures III-5, III-7, III-8, and III-11).

To simulate wear in the Tie rod, a practical limitation in MSC Adams, we manipulate angular acceleration parameters by altering degrees. This adjustment exploits the fundamental correspondence between angular acceleration and (Degree/sec^2).

3.3. Data Preparation

In this section, a dataset of 1000-point signals underwent two tests— one with and one without noise. A third test introduced noise, subsequently denoised using a wavelet denoising MATLAB function. The chosen signal length and noise variations simulate real-world scenarios, enabling comprehensive methodology evaluation. This section details the steps taken to address noise challenges and emphasizes the importance of the denoising process in enhancing signal quality for subsequent analysis.

3.3.1. Operational Modes

The initial mode focuses on distinct signal behavior. Normal signals exhibit a fluctuation range spanning from 0 to around 0.001. Faults are systematically introduced, encompassing a range from 0.005 to 2 degrees across diverse road types (Figure III-13). This design establishes four distinct fault levels, contributing to a collective pool of 400 signals. Each fault level comprises precisely 100 signals. Post-signal generation, healthy, and fault signals are categorized and labeled meticulously. Signal shuffling ensues, culminating in their partitioning into 80 signals for training and 20 for testing (Figure III-14).

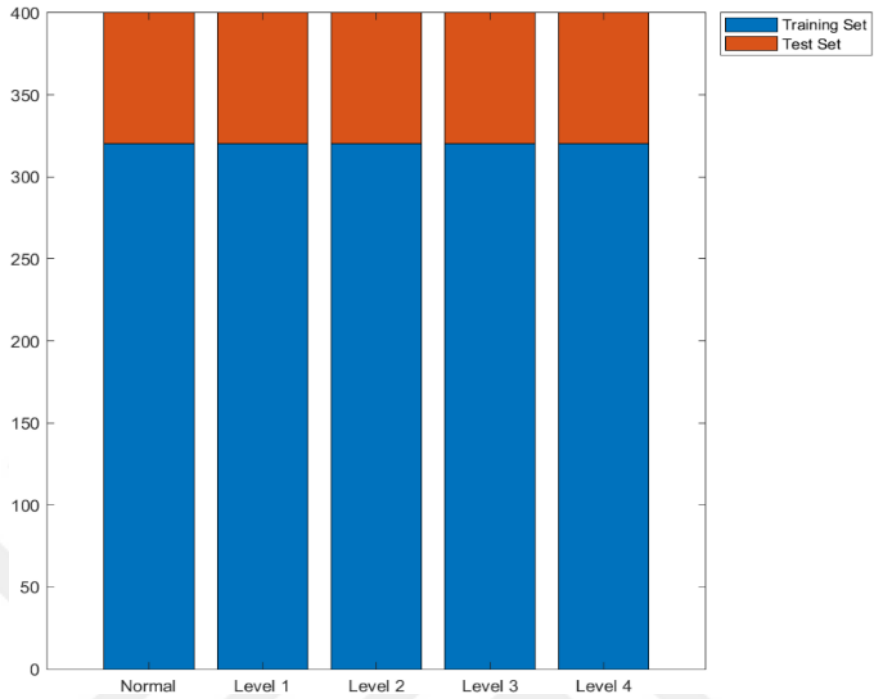

Figure III-14 First Test Mode with Four Level of Faults 80% Training 20% Testing

The ensuing dataset is then prepared for subsequent feature extraction. A rigorous evaluation of the 15 fault detection models is conducted, with the most accurate models subjected to a two-step testing process involving the introduction of noise, simulating conditions akin to a grainy road, followed by denoising testing.

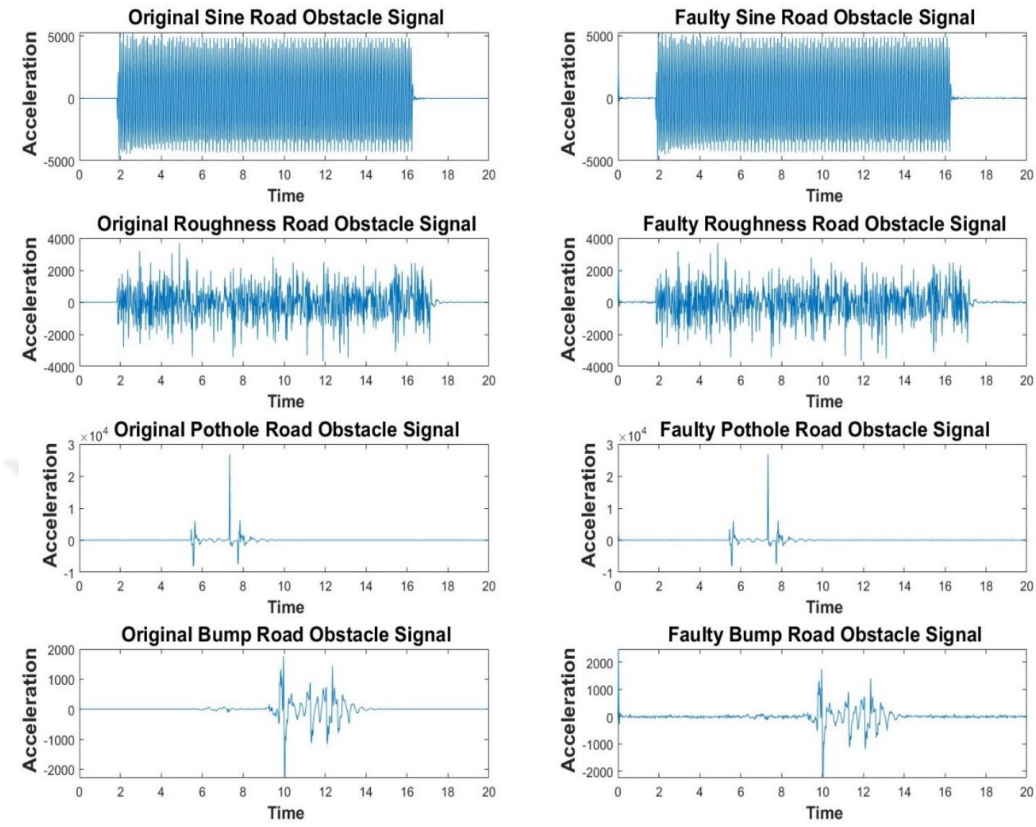

Figure III-15 Normal Signal and Generated Faulty Signals for Four Road Obstacles

In the second mode, the focus is on healthy and wearily representation (Figure III-15). A set of 400 healthy signals per road type is employed. Wear is simulated, ranging from no wear to 0.001, while faults span between 0.005 to 2 degrees, emulating actual conditions. Similar to the preceding mode, signals undergo meticulous labeling, shuffling, and partitioning for training and testing (Figure III-16).

The ensuing dataset is also primed for feature extraction. A comprehensive evaluation of 15 fault detection models is executed under three key scenarios unaltered signals, signals influenced by added noise mimicking a grainy road, and signals subjected to denoising prior to testing.

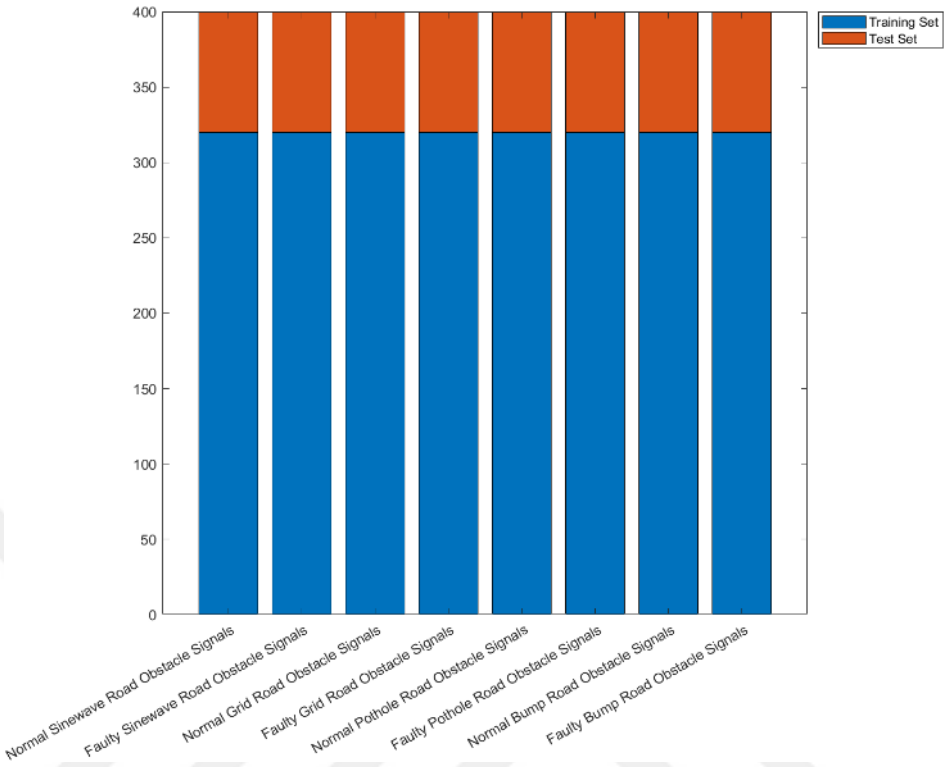

Figure III-16 Roads Under Stress Testing Faulty and Weary Conditions with an 80-20 Split

3.3.2. Noise additions

The noises that will be added is a gaussian noises with standard deviation 30.

we use the code in MATLAB to add this noise as below

noiser = 30 * randn(size(AccData(2end,))

The code snippet provided generates random noise using the formula

noiser = 30 * randn(size(AccData(2end,)))

This code is used to generate random noise with a normal distribution and a mean of 0 and standard deviation of 30. The `randn` function in generates random numbers from a standard normal distribution, and the `size` function is used to determine the size of the `AccData` matrix. The mathematical model related to this code is the random walk

plus noise model. The random walk plus noise model is a commonly used model in time series analysis and econometrics. It consists of a random walk component and a noise component. The random walk component represents a stochastic process that evolves, while the noise component represents random fluctuations around the random walk. The random walk plus noise model can be represented mathematically as:

$$Yt = Yt - 1 + \varepsilon t \quad \text{(III.1)}$$

where Y(t) is the value of the random walk plus noise process at time t, and ε(t) is the noise component at time t. The random noise generated by the code snippet can be seen as the ε(t) term in the model. It represents the random fluctuations around the random walk component [83].

3.3.3. Denoiser

When addressing the challenge of noise reduction, particularly in scenarios like the generation of noise from a grainy road assumption, MATLAB's wdenoise function emerges as a robust and comprehensive solution. This function employs a sophisticated combination of techniques, seamlessly integrated to achieve optimal results in signal denoising.

3.3.3.1. Empirical Bayesian Method

At the core of the denoising process is the application of the empirical Bayesian method. This advanced statistical approach dynamically tailors the prior information based on the observed data. In contrast to traditional Bayesian methods, which rely on fixed priors, empirical Bayesian estimation adapts the prior distribution using the available data.

The central equation, expressing the posterior given the data, exemplifies this adaptability:

$$P(\theta \mid y) \propto P(y \mid \theta) \cdot P(\theta) \quad \text{(III.2)}$$

The equation balances the updated belief distribution of a parameter (θ) following the observation of data (y). It achieves this balance by considering the likelihood of observing the data given the parameter P(y|θ), the initial belief distribution of the parameter P(θ), and the overall probability of observing the data P(y). This equation symbolizes the adjustment of the parameter distribution after data observation, effectively reconciling the likelihood and the initial belief (prior). The empirical Bayesian method serves as a bridge between the challenge of determining a suitable prior and the necessity for data-driven inference. However, caution is advised in its application to prevent potential biases [84].

3.3.3.2. Cauchy Prior

The method incorporates a Cauchy prior, which is a probability distribution characterized by its heavy tails. This characteristic makes it particularly effective in modeling data with outliers or heavy-tailed noise. The probability density function (PDF) of the Cauchy distribution, represented as:

$$f(x;\, \mu, \gamma) \;=\; \left(\frac{1}{\pi\gamma}\right) * \left(\frac{\gamma^2}{(\gamma^2 \;+\; (x \;-\; \mu)^2)}\right), \quad \text{(III.3)}$$

The Cauchy distribution is characterized by three key parameters x (independent variable), μ (location parameter), and γ (scale parameter). μ determines the center of the distribution, while γ controls its spread. The reciprocal of πγ ensures normalization.

Cauchy distributions have heavy tails and lack finite moments, making them useful in statistical modeling.

By integrating the Cauchy prior into a statistical method, it demonstrates robustness in handling scenarios with a high presence of outliers. The heavy tails of the Cauchy distribution allow it to assign non-negligible probabilities to extreme observations, making it resilient to the influence of outliers. This robustness contributes to the method's effectiveness in noise reduction, especially in situations where traditional methods might be sensitive to extreme values [85].

3.3.3.3. Sym4 Wavelet

A key element in the denoising process is the utilization of the Symlet 4 wavelet function. Wavelets serve as fundamental tools in wavelet-based signal processing and denoising, enabling the analysis of signals across various scales. The Symlet 4 wavelet, available both in the time domain and as a representation through the Fourier transform, facilitates a nuanced understanding of signal characteristics. Although the mathematical formulation of this wavelet is intricate, its adoption in the denoising process signifies the method's adaptability to diverse signal structures and its capability to accurately capture important features [86].

$$\hat{\theta}_{\alpha}(X)_i = \begin{cases} X_i - t(\alpha)\ if\ X_i > t(\alpha) \\ 0 \qquad\quad if\ |X_i| \leq t(\alpha) \\ X_i + t(\alpha)\ if\ X_i > -t(\alpha) \end{cases} \tag{III.4}$$

Here, $\hat{\theta}_{\alpha}(X)_i$is the posterior median estimate of the i-th coordinate of the true signal θ, Xi is the i-th observation, α is the prior probability of non-zero coordinates, and t(α) is a threshold that depends on α and the noise level.

3.3.3.4. Posterior Median Threshold Rule

Integral to the denoising methodology is the implementation of the posterior median threshold rule. This strategy guides the denoising process by determining the wavelet coefficients that predominantly represent noise. The main equation representing the posterior median threshold rule is the posterior median (z;w) through the distribution function:

$$\widetilde{F1}\ (\mu \mid z) = \int_{-\infty}^{\mu} f_1(u \mid z)du \qquad \text{(III.5)}$$

Here, $\widetilde{F1}$(μ|z) is the cumulative distribution function (CDF) of the posterior distribution, where μ is the variable representing the value of the posterior median, z is the parameter representing the observed data, and f_1(u|z) is the probability density function (PDF) of the posterior distribution. This distribution function is crucial in determining the threshold t(α) and, consequently, guiding the denoising process [87].

In summary, the wdenoise function within MATLAB harnesses the power of the empirical Bayesian method, integrates the Cauchy prior, employs the Symlet 4 wavelet, and incorporates the posterior median threshold rule. This synergistic amalgamation of techniques establishes the foundation for effective signal denoising. By addressing outliers, accommodating diverse signal structures, and retaining crucial signal characteristics, this approach yields high-quality denoised signals, making it an invaluable tool in noise reduction scenarios.

3.4. Feature extraction

In the realm of signal processing and pattern recognition, feature extraction stands as a pivotal stage. It entails the conversion of raw data into a pertinent set of features that holds the potential for subsequent analysis or categorization. Two prominent

methodologies for feature extraction have been studied discrete wavelet transform (DWT) and wavelet scattering. This discussion delves into the underlying principles of these techniques, their practical applications in distilling significant attributes from signals, and a comprehensive evaluation of their efficacy in portraying signal characteristics.

3.4.1. Discrete wavelet Transform (DWT)

The Discrete Wavelet Transform (DWT) stands as a fundamental mathematical technique employed to dissect and scrutinize signals and datasets. Its significance resonates across diverse domains like image and signal processing, data compression, and feature extraction [88].In contrast to the conventional Fourier Transform, which portrays data through sine and cosine waves spanning distinct frequencies, the DWT dissects data across multiple scales and resolutions. This intricacy is achieved by breaking down the input into an array of wavelets, adept at capturing temporal and frequency insights [89]. Such partitioning opens the door to more versatile and efficient analysis of signals and data, rendering the DWT a pervasive choice in a multitude of applications.

3.4.1.1. Concept of Wavelets

Wavelets are small oscillating functions that are used in various applications of signal processing, data compression, and analysis. They provide a way to analyze signals and data at different scales, allowing us to capture both localized and global features in a more flexible manner than traditional Fourier-based methods.

Within the domain of the Discrete Wavelet Transform (DWT), wavelets assume a pivotal role in the intricate processes of signal decomposition and subsequent

reconstruction. Employing wavelet functions, the methodology entails a meticulous traversal along the signal, resulting in the extraction of salient insights [90].

The efficacy of the DWT becomes manifest in its ability to fractionate signals into distinct frequency constituents, encapsulating both low-frequency trends and high-frequency intricacies. This decomposition unfolds through the convolution of the signal with a gamut of wavelet functions, each distinguished by unique scales and positional parameters [90].

The Discrete Wavelet Transform (DWT) involves a distinct methodology wherein the discretization of scaling and shifting parameters is employed, obviating the direct sampling of either the signal or its transform. This strategic approach engenders heightened high-frequency resolution at low frequencies, concurrently facilitating elevated time resolution for higher frequencies. Notably, this approach ensures a uniform level of time and frequency resolution across all frequency bands.

The decomposition of a discrete signal (x[n]) can be succinctly articulated as follows, drawing inspiration from the seminal work of [88].

$$x[n] = \sum_{k} a_{j0,k0}\,, \emptyset_{\,j0,K0}[n] + \sum_{j=j0}^{j-1} \sum_{k} d_{j0,k0}, \emptyset_{j0,k0} \qquad \text{(III.6)}$$

In this context, the discrete-time signal (x[n]) is represented as the sum of two terms. The first term involves a summation over indices (j_0) and (k_0), denoted as ($a_{j0,k0}$,$\emptyset_{j0,k0}$[n]), where (a) represents coefficients associated with these indices. The second term consists of a double summation over indices (j) and (k), ranging from (j_0) to (j-1) and across all (k) respectively, denoted as ($d_{j0,k0}$,$\emptyset_{j0,k0}$). The parameters (j_0) and

(k_0) serve as initial indices, defining the range over which the summations occur within the signal (x[n]).

This decomposition methodology uniquely equips the DWT to adeptly scrutinize the temporal (Figure III-17) and frequency attributes of a signal across a panorama of resolutions. Consequently, the technique adeptly captures both low-frequency trends and high-frequency intricacies, thereby substantiating its indispensability as a potent tool within the realm of signal processing [88].

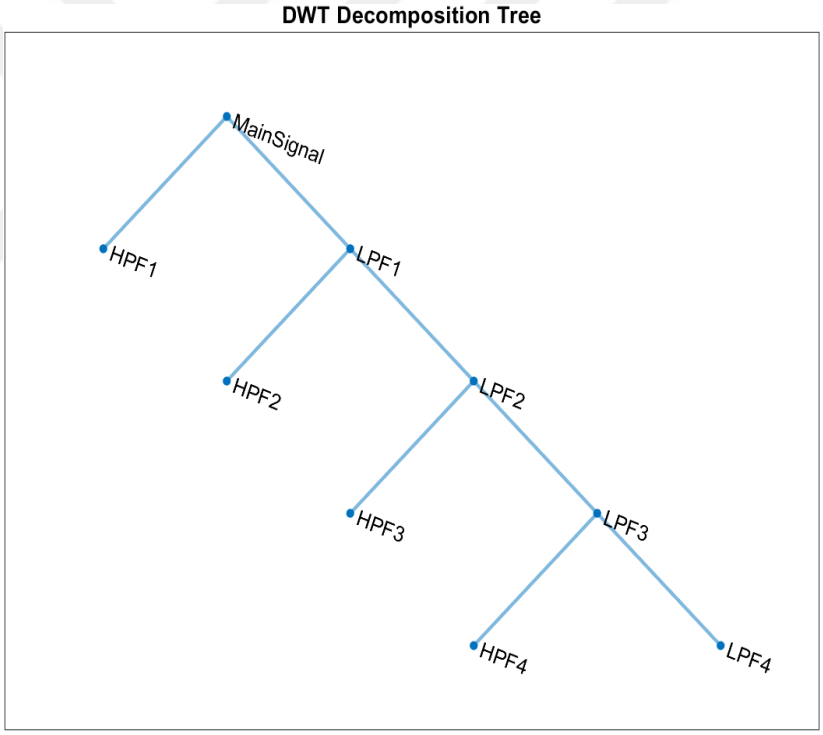

Figure III-17 Discrete Wavelet Transform Decomposition Tree

3.4.1.2. Properties of Wavelets

Wavelets possess two significant properties, namely orthogonality and biorthogonality. In our project, we will mainly focus on utilizing the orthogonality of wavelets. Orthogonal wavelets, in particular, exhibit a crucial attribute where the inner product

of the wavelet function at one scale and position with the wavelet function at another scale and position is equal to zero (Figure III-18). This inherent orthogonality simplifies both the analysis and reconstruction procedures. Moreover, orthogonality enables the conservation of energy during the transformation process in orthogonal wavelet transforms. By leveraging these orthogonality properties, we can enhance the efficiency and accuracy of our project's analysis and reconstruction techniques [91].

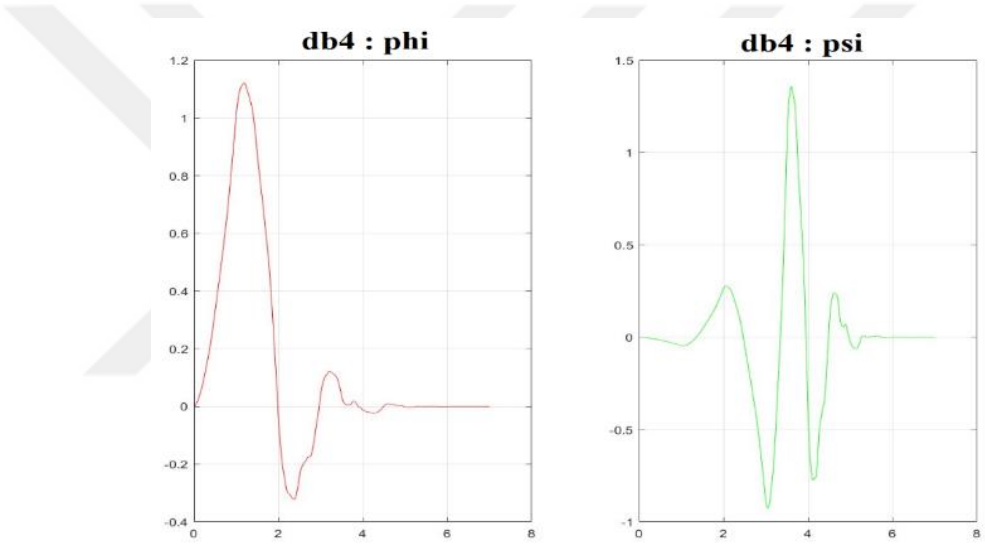

Figure III-18 Orthogonal Wavelets Two Different Scale and Position But the Product is Zero.

3.4.1.3. Types of Wavelet Families

Diverse wavelet families are at one's disposal, each characterized by unique attributes aptly suited for various signal types and applications. Among the well-established wavelet lineages are the Haar, Daubechies, Symlets, Coiflets, Biorthogonal, Morlet, and Mexican Hat families [92].

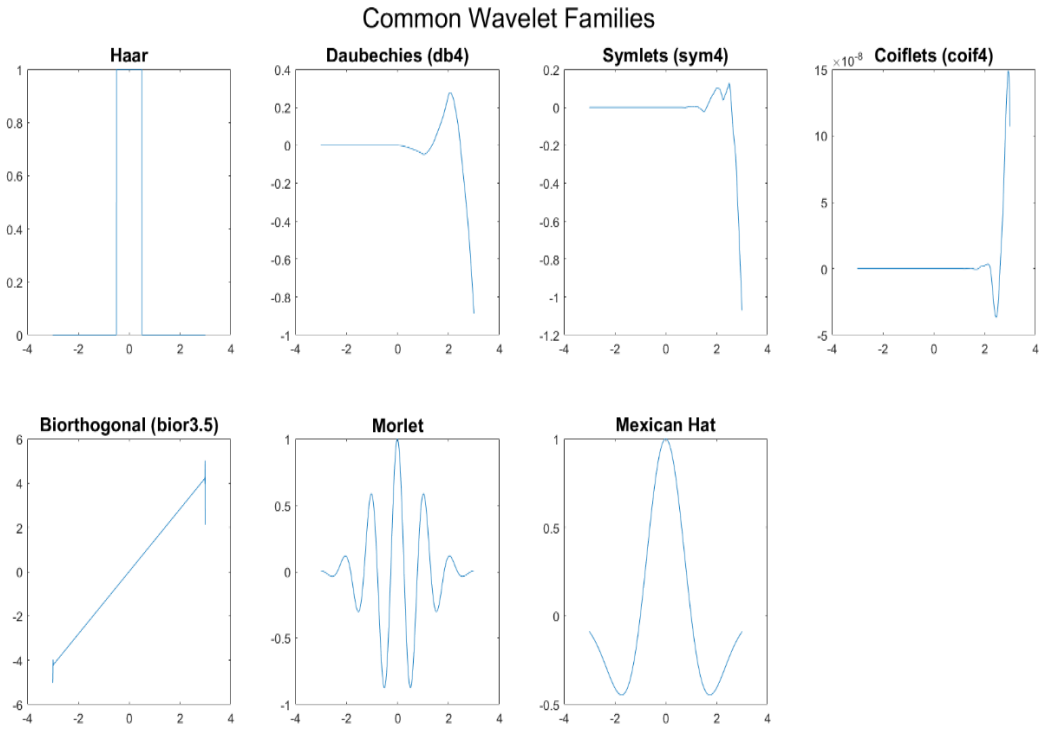

Figure III-19 Common Wavelet Families

Our current experimental focus is dedicated to the exploration of the Daubechies 4 and 5 wavelet families (Figure III-19), with a particular emphasis on decomposition at level 5. Situated within the intermediate range spanning from db2 to db8, these chosen wavelet families have garnered recognition for their notable ability to refine precision within the field of vibration analysis. A noteworthy point of intrigue lies in the Daubechies wavelets, often denoted as "dbN," wherein the parameter N signifies the count of vanishing moments. This numerical identifier underscores their exceptional efficacy in capturing intricate patterns and nuances during the decomposition process, particularly when employed at the fifth level [93]. Renowned for their succinct support and efficient frequency localization, as illustrated in (Figure III-20), these wavelets present a distinguished presence in the field.

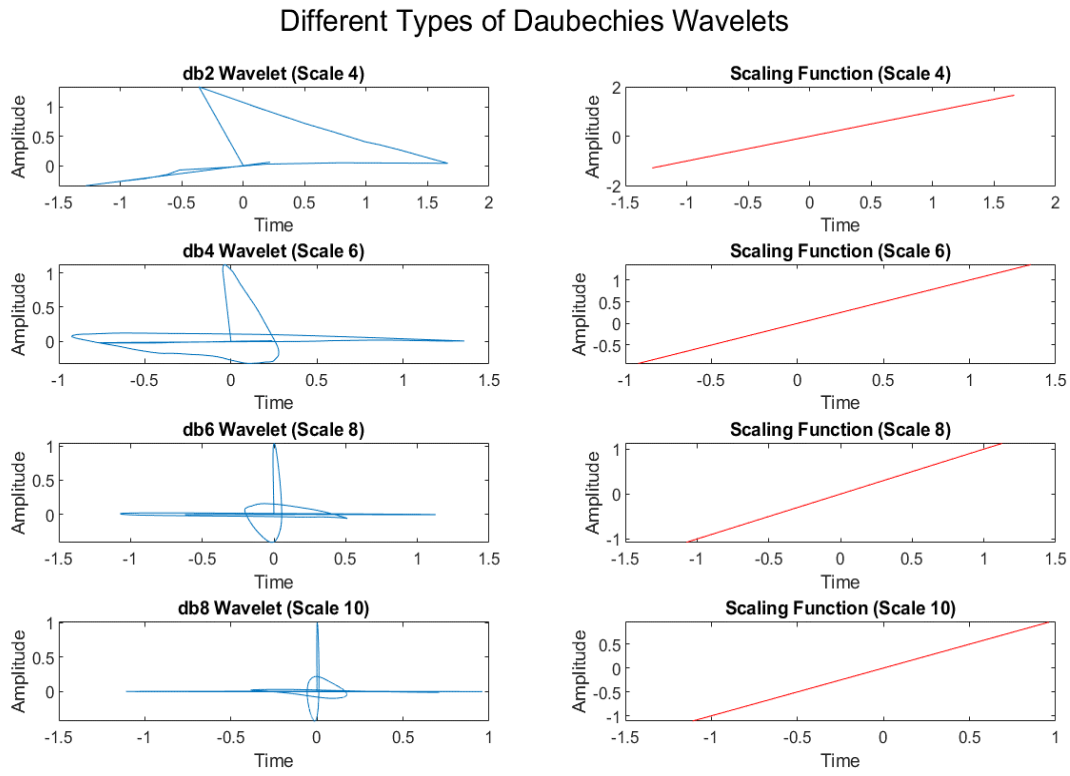

Figure III-20 Different Types of Daubechies Wavelets

Filter banks play a crucial role in the practical application of the Discrete Wavelet Transform (DWT), a method that deconstructs signals into various frequency components.

In the context of the DWT method, a pair of filters known as the low-pass filter (LPF) and the high-pass filter (HPF) form a filter bank. The LPF allows the passage of low-frequency elements, while the HPF permits the transmission of high-frequency components. This process is combined with down-sampling, which reduces the sampling rate and provides a more concise representation of the original signal [94].

Mathematically by sampling the Continuous Wavelet Transform (CWT) on a dyadic grid, which means selecting specific values for translation and scale parameters [95]. In this context, 'n' and 'm' are sets of positive integers, and 'N' represents the number of

samples. These parameters are discretized to create the DWT. Once discretized, the wavelet function can be defined as follows:

$$DWT_{\varphi}(j,k) = \int_{-\infty}^{\infty} s(t)\varphi_{j,k}^{*}(t)dt. \quad \text{(III.7)}$$

In this equation, ' j represents the scale parameter indicating the scale of the wavelet function, k denotes the translation parameter specifying the shift of the wavelet function, and $\psi_{(j,k)}(t)$ signifies the complex conjugate of the wavelet function with ψ scale j and translation k. The integral computes the inner product between the signal s(t) and the complex conjugate of the wavelet function $\psi_{j,k}$ over the entire range of t, facilitating the calculation of the discrete wavelet transform (DWT). The outcome DWT $\psi_{j,k}$ provides information about the signal's decomposition at scale j and translation k, aiding in signal analysis and feature extraction.

For the wavelet function $\psi_{(j,k)}(t)$, it is defined as:

$$\varphi_{(j,k)}(t) = 2^{(2/j)}\varphi\left(2^{j}t - k\right) \quad \text{(III.8)}$$

The DWT breaks down a signal into its coarse and detailed components by applying a sequence of high-pass and low-pass filtering operations based on the following equations:

$$y_{high}(k) = \sum_{n} s(n).h(2k-n) \quad \text{(III.9)}$$

$$y_{low}(k) = \sum_{n} s(n).h(2k-n) \quad \text{(III.10)}$$

In equations (III.9) and (III.10), $y_{high}(k)$, $y_{low}(k)$ represent the results of applying high-pass and low-pass filters, respectively, to the signal s(n), followed by down sampling

by a factor of 2. These operations are performed using the impulse responses h(n) and g(n). Specifically, $y_{high}(k)$ is obtained by convolving s(n) with the impulse response h(2k−n), while $y_{low}(k)$ is obtained by convolving s(n) with the impulse response g(2k−n). The summation is carried out over all samples n. The coefficients obtained from the low-pass filter are referred to as "approximation" coefficients (A), while those from the high-pass filter are called "detail" coefficients (D). The detail coefficients $b_{(j,k)}$ can be calculated using the following equation:

$$b_{(j,k)} = \int s(t)\varphi^*_{j,k}\,(t)dt \tag{III.11}$$

Where $\psi_{(j,k)}$ represents the wavelet functions defined as:

$$\varphi_{(j,k)}(t) = 1/\left(\sqrt{2j}\right)\ \varphi\left(\frac{t-k2^j}{2^j}\right) \tag{III.12}$$

Similarly, the approximate coefficients $C_{(j,k)}$ are calculated as:

$$c_{j,k} = \int s(t)\emptyset^*_{j,k}\,(t)dt \tag{III.13}$$

Where $\emptyset_{(j,k)}$ are the scaling functions, defined as:

$$\emptyset_{(j,k)}(t) = 1/\left(\sqrt{2j}\right)\ \emptyset\left(\frac{t-k2^j}{2^j}\right) \tag{III.14}$$

The discrete inverse transform f(t) is computed by summing the translated and dilated wavelets, weighted by their respective coefficients:

$$f(t) = \sum_{j,k} b_{j,k}\varphi_{j,k}(t) \tag{III.11}$$

The DWT is a valuable tool for obtaining a multi-resolution representation of a signal, making it useful for real-time signal analysis.

Following the passage of the signal through a low-pass filter, the outcome yields approximation coefficients that encapsulate the low-frequency components of the signal. Subsequently, the signal undergoes a down sampling operation by a factor of two, effectively halving the number of samples in the signal. This deliberate reduction in sample count serves the purpose of mitigating computational complexity and accelerating the overall algorithmic process [96].

In a similar vein, once the signal has traversed through a high-pass filter, the resultant detail coefficients emerge, characterizing the high-frequency components of the signal. Simultaneously, the signal undergoes another round of down sampling by a factor of two, effectively diminishing the sample count. This strategic down sampling operation is employed to curtail the number of samples and facilitate computational efficiency

This down-sampling equation constitutes a crucial cornerstone within the DWT algorithm, serving to partition a signal into its distinct frequency components across multiple resolution tiers. Its significance resonates across diverse domains, finding application in areas like image processing, signal compression, and feature extraction [88]. Through the partitioning of frequency constituents into approximation and detail coefficients, the DWT expands the signal's representation across multiple scales. The low-pass filter (LPF) facilitates the transmission of low-frequency elements, whereas the high-pass filter (HPF) accommodates the high-frequency components. Consequently, the DWT excels at detecting and scrutinizing both overarching patterns and intricate particulars within a signal. This duality in frequency components gives

rise to a spectrum of applications, including signal compression, noise mitigation, feature extraction, and data analysis.

Multiresolution Analysis (MRA) constitutes a fundamental aspect of the Discrete Wavelet Transform (DWT), allowing the decomposition of a signal into different levels of detail. These decomposition levels in the DWT correspond to distinct scales or resolutions of the signal, capturing specific frequency components or details (Figure III-21) [97].

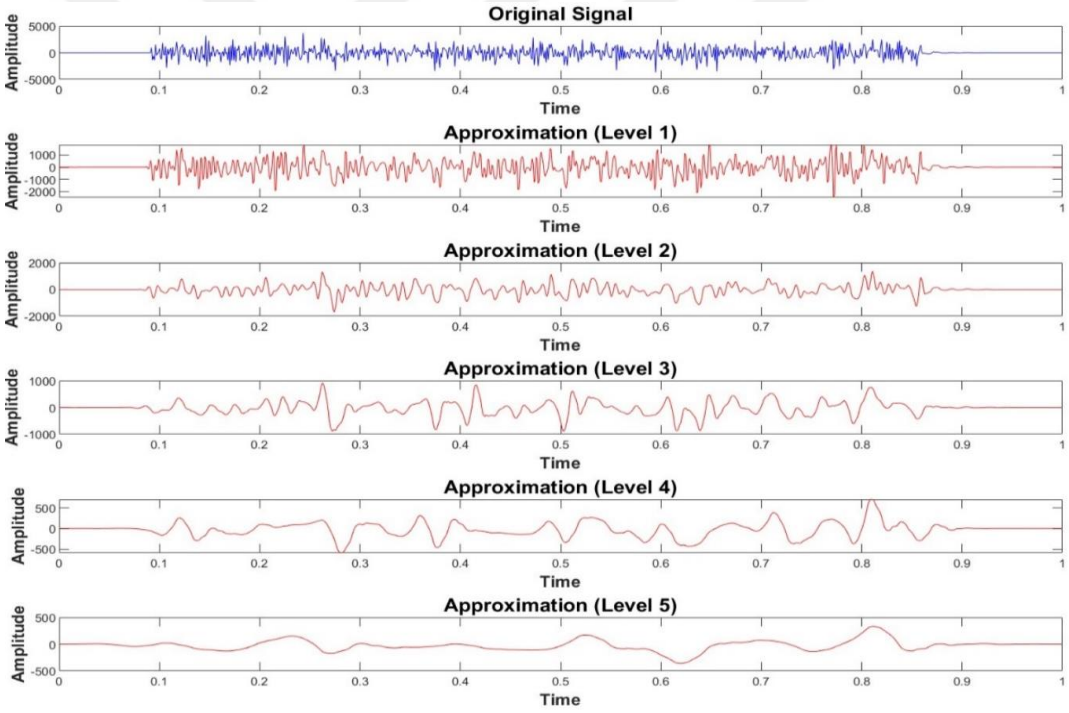

Figure III-21 Discrete Wavelet Five Levels (Wavelet Db4)

The hierarchical structure of the DWT enables analysis at multiple scales, providing a comprehensive understanding of the signal. In our project, we specifically applied a 5-level DWT using the 'db4' wavelet to signals of length 1000. The extracted coefficients at each level are as follows Level 1 - 503 coefficients, Level 2 - 251 coefficients, Level

3 - 125 coefficients, Level 4 - 62 coefficients, and Level 5 - 31 coefficients. These coefficients encompass both detail and approximation coefficients from all levels of the wavelet decomposition. The approximation coefficients represent the low-frequency components or the coarse approximation of the signal, while the detail coefficients capture the high-frequency components or the details of the signal [97]. The decomposition process involves convolving the signal with a low-pass filter (scaling filter) and a high-pass filter (wavelet filter), followed by down sampling to reduce the resolution [98]. This process is repeated iteratively to obtain multiple levels of decomposition, each representing a different scale or level of detail [98]. The concept of MRA is particularly useful in various applications, including computer vision, signal coding, texture discrimination, edge detection, matching algorithms, and fractal analysis. By analyzing the signal at different scales, MRA allows for the extraction of relevant information at each level, enabling more efficient and effective analysis [97]. For example, in computer vision, analyzing images at each resolution level would be redundant, and it is more efficient to focus on the additional details available at higher resolutions. MRA provides a framework for this selective analysis of different levels of detail. The hierarchical structure of MRA also facilitates the interpretation of resolution and scale concepts [97]. The decomposition into different scales or levels of detail allows for a better understanding of the signal's characteristics at different resolutions. This understanding is crucial in various fields, such as climate teleconnection studies, where the analysis of signals at different temporal scales is essential. MRA-based methods enable the examination of interdependence patterns between climate signals and the identification of temporal variability in precipitation [99].

S. In summary, MRA is a fundamental concept in the DWT that enables the decomposition of a signal into different levels of detail. The decomposition levels represent different scales or resolutions of the signal, allowing for analysis at multiple scales. This hierarchical structure facilitates the selective analysis of relevant information and provides a better understanding of the signal's characteristics at different resolutions. MRA has applications in various fields, including computer vision, signal coding, and climate teleconnection studies.

3.4.1.4. Discrete wavelet transforms MATLAB Implementation

The implementation of the discrete wavelet transforms (DWT) in MATLAB using the "wavedec" function is a widely used technique for signal processing and analysis. The "wavedec" function is utilized to decompose a signal into its wavelet coefficients at different levels of resolution. To employ the "wavedec" function, users need to specify the desired wavelet and the number of decomposition levels. The wavelet is a mathematical function that allows for the analysis of the signal at various scales. Different wavelets possess distinct properties and are suitable for different types of signals. For instance, the Daubechies wavelet is commonly chosen for its favorable time-frequency localization properties. The number of decomposition levels determines the level of detail in the decomposition. A higher number of levels provides more detailed information about the signal, but also increases the computational complexity. Once the signal is decomposed using the "wavedec" function, the resulting wavelet coefficients can be utilized for various purposes such as denoising, feature extraction, or signal reconstruction. These coefficients capture the different features present in the signal at different scales. The approximation coefficients represent the low-frequency

components of the signal, while the detail coefficients represent the high-frequency components. These coefficients can be further processed or analyzed based on the specific application requirements. In addition to the "wavedec" function, MATLAB provides other functions for working with wavelet coefficients, such as "wrcoef" for signal reconstruction. The MATLAB Wavelet Toolbox offers a range of wavelet families and functions to cater to different applications, allowing users to select the most suitable wavelet for their specific needs. In summary, the implementation of the discrete wavelet transforms in MATLAB using the "wavedec" function provides a powerful tool for signal processing and analysis. It enables the decomposition of signals into wavelet coefficients at different levels of resolution, providing valuable information about the signal's frequency content and features. These coefficients can then be utilized for various applications such as denoising, feature extraction, or signal reconstruction.

3.4.2. Wavelet Scattering

Wavelet scattering is a widely applicable mathematical technique that finds utility across various research domains. It enables the creation of translation-invariant representations for signals and images, thereby proving valuable for tasks such as texture classification, image recognition, and signal analysis [100]. The scattering transform is constructed through a series of wavelet convolutions and modulus operations, which facilitate the computation of the wavelet transform's magnitude. This transform exhibits stability in the presence of time-warping deformations and possesses the ability to capture transient phenomena. Notably, wavelet scattering has demonstrated promising outcomes in diverse applications, including music genre and

phone classification, texture discrimination, rainfall classification, and the analysis of meteorological data. By providing translation-invariant representations and stable features, wavelet scattering emerges as a powerful tool for signal and image analysis in various domains [101].

3.4.2.1. Theory and principles

In essence, wavelet scattering mirrors a deep convolutional network (Figure III-22), meticulously crafted through a cascade of wavelet modulus nonlinearities and low-pass filters. This construction empowers the derivation of low-variance features from real-valued time series and image data, all while demanding minimal configuration. The outcomes are representations endowed with the attributes of translation invariance and robustness against the distortions induced by time warping [102]. Its efficacy in diverse classification tasks and its capacity to achieve state-of-the-art results, even with limited datasets [103], further underline the significance of wavelet scattering in the realms of machine learning and deep learning applications [104].

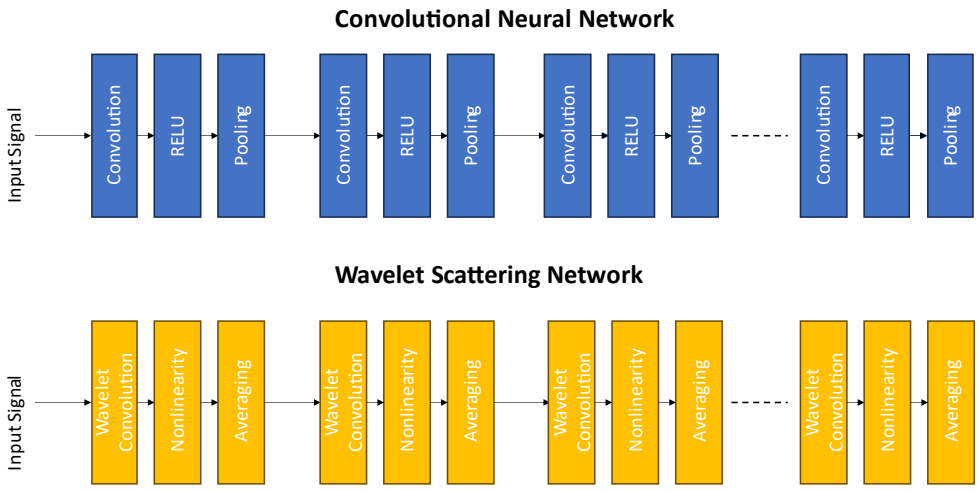

Figure III-22 Wavelet Scattering Vs Convolutional Neural Network

This permits the description of short-lived events like assaults and changes in signal amplitude [100]. The foundation of wavelets in signal analysis is tied to the notion of multiresolution analysis (MRA), a structured approach for forming orthonormal wavelet bases [105]. Wavelets offer a way to represent signals with focus on specific time and frequency segments. This enables the examination of signal attributes across various scales. Breaking down a signal into wavelet coefficients at different scales allows for a comprehensive grasp of both the signal's particular traits and its broader properties. Going beyond the conventional wavelet analysis, wavelet scattering broadens this approach by introducing multiple stages of wavelet convolutions and modulus operations. This permits the retrieval of insights into the signal's modulation spectrum across various scales and orientations. The scattering transform maintains local translation invariance, ensuring resilience against distortions in the timing of the signal [100]. Additionally, it furnishes constants that remain unaffected by changes in scale, shearing, and minor distortions [101]. Such characteristics endow wavelet scattering with substantial utility in diverse signal analysis contexts. The dissection of signals into multiple scales and orientations within wavelet scattering emerges through a series of linked wavelet convolutions and modulus operations. Wavelet convolutions apprehend the signal's frequency attributes across varied scales, while the modulus operators draw out amplitude particulars. By linking these processes, wavelet scattering adeptly grasps both the signal's frequency and amplitude fluctuations over diverse scales and orientations. The integration of wavelet scattering into signal analysis has unveiled encouraging outcomes across diverse fields. Instances include its role in categorizing musical genres and distinguishing phone characteristics [100].

Furthermore, this technique has found application in distinguishing textures, yielding leading-edge classification outcomes across texture databases marked by uncontrolled viewing conditions [101]. Moreover, the utilization of wavelet scattering extends to the classification of electromagnetic signals, showcasing traits of translation invariance and resistance to deformation [106]. In summary, wavelet scattering is a signal analysis technique that utilizes wavelet convolutions and modulus operators to decompose signals into different scales and orientations. It provides a locally translation invariant representation that is stable to time-warping deformations and captures transient phenomena. Wavelet scattering has been successfully applied in various domains, including music classification, texture discrimination, and electromagnetic signal classification

3.4.2.2. Mathematical formulations

Scattering in the First Order Within the domain of Continuous Wavelet Transform (CWT), the acquisition of first-order scattering coefficients involves a fundamental procedure. This procedure entails the convolution of the modulus of the CWT with a low-pass filter, a technique commonly referred to as "temporal averaging." This strategic averaging process not only introduces time-shift variance but also guarantees stability when confronted with time-warping deformations.

Mathematically, the first-order scattering coefficients, denoted as $Sx(t, \lambda_1)$, are defined as the modulus of the convolution between the input signal x and the first-order wavelet function $\varphi\lambda_1$. Subsequently, a convolution operation with a low-pass filter Φ is applied [100].

$$Sx(t, \lambda^1) = |x * \varphi\lambda^1| * \Phi \qquad \text{(III.12)}$$

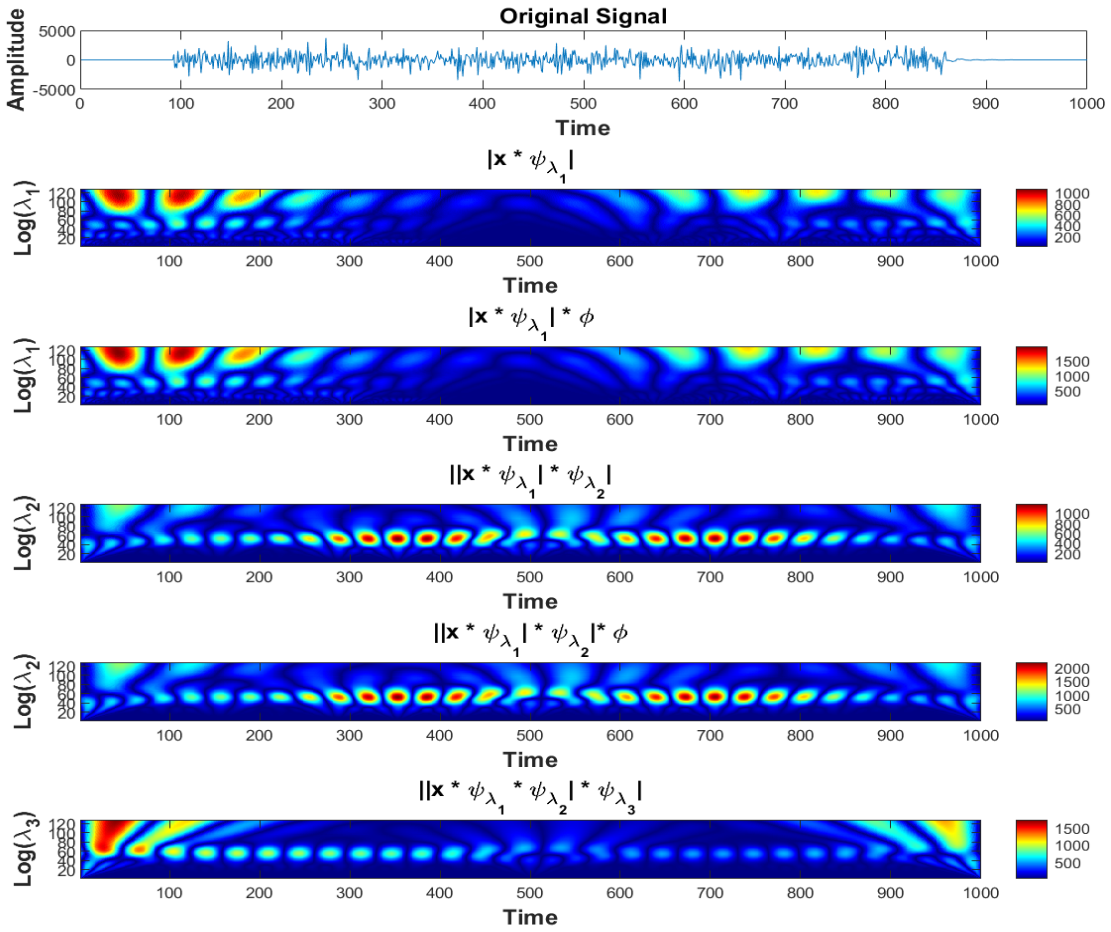

Figure III-23 Scattering Convolution Process

Here, the parameter λ_1 signifies the center frequency of the first-order wavelets, and the convolution is executed between the low-pass filter and each individual row of the Continuous Wavelet Transform (CWT).

The crux of this operation can be intuitively grasped as the act of sliding a wavelet across the signal, computing its modulus ($|x * \varphi\lambda_1|$), and subsequently convolving it with Φ. This convolution with Φ serves as a mechanism for averaging on the modulus, ultimately yielding an invariant feature representation. This comprehensive process, recognized as the wavelet scattering transform, encompasses the sequential steps of sliding the wavelet over the signal, calculating the modulus, and then culminating with

an additional convolution operation utilizing Φ, which assumes the role of a low-pass filter (Figure III-23).

In Second Order Scattering The second-order scattering procedure extends the methodology by performing convolution on the modulus of the first-order scattering coefficients with second-order wavelets [100].

In mathematical terms, this process is expressed as:

$$W_2 x(t, \boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}) = |\boldsymbol{x} * \varphi_{\lambda 1}| * \varphi_{\lambda 2} \quad \text{(III.13)}$$

In this equation, the first-order scattering coefficients Sx(t, λ_1) are treated as the new input (x), and they undergo convolution with a set of second-order wavelets $\varphi\lambda_2$. This convolution is subsequently followed by an additional round of low-pass averaging.

$$S_2 x(t, \boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}) = |\boldsymbol{x} * \varphi_{\lambda 1}| * \varphi_{\lambda 2}| * \emptyset \quad \text{(III.14)}$$

Higher-Order Scattering The framework can be further expanded to encompass higher-order scattering, where coefficients are derived through successive convolutions. This involves convolving wavelets with the modulus of coefficients obtained at the previous order, followed by modulus computation and low-pass filtering:

$$S_m x(t, \boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_m}) = |\boldsymbol{x} * \varphi_{\lambda 1}| * \varphi_{\lambda \mathrm{m}}| * \emptyset \quad \text{(III.15)}$$

Wavelet Scattering Energy The selection of wavelets spanning diverse scales and their application by sliding across the signal ensures the coverage of distinct segments within the frequency spectrum. Notably, this procedure conserves energy, meaning that the energy observed in the time domain aligns with the energy present in the frequency domain. This energy preservation property forms a fundamental aspect of wavelet scattering (Figure III-24).

Dilated wavelets$\varphi_{\boldsymbol{\lambda}}(t) = 2^{-\frac{1}{Q}}\varphi\left(2^{-\frac{j}{Qt}}\right), with\ \boldsymbol{\lambda} = \mathbf{2}^{-\frac{j}{Q}}$ (III.20)

Dilated Wavelets Dilated wavelets, characterized as φλ(t), are strategically designed to encompass a wide range of scales. This feature empowers them to adapt effectively to various frequency ranges, thereby contributing to the creation of a comprehensive feature representation [100].

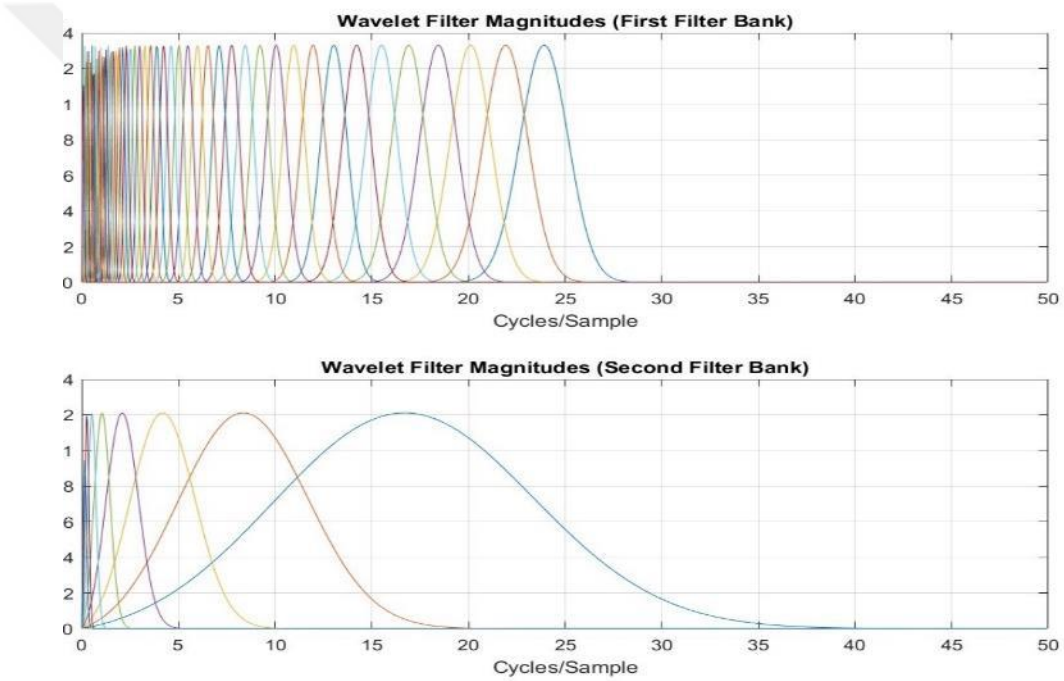

Figure III-24 Wavelet Scattering Filter Bank

Wavelet Transform The wavelet transforms, denoted as Wx(t), involves the convolution of the input signal x with both the original wavelet φ(t) and the dilated wavelets φλ(t) across various scales denoted as λ.

Wavelet transform Wx(t)=$\{x * \varphi(t), x * \varphi_{\boldsymbol{\lambda}}(t)\}_{\lambda}$ (III.21)

$$\text{If } \left|\widehat{\emptyset}\ (w)\right|^2 + \sum_{\lambda} |\hat{\varphi}\lambda\ (w)|^2 = 1 \tag{III.22}$$

Then is unitary

$$\left||Wx|\right|^2 = \left||x * \varphi|\right|^2 + \sum_{\lambda}\left||x * \varphi_\lambda|\right|^2 = \left||x|\right|^2 \tag{III.23}$$

Unitary Property: The unitary property is achieved when the sum of squared coefficients from the low-pass filter and squared coefficients of dilated wavelets equals 1. This property ensures the preservation of energy within the transformed signal. (Figure III-25).

Wavelet Scattering Feature Extraction: The wavelet scattering transform entails a sequence of critical operations, including convolution, modulus calculation, and low-pass filtering. The coefficients obtained from these operations are judiciously downsampled, effectively reducing computational complexity. In our specific application, 20 invariant features derived from this process yield 588 features per signal, enabling a detailed analysis. These coefficients, characterized by their interpretability and visualizability, form the essence of scattering features (see Figure III-25).

Paths in Scattering: Within the context of the scattering transform, the term "paths" delineates sequences of operations that are systematically applied to the input signal. These paths effectively capture crucial hierarchical relationships and underlying features inherent in the signal representation [107] (Figure III-25).

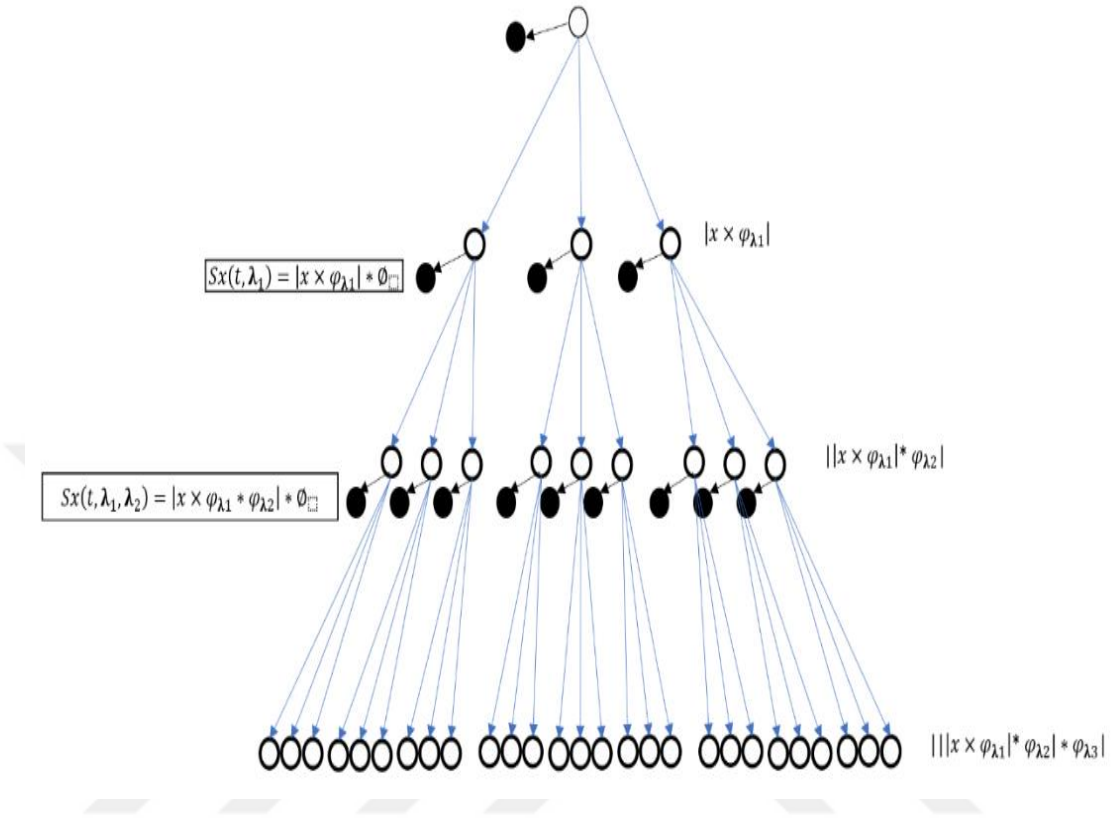

Figure III-25 Wavelet Scattering Tree

In summation, the wavelet scattering transforms represent a formidable approach for feature extraction, encompassing the intricate interplay of wavelet convolutions, modulus operations, and low-pass filtering. These operations are designed to capture the quintessential characteristics of a signal while simultaneously ensuring stability and invariance against deformations. The resulting scattering coefficients hold substantial utility across a diverse spectrum of signal analysis and processing endeavor.

3.4.2.3. Implementation of Wavelet Scattering in MATLAB

The implementation of wavelet scattering in MATLAB involves utilizing functions such as "waveletscattering" to compute the scattering coefficients. Wavelet scattering is a technique used for image and signal processing tasks, providing a translation-

invariant and stable representation of data that is robust to deformations and preserves high-frequency information. The wavelet scattering network consists of cascaded wavelet transform convolutions with nonlinear modulus and averaging operators. This network computes a translation-invariant image representation that is stable to deformations and can be used for tasks such as texture discrimination and handwritten digit classification. To use the wavelet scattering function in MATLAB, users need to specify the wavelets and the number of levels for the scattering transform. The choice of wavelets depends on the specific application and the desired properties of the scattering coefficients. Different wavelet families, such as Morlet or Haar wavelets, can be employed. The number of levels determines the depth of the scattering network and affects the level of detail captured in the representation. The wavelet scattering function in MATLAB has been applied to various applications, including glaucoma detection in retinal fundus images. In this application, the wavelet image scattering network developed in MATLAB is used to perform the scattering decomposition on the images, and the resulting scattering coefficients are utilized as features for automatic diagnosis. It is worth noting that there are alternative implementations of wavelet scattering available in different programming languages. For instance, the Kymatio software package provides a Python implementation of the scattering transform in 1D, 2D, and 3D. This implementation is compatible with modern deep learning frameworks and can be executed on both CPU and GPU, offering improved performance. In conclusion, the implementation of wavelet scattering in MATLAB involves using functions like "waveletscattering" to compute the scattering coefficients. The choice of wavelets and the number of levels determines the properties of the scattering transform. Wavelet

scattering has been successfully applied to various applications, such as glaucoma detection, and there are also alternative implementations available in other programming languages.

3.5. Feature Selection and Optimization

3.5.1. Principal Component Analysis (PCA)

is a widely-used dimensionality reduction technique that helps transform the original feature space into a new set of orthogonal variables known as principal components. This process is valuable in various applications, including machine learning, as it can alleviate issues such as multicollinearity and enhance model generalization [108]. Here's a breakdown of how PCA is typically performed

Covariance Matrix Computation PCA begins by calculating the covariance matrix from the original feature data. The covariance matrix summarizes the relationships between the different features in the dataset [108].

The covariance matrix for a set of features, denoted as X, with n samples and m features, can be computed as follows:

$$Var(\hat{\beta}) = (X^T X)^{-1}\sigma^2 \quad \text{(III.24)}$$

Where Var ($\hat{\beta}$) represents the variance-covariance matrix of the coefficient estimates $\hat{\beta}$ This matrix captures the uncertainty and interrelationships among the estimated coefficients. The equation involves several key components: X, is the design matrix of the predictor variables; X^T, is the transpose of the design matrix. σ^2 is the variance of the error term (residuals) in the linear regression model. The term $(X^T X)^{-1}$ is the inverse of the matrix product of the transpose of the design matrix and the design matrix itself.

Eigenvalue and Eigenvector Calculation Next, PCA calculates the eigenvalues and corresponding eigenvectors of the covariance matrix. These eigenvectors represent the directions in the original feature space along which the data varies the most, and the eigenvalues represent the variance explained by each of these directions [108].

The equation for calculating eigenvectors and eigenvalues is:

$$X\widehat{\beta}^{ridge} = \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2 + \lambda} u^T{}_j \, y \qquad \text{(III.25)}$$

The equation for computing eigenvectors and eigenvalues in ridge regression, denoted by Equation (III.25), offer s a method to estimate coefficients $\widehat{\beta}^{ridge}$ considering regularization. In this equation, X represents the design matrix of predictor variables, and y represents the response variable. The summation over p features encompasses the contribution of each eigenvector (u_j), where (d_j^2) denotes the square of the jth eigenvalue. Additionally, λ serves as the regularization parameter, controlling the balance between data fitting and coefficient magnitude. The equation scales the impact of each eigenvector by combining it with its associated eigenvalue and the regularization parameter. By summing these scaled contributions, $\widehat{\beta}^{ridge}$ is computed, providing a means to estimate the coefficients in ridge regression while considering the influence of eigenvectors, eigenvalues, and regularization.

Choosing the Number of Principal Components To determine the optimal number of principal components to retain, various methods can be used, such as scree plots and explained variance. Scree plots show the eigenvalues in decreasing order, and the point at which the eigenvalues begin to level off indicates a suitable number of principal

components to retain (Figure III-26). Explained variance (Figure III-27) helps assess how much information is preserved by each principal component [109].

Reducing Dimensionality Once the number of principal components is decided, the dataset is transformed into a reduced feature set using the selected principal components. This reduction helps mitigate the curse of dimensionality and can enhance the performance of machine learning models [109].

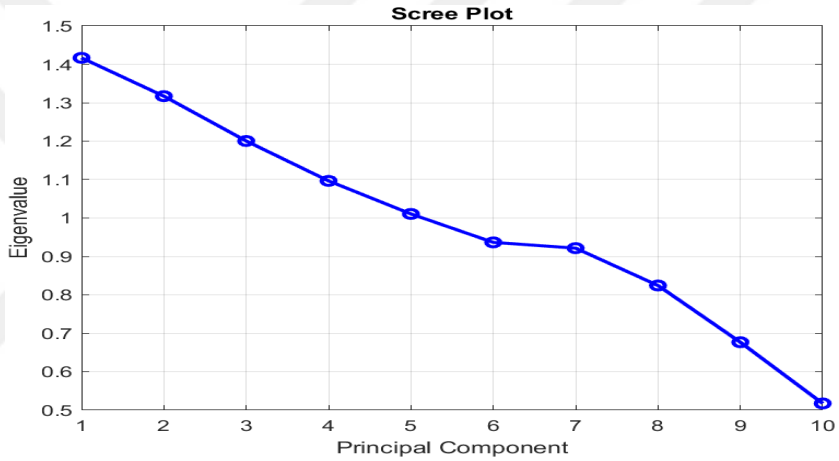

Figure III-26 Scree Plot

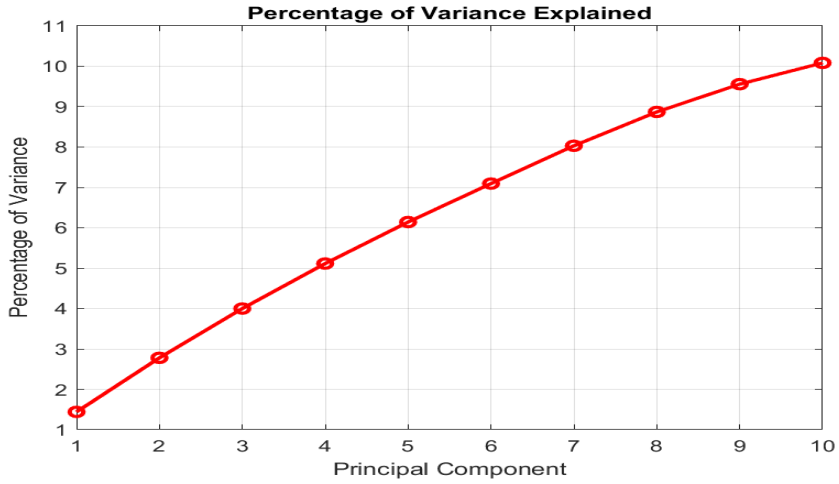

Figure III-27 Percentage of Variance

Here's a MATLAB code snippet for applying PCA to reduce dimensionality

```
% apply PCA to reduce dimensionality
numDims = 50; % set desired number of dimensions
scTrain = reshape(scTrain, [], size(scTrain, 3))';
scTest = reshape(scTest, [], size(scTest, 3))';
[coeff,score,~,~,explained] = pca([scTrain; scTest]);
scoreTrain = score(1:size(scTrain,1), 1:numDims);
scoreTest = score(size(scTrain,1)+1:end, 1:numDims);
```

3.5.2. Linear Discriminant Analysis (LDA)

is a powerful supervised dimensionality reduction technique, primarily designed for classification tasks. It aims to maximize the separation between different classes in a dataset while minimizing the variance within each class. In our analysis, we leveraged LDA to identify the most discriminatory features, rank them based on LDA coefficients, and seamlessly integrate them into our classification models to potentially enhance classification accuracy and model robustness.

3.5.2.1. Here's an explanation of the key steps in applying LDA

Executing LDA with Class Labels LDA operates in a supervised manner, considering class labels. It identifies features that contribute the most to separating different classes. Feature Ranking with LDA Coefficients LDA coefficients are used to rank and select features that significantly aid in defining class boundaries. Features with higher LDA coefficients are considered more important for classification.

Integration into Classification Models The features refined through the LDA process are integrated into our classification models. This integration aims to improve the classification accuracy and overall robustness of our models.

Below is the MATLAB code snippet for applying LDA:

```
% Apply Linear Discriminant Analysis (LDA) on the training data for this
fold

ldaModel = fitcdiscr(FoldTrainFeatures, FoldTrainLabels);

% Apply LDA to the training and test data for this fold
FoldTrainFeaturesLDA = predict(ldaModel, FoldTrainFeatures);
FoldTestFeaturesLDA = predict(ldaModel, FoldTestFeatures);

% Convert LDA-transformed features to a numeric matrix
FoldTrainFeaturesLDA = double(FoldTrainFeaturesLDA);
```

in this code, ldaModel represents the trained LDA model, and FoldTrainFeaturesLDA and FoldTestFeaturesLDA contain the LDA-transformed features for the training and test data, respectively. The main equation regarding LDA is based on finding the optimal linear projection that maximizes the between-class scatter while minimizing the within-class scatter

3.5.3. Sequential Feature Selection (SRE) and Recursive Feature Elimination (RFE)

Sequential Feature Selection (SRE) and Recursive Feature Elimination (RFE) are two iterative approaches to feature selection, both aiming to enhance the importance of selected features in a machine learning model. Here's an overview of each technique

3.5.3.1. Sequential Feature Selection (SRE)

SRE begins by ranking features based on relevant scoring metrics that indicate their importance or relevance to the problem.

It then iteratively includes the most pertinent features, one at a time, in a sequential manner.

The goal is to create a subset of features that optimally contribute to model accuracy and generalizability.

Here's the MATLAB code for SRE

```
% Compute SRE for training set
sreTrain = computeSRE(scTrain);
% Compute SRE for test set
sreTest = computeSRE(scTest);
```

3.5.3.2. Recursive Feature Elimination (RFE)

RFE takes a different approach by employing specific machine learning algorithms (e.g., decision trees) to consecutively rank and eliminate the least influential features. Features are eliminated one at a time based on their impact on model performance.

In our experiment, utilizing Recursive Feature Elimination (RFE), we aimed to identify a subset of features maximizing model accuracy and generalizability while reducing dimensionality, employing a total of 10 features.

Throughout these iterative processes, careful monitoring of model performance is essential to ensure that the chosen feature subsets positively contribute to model accuracy and robustness.

Here's the MATLAB code for RFE:

```
% Perform RFE
selectedFeatures = rfe(X_train, Y_train', numFeatures);
% Keep only the selected features in the training and testing data
TrainFeaturesSelected = cell(size(TrainFeatures));
for i = 1:numel(TrainFeatures)
    TrainFeaturesSelected{i} = TrainFeatures{i}(selectedFeatures);
end
TestFeaturesSelected = cell(size(TestFeatures));
for i = 1:numel(TestFeatures)
    TestFeaturesSelected{i} = TestFeatures{i}(selectedFeatures);
end
```

3.6. Validation and Evaluation

In the aftermath of feature selection, we meticulously undertook the validation and evaluation of our machine learning models. These evaluations were conducted with the aid of pertinent metrics, including accuracy, precision, recall, and F1-score. This judicious assessment ensured that the optimized feature subsets indeed yielded models that aligned with our predefined objectives effectively.

Comparison and Conclusion:

To summarize, the strategic application of feature selection methodologies, encompassing PCA, LDA, SRE, and RFE, assumed a pivotal role in the optimization of our machine learning and data analysis pipeline. These methodologies facilitated dimensionality reduction, augmentation of feature relevance, and the amelioration of model performance. It is important to acknowledge that each methodology possesses its unique strengths and limitations. Their synergistic application, however, endowed our research project with valuable insights and significantly contributed to its success.

3.7. Fault Detection and Classification

Fault detection and classification (FDC) represent pivotal processes in numerous industries, safeguarding the reliability and safety of systems, machinery, and operations. In the domain of machine learning, two powerful methodologies, namely Support Vector Machines (SVM) and Neural Networks (NN), have emerged as indispensable tools for FDC applications.

Support Vector Machines (SVM) are esteemed for their robustness in delineating data into distinct classes. SVMs are exceptionally skilled at identifying optimal boundaries that maximize the separation margin between different data points. This capability

renders them well-suited for the task of fault detection, enabling them to address a wide array of scenarios involving both linear and non-linear data separation [110].

In contrast, Neural Networks (NN), particularly deep neural networks, have ushered in a new era of fault detection by harnessing the intricacies present within data patterns. NNs, constructed with interconnected layers of artificial neurons, excel in capturing intricate relationships and dependencies, making them adept at detecting even subtle faults that might elude traditional detection methods [111].

Furthermore, we will navigate through the merits and demerits of SVMs and NNs in the context of FDC, offering guidance on the judicious selection of each technique based on the problem at hand. By the culmination of this chapter, you will possess a comprehensive understanding of how to implement SVMs and NNs for fault detection and classification, equipping you to effectively address real-world challenges and uphold system reliability and safety.

3.7.1. The Purpose of Using SVM and Neural Network

In this research, Support Vector Machines (SVM) and Neural Networks (NN) are employed to classify faults in the tie rod. The tie rod is a critical component of a vehicle's steering system, and accurately detecting and classifying faults in the tie rod is crucial for vehicle safety and performance.

The research objective is to develop a fault detection and classification system for the tie rod. The hypothesis is that SVM and NN can effectively classify tie rod faults based on different levels of fault severity. This classification allows for targeted maintenance actions based on the severity of the fault.

Two tests are conducted in this research. In the first test, SVM and NN are employed to classify tie rod faults into five levels, ranging from normal to severe faults (Figure III-28). In (Figure III-28), the diagonal cells represent the count or frequency of correctly classified instances. Additionally, it is observed that one fault in level 1 is predicted as normal. This detailed classification provides a better understanding of the tie rod's condition and facilitates appropriate maintenance actions.

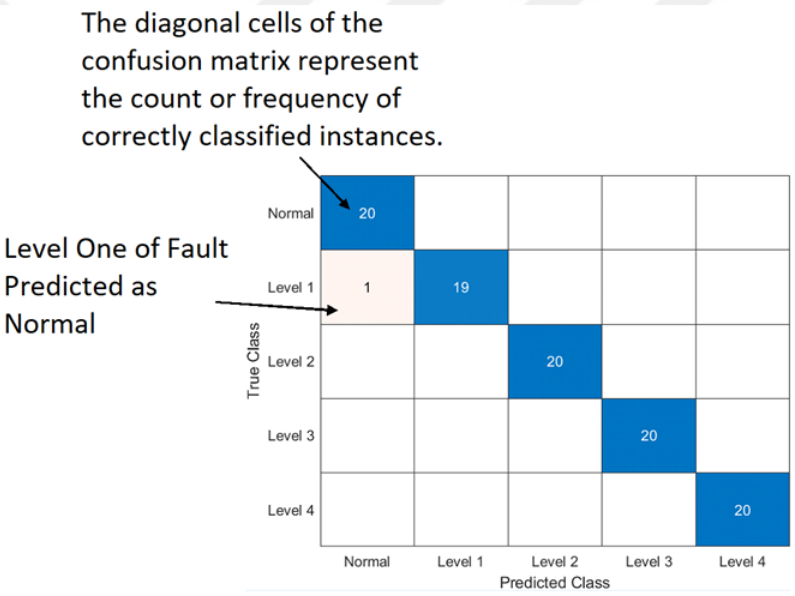

Figure III-28 SVM Four Faulty Levels Classification

In the second test, SVM and NN models are employed to classify the tie rod's condition as either 'faulty' or 'normal' when encountering various road obstacles. Four types of road obstacles are considered sinewave road obstacles, roughness road obstacles, pothole road obstacles, and bump road obstacles. By analyzing the tie rod's response to these obstacles, SVM and NN can classify the tie rod's condition as normal or abnormal, indicating the presence of a fault (Figure III-29). For example, in (Figure III-29), the diagonal cells represent correctly predicted instances, and there is an error in the bumpy

road obstacle signals where two instances were incorrectly classified as faulty when they should have been classified as normal.

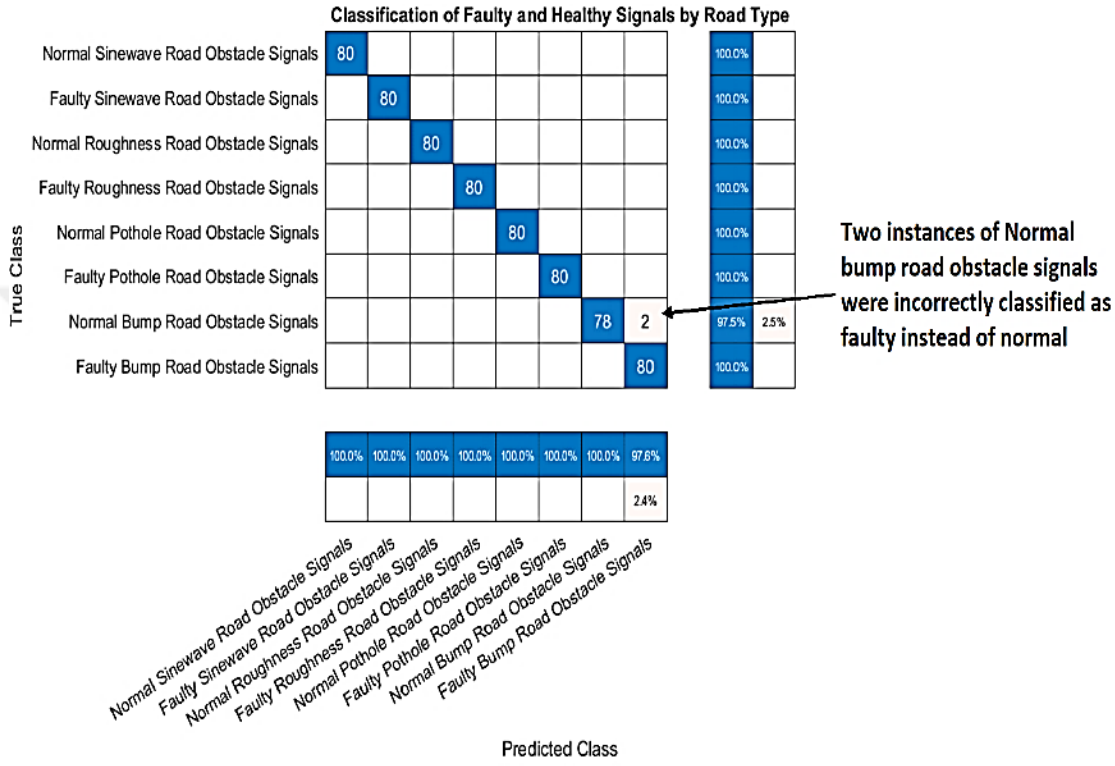

Figure III-29 SVM Obstacle Types and Fault Classifications

SVM and the Neural Network are chosen for this research due to their abilities to handle high-dimensional data and nonlinear relationships. They work by finding an optimal hyperplane (in the case of SVM) or a complex network of interconnected nodes and layers (in the case of Neural Network) that maximally separates different classes in the feature space. By training both classifiers with a dataset of tie rod responses to various fault levels and road obstacles, they can learn the patterns and characteristics associated with each class. These trained models can then be used to classify new instances of tie rod responses, accurately identifying the fault level or abnormality.

In conclusion, the purpose of using SVM and Neural Network (NN) in this research is to develop a fault detection and classification system for the tie rod. By employing these machine learning techniques, we can accurately classify faults and abnormalities, enabling timely maintenance actions to ensure vehicle safety and performance. The use of SVM and NN aligns with the research objectives by providing robust and accurate classification methods for tie rod faults, enhancing our ability to maintain and optimize vehicle systems.

3.7.2. Support Vector Machines (SVM)

is a popular machine learning algorithm used for classification and regression tasks. It works by finding the best hyperplane that separates the data into different classes. The choice of the best hyperplane is based on maximizing the separation or margin between the two classes [112].

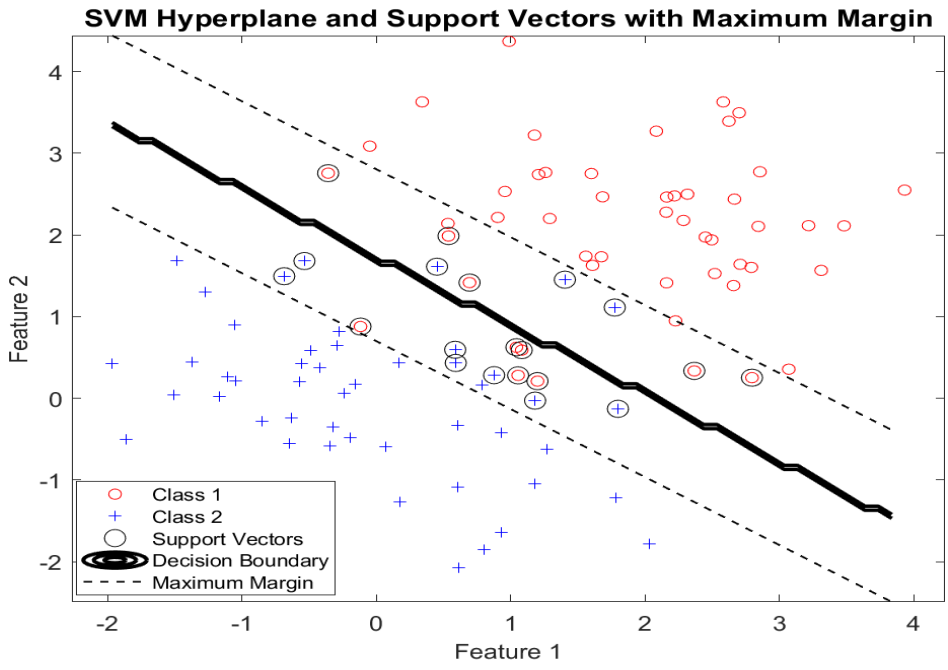

Figure III-30 Hyperplane and Support Vectors with Maximum Margin

In the case of linearly separable data, SVM selects the hyperplane that has the maximum distance from the nearest data point on each side. This hyperplane is known as the maximum-margin hyperplane or hard margin (Figure III-30). The goal is to find a hyperplane that can classify new data points accurately. SVM is robust to outliers, meaning it can ignore outliers and still find the best hyperplane [113],[112].

In cases where the data is not linearly separable, SVM introduces soft margins. Soft margins allow for some misclassification of data points, but penalize violations of the margin (Figure III-31). The SVM algorithm tries to minimize the hinge loss, which is a commonly used penalty function. The hinge loss is proportional to the distance of the violation from the margin [113].

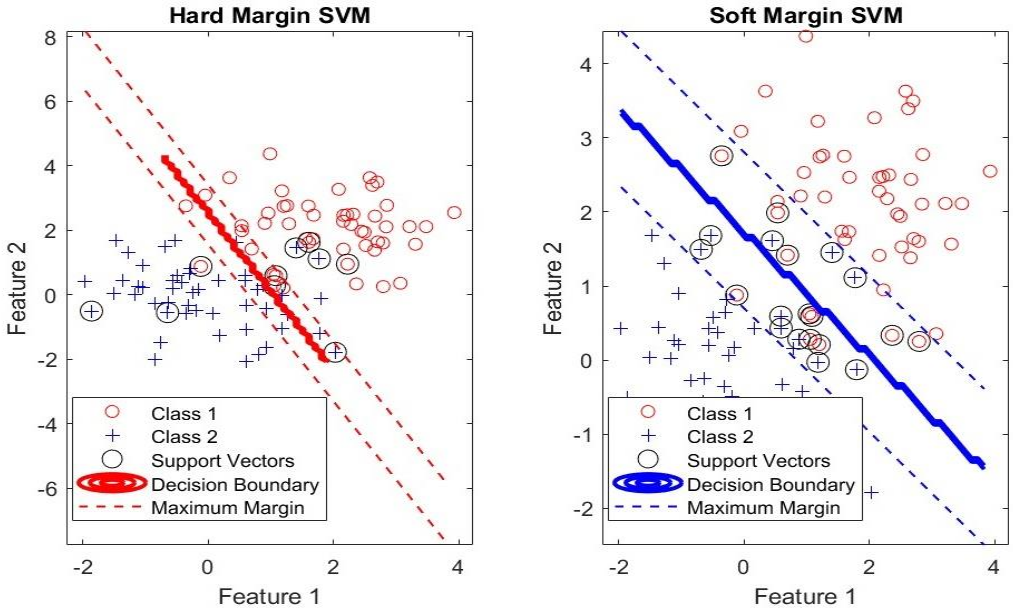

Figure III-31 Hard Margin SVM Vs Soft Margin SVM

However, in real-world scenarios, data is often not linearly separable. SVM addresses this issue by using a technique called kernelization. It maps the original data to a higher-dimensional space where it becomes linearly separable [114]

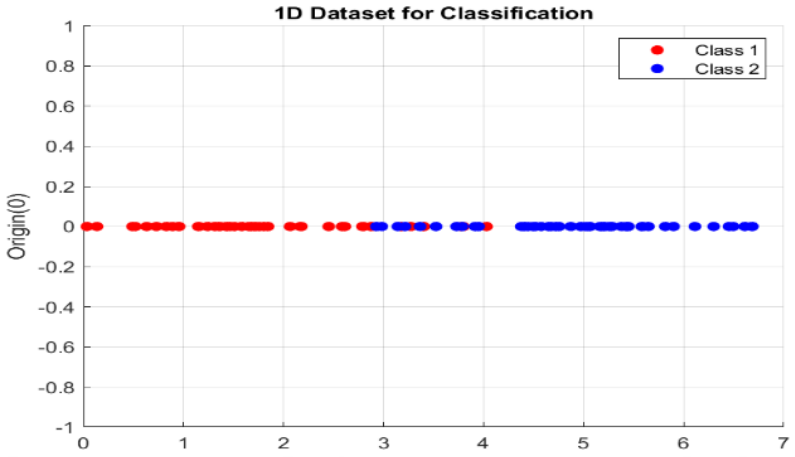

Figure III-32 1D Dataset for Classification

Let's consider the data shown in the Figure III-32. In the context of Support Vector Machines (SVM), we solve this problem by creating a new variable using a kernel. For simplicity, let's denote a point as xi on the line, and we create a new variable yi as a function of the distance from the origin, which we can visualize on a (Figure III-33).

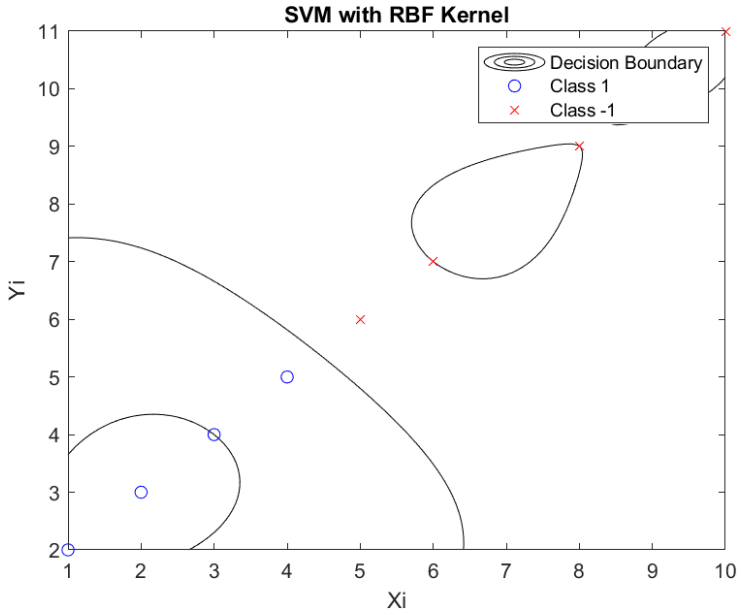

Figure III-33 SVM with RBF Kernel

In this case, the new variable y is generated as a function of the distance from the origin. Such a non-linear function that creates a new variable is referred to as a kernel.

The process of mapping the data to a higher-dimensional space is achieved through the use of a kernel function. A kernel function is a non-linear function that transforms the original data into a new feature space. Commonly used kernel functions include polynomial kernels, Gaussian kernels, and sigmoid kernels [115].

3.7.2.1. Support Vector Machine Terminology

Hyperplane: A hyperplane serves as the decision boundary that separates data points belonging to different classes within a feature space. In the context of linear classification, it is represented by the equation $wx + b = 0$.

Support Vectors: Support vectors are crucial data points that are nearest to the hyperplane, playing a pivotal role in determining both the hyperplane's position and the margin [116].

Margin: The margin is defined as the distance between the hyperplane and the support vectors. In SVM, the primary goal is to maximize this margin because a wider margin generally indicates better classification performance.

Kernel: A kernel is a mathematical function employed in SVM to transform original input data points into higher-dimensional feature spaces. This transformation enables the identification of a hyperplane even when the data points are not linearly separable in the original input space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels [116].

Hard Margin: The hard margin hyperplane refers to a hyperplane that effectively separates data points of different classes without any misclassifications. It is the ideal scenario where the margin is maximized.

Soft Margin: In situations where data is not perfectly separable or contains outliers, SVM allows for a soft margin approach. This introduces slack variables for each data point, relaxing the strict margin requirement and permitting some degree of misclassification or violations. It seeks a balance between increasing the margin and minimizing violations.

C: The regularization parameter, denoted as C, is a crucial factor in SVM. It balances the trade-off between margin maximization and misclassification penalties. A higher value of C imposes a stricter penalty, resulting in a smaller margin and potentially fewer misclassifications [116].

Hinge Loss: SVM commonly employs the hinge loss as its loss function. It penalizes misclassifications and margin violations. The objective function in SVM is typically formulated by combining the hinge loss with a regularization term. In our Model we set the maximum number of objective function evaluations to 100.

Dual Problem: The dual problem of SVM optimization involves finding the Lagrange multipliers associated with the support vectors. Solving the dual problem offers advantages, including the ability to utilize kernel tricks and more efficient computation [117].

3.7.2.2. Mathematical Intuition of Support Vector Machine (SVM)

In the context of a binary classification problem with classes labeled as +1 and -1, SVM seeks to find a decision boundary in the form of a hyperplane, which can be expressed mathematically as

$$w^T x + b = 0 \quad \text{(III.26)}$$

Here, 'w' represents the normal vector to the hyperplane, indicating the direction that is perpendicular to the hyperplane. The parameter 'b' represents the offset or distance of the hyperplane from the origin along this normal vector 'w'.

The distance between a data point 'x_i' and the decision boundary can be calculated using

$$d_i = \frac{(w^T x + b))}{\|w\|} \quad \text{(III.27)}$$

$$d_i = \frac{(w^T x + b))}{\|w\|} \quad \text{(III.28)}$$

Here, '||w||' denotes the Euclidean norm of the weight vector 'w' [118].

For a Linear SVM classifier, the classification rule is defined as

$$\hat{y}\begin{cases} 1\ (w^T x + b) \geq 0 \\ 0\ (w^T x + b) < 0 \end{cases} \quad \text{(III.29)}$$

In the case of a Hard Margin Linear SVM classifier, the optimization objective is to minimize the square of the Euclidean norm of 'w' while ensuring that all training instances are correctly classified and lie at a distance of at least 1 from the decision boundary [118]

$$minimize_{w,b} \frac{1}{2} w^T\, w = minimize_{w,b} \frac{1}{2} ||w^2|| \quad \text{(III.30)}$$

$$Subject\ to\ y_i(w^T x_i + b) \geq 1 \quad for\ i = 1,2,3 \ldots . m \quad \text{(III.31)}$$

In this context, 't_i' represents the target variable or label for the 'i'-th training instance. 't_i' is -1 for negative occurrences (when 'y_i' = 0) and 1 for positive instances (when 'y_i' = 1). The constraint ensures that the decision boundary separates the classes with a margin of at least 1.

$$t_i(w^T x_i + b) \geq 1 \tag{III.32}$$

$$t_i(w^T x_i + b) \geq 1 \tag{III.33}$$

For a Soft Margin Linear SVM classifier, a regularization term is introduced to allow for some misclassification (soft margin), and the optimization problem becomes

$$minimize_{w,b} \frac{1}{2} w^T w + c \sum_{i=1}^{m} \zeta_i \tag{III.34}$$

$$Subject\ to\ y_i(w^T x_i + b) \geq 1 - \zeta_i\ and\ \zeta_i \geq 0\ for\ i = 1,2,3 \ldots . m \tag{III.35}$$

Here, 'C' controls the trade-off between maximizing the margin and minimizing the classification errors [118].

Dual Problem

To solve the SVM optimization problem, a dual problem is often formulated to find the Lagrange multipliers 'α(i)' associated with the support vectors [118].. The dual objective function to be maximized is

$$Maximize_{\alpha}\ \frac{1}{2} \sum\nolimits_{1 \to m} \sum_{j \to m} \alpha i \alpha j t i t j K(xi, xj) - \sum\nolimits_{i \to m} \alpha i \tag{III.36}$$

Here

'αi' is the Lagrange multiplier associated with the 'i'-th training sample.

'K(xi, xj)' is the kernel function, which calculates the similarity between two samples 'xi' and 'xj,' allowing SVM to handle nonlinear classification by implicitly mapping samples into a higher-dimensional feature space.

Once the dual problem is solved, the SVM decision boundary can be described in terms of the optimal Lagrange multipliers and the support vectors [118]. The support vectors are the training samples with 'αi > 0,' and the decision boundary is determined by

$$W = \sum_{i \to m} \alpha i t i K(xi, x) + b \quad \text{(III.37)}$$

$$ti(w^T xi - b) = 1 \Leftrightarrow b = w^T xi - ti \quad \text{(III.38)}$$

3.7.2.3. Hyperparameter Selection

The process of selecting hyperparameters is pivotal in constructing an effective SVM model. Notable SVM hyperparameters include

Regularization parameter (C): This parameter balances the trade-off between maximizing the margin and minimizing classification errors. Smaller 'C' values prioritize a larger margin but may allow for some misclassification, while larger 'C' values emphasize correct classification but risk overfitting [119].

Kernel type and parameters: In the case of kernel SVMs, the choice of kernel (e.g., linear, polynomial, RBF) and their associated parameters (e.g., degree for polynomial, gamma for RBF) must be determined [119].

Hyperparameter selection can be accomplished using techniques such as cross-validation and grid search

Cross-validation: The dataset is split into training and validation sets. For each set of hyperparameters, the model is trained on the training set and assessed on the validation set. The hyperparameters that yield the best validation performance are chosen [119].

Grid search: A grid of potential hyperparameter values is defined, and the model's performance is systematically evaluated on the validation set for all combinations of hyperparameters. The combination with the best performance is selected [119].

3.7.2.4. Model Training

To train an SVM model, the following steps are followed

Data Preprocessing This includes tasks like feature scaling, handling missing values, and encoding categorical variables [120].

Data Splitting The dataset is partitioned into training and testing sets (or cross-validation sets) [121].

SVM Type Selection Choose between linear or kernel SVM based on the data's characteristics.

Hyperparameter Tuning Utilize techniques like cross-validation or grid search to identify optimal hyperparameters (C and kernel parameters) [122].

Model Training Fit the SVM model to the training data using the selected hyperparameters .

Model Evaluation Assess the model's performance on the testing/validation dataset.

Considerations such as dealing with imbalanced classes or outliers should be taken into account during the training process [123].

3.7.2.5. Model Evaluation

The selection of evaluation metrics depends on the problem type (binary or multiclass classification, regression) and the specific objectives. Common metrics for assessing SVM models include:

Accuracy: Measures overall classification correctness.

Precision: Calculates the ratio of true positive predictions to total predicted positives, assessing the model's ability to minimize false positives.

Recall (Sensitivity): Computes the ratio of true positive predictions to total actual positives, gauging the model's capacity to capture all positive instances.

F1-score: Provides a balance between precision and recall by calculating their harmonic mean.

Confusion matrix: A tabular summary of true positive, true negative, false positive, and false negative predictions [123].

The choice between cross-validation and holdout validation hinges on factors such as dataset size, available computational resources, and the need for robust performance estimation. Cross-validation is often favored for its ability to provide a more reliable evaluation.

Ultimately, the choice of evaluation metrics and validation strategy should align with the machine learning task's objectives and the dataset's characteristics.

The implementation of SVM in MATLAB is facilitated by the SVM MATLAB function, which allows for training and testing SVM models [124]. The SVM MATLAB function supports different types of SVM, including C-SVM, nu-SVM, and k-NN [125].

In MATLAB, the SVM MATLAB function can be used to optimize the performance of SVM models. This can be done by selecting appropriate parameter values and incorporating cross-validation techniques. The running time performance of different SVM variants can be measured using the tic and toc functions in MATLAB [123].

The MATLAB code that used for SVM is accordingly

```
% Train the SVM model on the training data
model = fitcecoc(TrainFeatures, YTrain);
% Train and tune the SVM model using the training data
t = templateSVM('Standardize',true);
SVMModel = fitcecoc(TrainFeatures, YTrain, 'Learners', t,
'FitPosterior',true,...
'OptimizeHyperparameters', {'BoxConstraint','KernelScale'},...
'HyperparameterOptimizationOptions', optimizationOptions, 'Verbose', 1,...
'HyperparameterOptimizationOptions', struct('AcquisitionFunctionName',...
'expected-improvement-plus', 'MaxObjectiveEvaluations', 20,
'UseParallel',true), 'Options', statset('UseParallel',true));
% Predict the labels of the test data using the tuned model
YPred = predict(SVMModel, TestFeatures);
```

To further optimize the performance of SVM, the MATLAB code for SVM tuning can be equipped with a brute force method to estimate a large number of parameter values and select the optimized values for generating the best model. Additionally, the SVM MATLAB code can incorporate a fivefold cross-validation for model evaluation [126].

In summary, the SVM MATLAB function provides a convenient way to implement SVM models in MATLAB. It supports different types of SVM and can be used in various applications. The performance of SVM can be optimized by selecting appropriate parameter values and incorporating cross-validation techniques. The MATLAB implementation of SVM has been utilized in fields such as robotics and medical imaging, and it can be further optimized by incorporating brute force methods and cross-validation techniques.

3.7.3. Neural network

3.7.3.1. Long Short-Term Memory (LSTM) Networks

LSTM networks represent a specific category within recurrent neural networks (RNNs), and their architecture is tailored for handling sequential data, including but not limited to time series data and natural language text. The fundamental mechanisms of LSTMs can be elucidated as follows

Neurons and Layers: Within LSTM networks, the architectural building blocks are recurrent cells, which can be organized into multiple layers (Figure III-34). These cells differ significantly from standard feedforward neurons due to their intricate internal structure, encompassing gating mechanisms designed to regulate the flow of information.

Input Layer: The input layer is responsible for receiving the raw data or features from the dataset. Each neuron within this layer represents a distinct feature [127].

Hidden Layers: Intermediate layers in neural networks, positioned between input and output layers, are crucial for pattern extraction. They perform mathematical operations on data, with complexity determined by the number of hidden layers. In your project, you've opted for an exceptionally large number of hidden layers - a staggering 1500 layers. Each of these hidden layers contains neurons with weight arrays matching the previous layer's neuron count. This abundance of hidden layers exponentially increases both training and evaluation times.

This extraordinary depth in our Neural Network architecture, while unconventional, can be useful for tackling highly complex and nonlinear data. It exemplifies the core principles of deep learning, which relies on deep neural networks to autonomously learn

intricate patterns and structures, often without the need for labeled data. Such depth has found applications in various domains, including computer vision and language processing.

Output Layer: The output layer generates the ultimate output of the network, which might encompass classification labels, regression values, or other types of predictions [128].

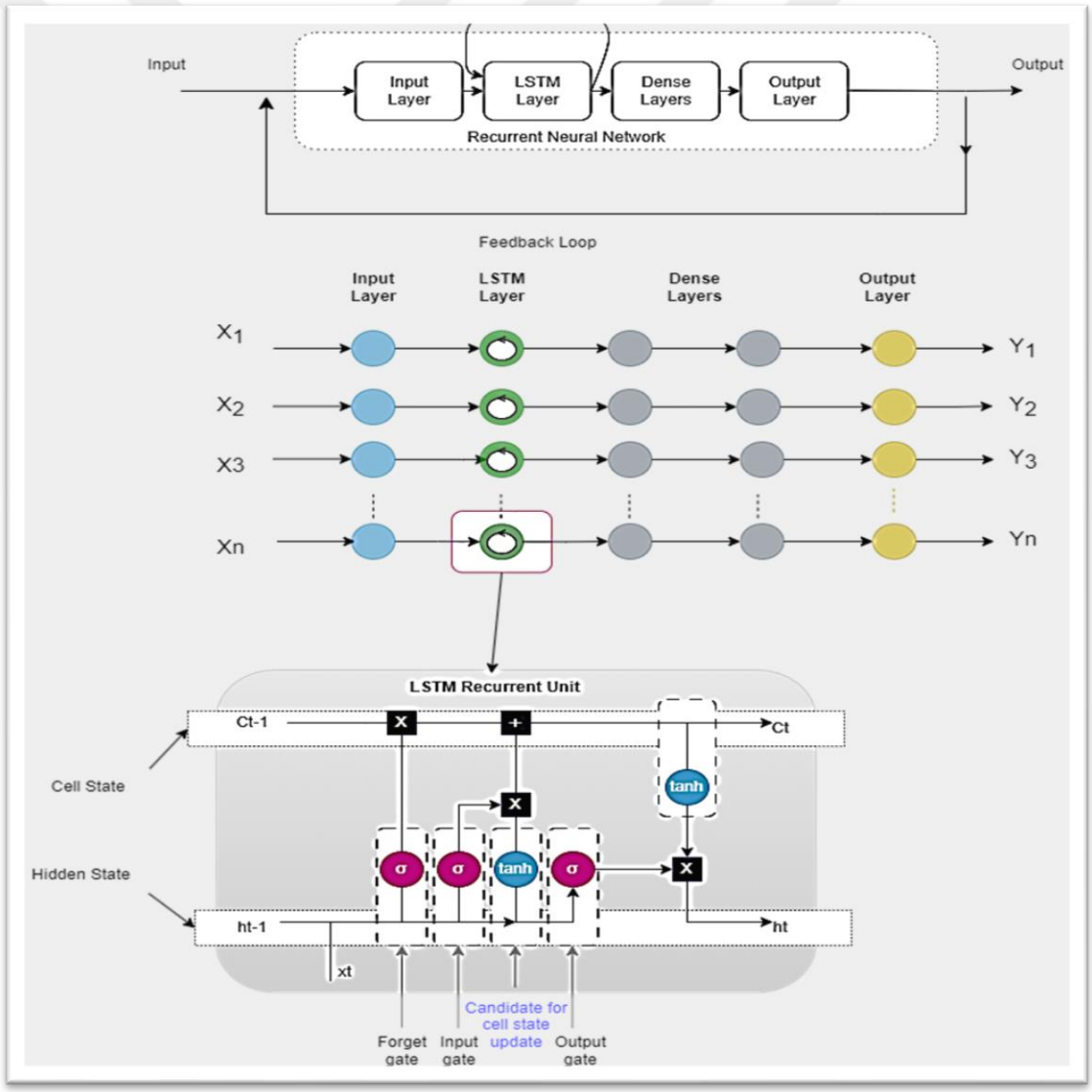

Figure III-34 LSTM Neural Network Architecture

Cell Memory, Input Gate, Forget Gate, Output Gate LSTMs exhibit a unique architecture consisting of specialized components

Cell Memory: The central feature distinguishing LSTMs is the presence of a dedicated cell memory, which assumes a critical role in preserving information over extended sequences, allowing the network to capture enduring dependencies within the data [129]. In an LSTM diagram, this cell memory is represented as a horizontal line stretching from ct-1 to ct, symbolizing the short-term memory of the cell (Figure III-35).

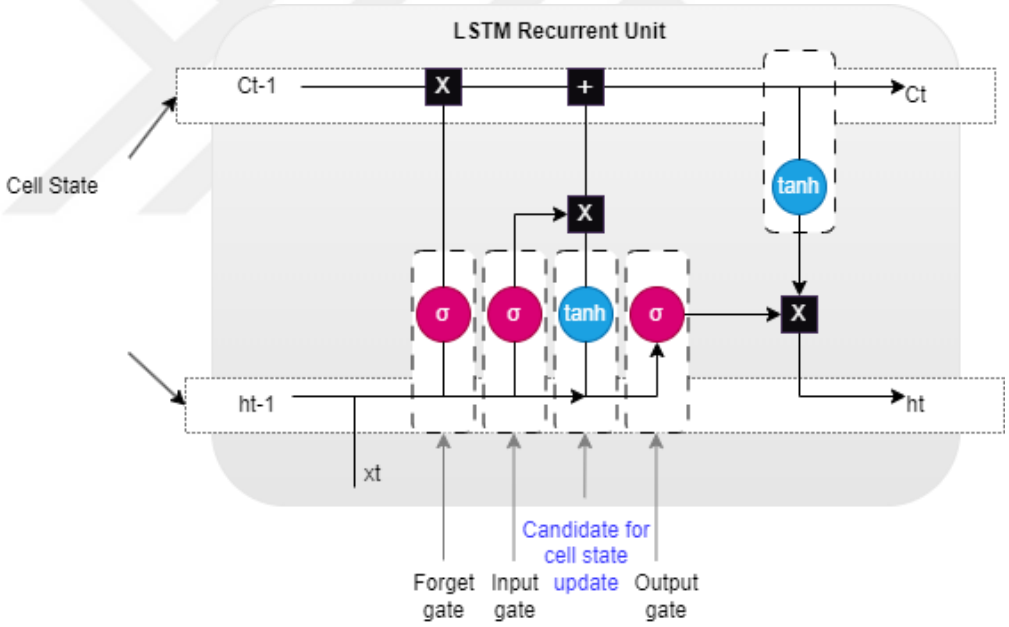

Figure III-35 LSTM Recurrent Unit and Gates

Input Gate: The "Input Gate" in an LSTM cell regulates the flow of new data into the cell's memory. It decides which information is relevant and should be stored, adding it to the cell memory (Figure III-36). This process involves sigmoid and tanh gates that control the input's impact on the cell state, with "i" controlling the extent of influence.

Additionally, both the previous output ("h") and current input ("x") have individual weights for precise control over their contributions [129].

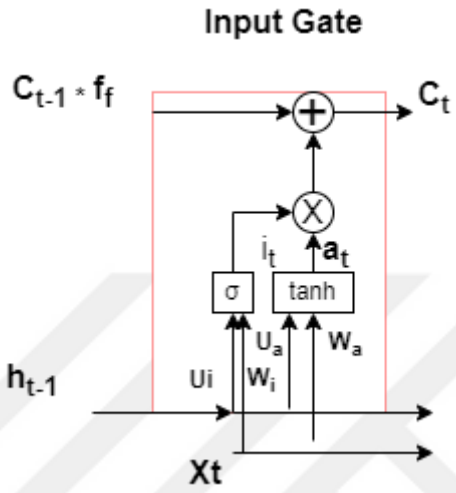

Figure III-36 LSTM Input Gate

$$i_t = \sigma\left(W_i * X_t + U_i * h_{(t-1)} + b_i\right) \quad \text{(III.39)}$$

$$a_t = tanh\left(W_a * X_t + U_a * h_{(t-1)} + a\right) \quad \text{(III.40)}$$

$$C_t = C_{(t-1)} * f_f + i_t * a_t \quad \text{(III.41)}$$

"I" determines the extent of "a's" impact on "c" within an LSTM cell. The hyperbolic tangent's (-1 to +1) range enables "a" to either decrease or increase "c," with the level of influence dictated by "i." Additionally, it's worth noting that both "h" and "x" possess distinct weights for both "i" and "a."

Forget Gate: The "Forget Gate" is a pivotal component within LSTM cells, responsible for orchestrating the selective removal or "forgetting" of information from the cell memory that is considered irrelevant or outdated (Figure III-37). This discerning amnesia capability empowers LSTMs to concentrate on the most pertinent data [129].

Operational details include the "σ" (sigmoid) function, which ranges from 0 to 1, governing the extent of information removal from the cell state. By adjusting the value of "σ," we control the degree of forgetfulness [129]. Specifically, both the current input (xt) and the previous output (ht-1) undergo multiplication by "σ."

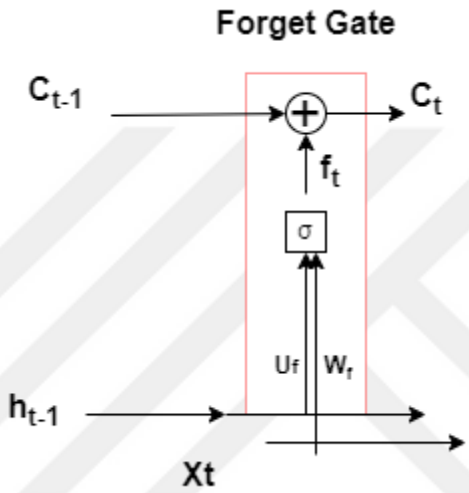

Figure III-37 LSTM Forget Gate

Operational details include the "σ" (sigmoid) function, which ranges from 0 to 1, governing the extent of information removal from the cell state. By adjusting the value of "σ," we control the degree of forgetfulness. Specifically, both the current input (xt) and the previous output (ht-1) undergo multiplication by "σ."

$$f_t = \sigma\left(W_f * X_t + U_f * h_{(t-1)} + b_f\right) \quad \text{(III.42)}$$

$$C_t = C_{(t-1)} * f_t + i_t * a_t \quad \text{(III.43)}$$

These operations involve weights (Wf and Uf), a bias (bf), and element-wise multiplication, symbolized by the blue circle with a cross. Notably, "h" and "x" possess their own sets of weights, further enhancing the network's adaptability for different tasks and temporal dependencies across time slots (t and t-1).

Output Gate: The "Output Gate" assumes a crucial role within the LSTM cell by determining which information stored in the cell memory (c) should contribute to the output (h) at a given time step (Figure III-38). This gate ensures that the LSTM yields outputs that are contextually relevant and suited to the task [129].

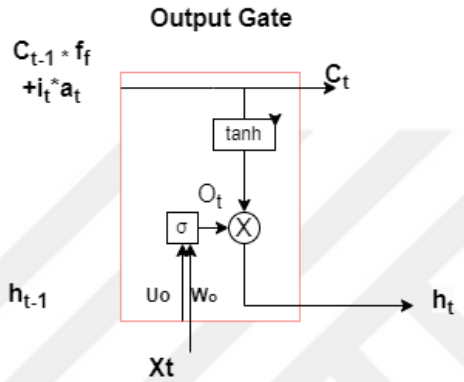

Figure III-38 LSTM Output Gate

$$O_t = \sigma\left(W_o * X_t + U_o * h_{(t-1)} + b_o\right) \quad \text{(III.44)}$$

$$h_t = tanh(C_t) + O_t \quad \text{(III.45)}$$

The output (h) is derived from the cell state (c) via the tanh function, which can modulate c to be positive or negative within the range of -1 to +1. The degree to which this tanh(c) affects h is precisely controlled by "o." The value of "o" is computed using distinct weights for the previous output (ht-1) and the current input (xt), each determined by a sigmoid function [129].

It's worth noting that there exists a variant of LSTM known as the Gated Recurrent Unit (GRU), which lacks an output gate but incorporates a reset gate and an update gate for managing information flow.

Connections and Weights

LSTM cells have internal weights and connections that govern the flow of information (Figure III-39). These weights are initialized using techniques like Xavier/Glorot initialization or He initialization, which are better suited for deep networks [130].

Activation Functions

LSTM cells employ various activation functions internally, including sigmoid and hyperbolic tangent (tanh) functions (Figure III-39). These functions are used to control the operations of the gates (input, forget, output), ensuring that information flow is managed appropriately. Additionally, the Rectified Linear Unit (ReLU) activation function is frequently utilized for processing the output of LSTM cells [131].

Forward Propagation

Forward propagation in LSTM networks involves processing sequential data one time step at a time. At each time step, the input and the previous cell's state are used to update the cell's internal state and produce an output.

Loss Function

The choice of a loss function depends on the specific task. For example, Mean Squared Error (MSE) can be used for regression tasks, while Cross-Entropy is suitable for sequence classification tasks.

Backpropagation

Backpropagation in LSTM networks is extended through time (BPTT) due to the sequential nature of data. Gradients are calculated and adjusted for each time step to train the network effectively [132].

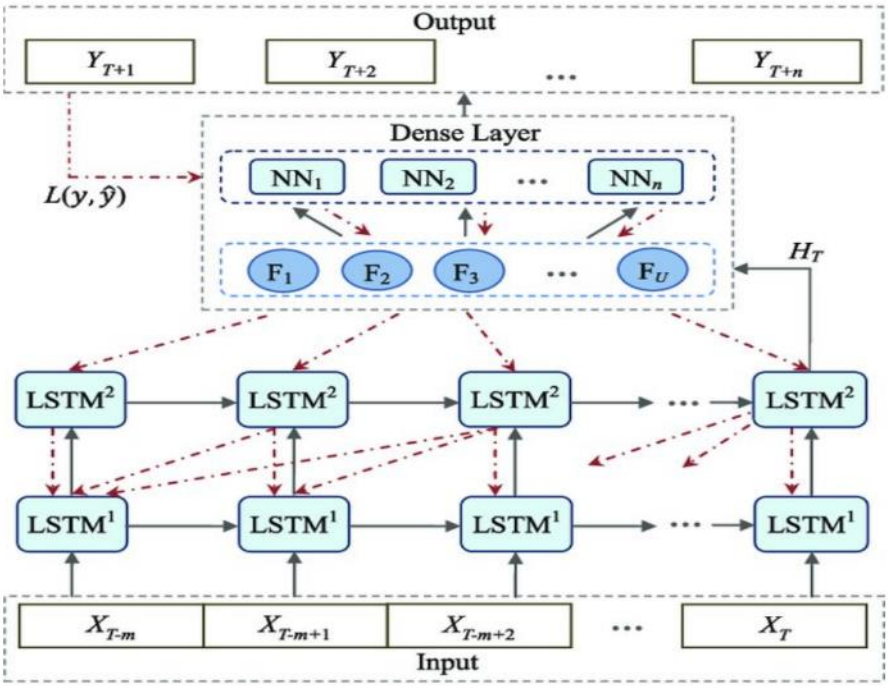

Figure III-39 Forward, Loss Function and Backward Propagation (This Figure by [133], [134]).

3.7.3.2. Optimization

Common optimization algorithms like Adam, RMSprop, or stochastic gradient descent (SGD) with learning rate annealing are used to update weights during training [134].

3.7.3.3. Training

Training LSTM networks involves handling sequential data efficiently, including techniques like sequence padding, batching, and potentially using teacher forcing or attention mechanisms for sequence-to-sequence tasks [135].

Bias Term

Each LSTM cell typically has its own bias terms for various gates and activations, which play a crucial role in the network's adaptability [136].

Learning Rate (α)

Selecting an appropriate learning rate (α) is critical for training, as it affects the convergence of the network [136].

Teacher Forcing

For sequence-to-sequence tasks, teacher forcing can be used during training to improve convergence by providing ground-truth data at each time step [135].

In summary, LSTM networks are specialized for sequential data and use complex LSTM cells with gating mechanisms for learning and encoding sequential patterns. Understanding the specifics of LSTM architecture and training is essential when working with sequences. Be sure to refer to LSTM-specific resources and tutorials for more in-depth information.

3.7.3.4. Mathematical Background behind LSTM RNNs

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to effectively model and capture dependencies in sequential data. LSTMs are particularly valuable for tasks involving time series analysis, natural language processing, and other sequential data processing tasks.

Forward Propagation in LSTM

LSTM cells operate through a series of gates and activation functions, allowing them to selectively retain and update information over time [137]. The key components of forward propagation in an LSTM include

Input Gate (i(t))

In this phase, we proceed to update the input gate of the LSTM unit, which takes into account the current input ($x^{(t)}$), the output from the previous LSTM unit $y^{(t-1)}$, and

the cell value from the previous time step $c^{(t-1)}$. This update operation is carried out using the following equation

$$i(t) = \sigma\left(W_i * x^{(t)} + R_i * y^{(t-1)} + p_i \odot c^{(t-1)} + b_i\right) \quad \text{(III.46)}$$

The input gate controls the addition of new information to the cell state.

Here, σ represents the sigmoid activation function, while Wi, Ri, and pi correspond to the weights associated with $x^{(t)}$, $y^{(t-1)}$, and $y^{(t-1)}$respectively. bi signifies the bias vector linked to this specific component.

Forget Gate ($f^{(t)}$)

In this phase, the LSTM unit determines what information to forget from its prior cell states $c^{(t-1)}$. The calculation of activation values $f^{(t)}$ for the forget gates at time step t is based on the current input $x^{(t)}$, the previous outputs $y^{(t-1)}$, the state of memory cells from the previous time step $c^{(t-1)}$, along with the peephole connections, and the bias terms bf associated with the forget gates. This computation can be expressed as

$$f^{(t)} = \sigma\left(W_f * x^{(t)} + R_f * y^{(t-1)} + p_f * y^{(t-1)} + p_f \odot c^{(t-1)} + b_f\right) \quad \text{(III.47)}$$

$$f^{(t)} = \sigma\left(W_f * x^{(t)} + R_f * y^{(t-1)} + p_f * y^{(t-1)} + p_f \odot c^{(t-1)} + b_f\right) \quad \text{(III.48)}$$

Here, Wf, Rf, and pf represent the weights corresponding to $x^{(t)}$, $y^{(t-1)}$, and $c^{(t-1)}$ respectively, while bf denotes the bias weight vector [137].

Cell State Update (c(t))

In this phase, the LSTM unit computes the cell value by combining the block input $\boldsymbol{z^{(t)}}$, the input gate $\boldsymbol{i^{(t)}}$and the forget gate $\boldsymbol{f^{(t)}}$ values with the previous cell value [137]. This operation is represented as follows

Top of Form

$$\boldsymbol{c(t)} = \boldsymbol{z^{(t)}} \odot i^{(t)} + \boldsymbol{c^{(t-1)}} \odot \boldsymbol{f^{(t)}} \quad \text{(III.49)}$$

The cell state is updated based on the input gate and forget gate.

Output Gate (o(t))

In this stage, the LSTM computes the output gate, which is a function of the current input $x^{(t)}$, the previous LSTM unit output $y^{(t-1)}$, and the cell value $c^{(t-1)}$ from the previous iteration [137]. The output gate calculation is expressed as

$$O^{(t)} = \sigma\left(W_0 * x^{(t)} + R_0 * y^{(t-1)} + p_0 \odot\ c^{(t-1)} + b_0\right) \quad \text{(III.50)}$$

Where Wo, Ro, and po represent the weights associated with $x^{(t)}, y^{(t-1)}$and $c^{(t-1)}$ respectively, and bo is the bias weight vector [137].

Output (y(t)) The final output is generated by applying an activation function g(·) to the cell state and scaling it by the output gate.

$$\boldsymbol{y^{(t)}} = \boldsymbol{g}\left(\boldsymbol{c^{(t)}}\right) \odot \boldsymbol{o^{(t)}} \quad \text{(III.51)}$$

The logistic sigmoid function, denoted as σ(x), is used as a gate activation function and is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$.

The hyperbolic tangent function, represented as g(x), and h(x), is frequently employed as the activation function for block input and output, and it is defined as tanh(x) [137].

Backpropagation in LSTM

Backpropagation is the process of calculating gradients (δ) of the loss function with respect to various components within the LSTM cell. These gradients guide the updates of the model's parameters during training. During the backward pass, the cell state c(t) accumulates gradients from both y(t) and the subsequent cell state $c^{(t-1)}$. These gradients are aggregated before being propagated back to the current layer.

In the final iteration T, the change $\boldsymbol{\delta_y^{(t)}}$represents the network error gradient $\partial E/\boldsymbol{\delta_y^{(t)}}$ with E denoting the loss function. For other iterations $\boldsymbol{\delta_y^{(t)}}$is the vector of delta values inherited from the layer above, including the recurrent dependencies. This process is expressed as follows

Gradient Descent and Parameter Updates

To train the LSTM, we use gradient-based optimization algorithms such as stochastic gradient descent (SGD), Adam, or RMSprop. These algorithms update the LSTM cell's parameters (weights and biases) using the gradients calculated during backpropagation [137].

For example, the update equations for weights and biases can be

$$\Delta W_* = -\alpha \sum_{t=0}^{T} \delta_*^{(t)} \otimes x^{(t)} \tag{III.52}$$

$$\Delta pi = -\alpha \sum_{t=0}^{T-1} c^{(t)} \odot \delta_i^{(t+1)} \tag{III.53}$$

$$\Delta R *= -\alpha \sum_{t=0}^{T-1} \delta_*^{(t+1)} \otimes y^{(t)} \tag{III.54}$$

$$\Delta pf = -\alpha \sum_{t=0}^{T-1} c^{(t)} \odot \delta_f^{(t+1)} \tag{III.55}$$

$$\Delta b * = -\alpha \sum_{t=0}^{T} \delta_*^{(t)} \tag{III.56}$$

$$\Delta po = -\alpha \sum_{t=0}^{T} c^{(t)} \odot \delta_o^{(t)} \tag{III.57}$$

In this context, the symbol $\otimes$ signifies the outer product of two vectors, while the symbol $*$ can represent any component related to the weights, such as the block input $\hat{z}$, the input gate $\hat{\imath}$, the forget gate $\hat{f}$, or the output gate $\hat{o}$.These equations form the foundation of LSTM RNNs, allowing them to effectively model sequential data, capture long-term dependencies, and adapt their parameters during training for various tasks, such as natural language processing, time series forecasting, and more [137].

3.7.4. LSTM MATLAB IMPLEMENTATOIN

Introduction This research explores the use of ensemble LSTM (Long Short-Term Memory) neural networks to enhance fault detection accuracy in signal classification tasks. Accurate fault detection is crucial in domains like industry and healthcare. This study focuses on improving classification performance through ensemble learning with LSTM networks.

```
numHiddenUnits = 1500;
numClasses = numel(unique(YTrain));
layers = [sequenceInputLayer(inputSize, 'Normalization', 'zscore')
          lstmLayer(numHiddenUnits, 'OutputMode', 'last')
          fullyConnectedLayer(numClasses)
          softmaxLayer
          classificationLayer];
```

Ensemble Learning: Trains multiple neural network models independently (specified as numModels) to introduce diversity among models.

```
numModels = 3;
YPredAll = zeros(length(YTest), numModels);
YTrain = categorical(YTrain);
for i = 1:numModels
    % Train a new model
    net = trainNetwork(TrainFeatures, YTrain, layers, options);

    % Predict on the test set
    YPred = classify(net, TestFeatures);

    % Store predictions
    YPredAll(:, i) = grp2idx(YPred);
end
```

Network Architecture Utilizes a neural network architecture with sequence input, LSTM, fully connected, softmax, and classification layers to capture temporal dependencies in signal data.

Prediction Aggregation Aggregates predictions from individual models using a mode operation to improve generalization and reduce overfitting.

```
% Create an ensemble prediction
YPredEnsemble = mode(YPredAll, 2);
YPredEnsemble = categorical(YPredEnsemble, 1:8,
catnames);
```

Results Reports accuracy of ensemble predictions compared to true labels (YTest) and visualizes classification performance using a confusion matrix.

```
% Calculate accuracy
accuracy = 100 * sum(YPredEnsemble == YTest') / numel(YTest);
% Display confusion matrix
figure;
cm = confusionchart(YTest, YPredEnsemble);
title('Classification Performance');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';

% Print accuracy and confusion matrix
fprintf('Accuracy: %.2f%%\n', accuracy);
```

the effectiveness of ensemble LSTM neural networks in fault detection for signal classification, highlighting the potential for enhanced accuracy and robustness. This research contributes to improving fault detection methodologies with applications in critical domains.

SUPPORT VECTOR MACHINES (SVMS)	NEURAL NETWORK
• Model Complexity SVMs are generally simpler models compared to neural networks. They aim to find a hyperplane that best separates classes. • Feature Engineering SVMs require careful feature engineering, especially for a moderate number of features. • Scalability SVMs can be computationally expensive, particularly with large datasets [138]. • Model Interpretability SVMs are often considered more interpretable due to their decision boundary being defined by support vectors [139]. • Performance SVMs perform well in tasks with moderate data availability and when data exhibits linear separability or can be effectively separated with a suitable kernel [140].	• Model Complexity NNs can be extremely complex, with the ability to learn intricate, non-linear relationships within the data. • Feature Engineering NNs can automatically learn features from raw data, making them suitable for high-dimensional or unstructured data. • Scalability NNs are highly scalable and parallelizable, making them suitable for large datasets and distributed computing. • Model Interpretability NNs, particularly deep architectures, are often considered black-box models, making it challenging to interpret their predictions [138]. • Performance NNs, especially deep learning models, have shown state-of-the-art performance in various domains, but they often require large amounts of labeled data [140].

3.7.5. SVM Vs Neural Network

In summary, there's no definitive answer regarding which is better between SVMs and NNs. The choice should be guided by the specific characteristics of your data, your computational resources, and your performance and interpretability requirements. It's often valuable to experiment with both approaches to determine which one best suit our particular task. Additionally, hybrid models that combine SVMs and NNs can be explored in some scenarios to leverage the strengths of both techniques.

3.7.6. Optimization of the Long Short-Term Memory (LSTM) Neural Network

In this section, we explore the optimization strategies applied to our Long Short-Term Memory (LSTM) neural network, a pivotal component of our study. Our primary aim was to fine-tune the LSTM's parameters to enhance its performance and expedite convergence. The cornerstone of this optimization endeavor was the utilization of the Adam optimization algorithm, a potent tool for training neural networks.

3.7.6.1. Adam Optimization Algorithm

The Adam optimizer, stemming from the term "Adaptive Moment Estimation," stands as a leading choice for optimizing neural networks, including LSTMs. It amalgamates the strengths of two renowned optimization techniques, Adagrad and RMSprop. What sets Adam apart is its capability to dynamically adjust learning rates for individual parameters, rendering the training process more efficient and effective. This algorithm maintains two moving averages for each parameter the first moment (mean) and the second moment (uncentered variance). These moving averages adapt the learning rate throughout training, ensuring optimal convergence [141].

In MATLAB, we implemented the Adam optimization algorithm, making use of built-in functions within deep learning frameworks or custom implementations tailored to our study's specific requirements. The optimization process was finely tuned through careful adjustment of hyperparameters, including the learning rate (α), decay rates (β1 and β2), and epsilon (ε) for numerical stability, to achieve not only optimal convergence but also superior performance tailored to our specific task [142].

3.7.6.2. Hyperparameter Tuning

The effectiveness of an LSTM network hinges on the judicious selection of hyperparameters. To discern the optimal configuration for our LSTM, we embarked on an exhaustive hyperparameter tuning process, covering several pivotal parameters

Learning Rate (α) This parameter governs the step size during weight updates within the optimization process. Extensive experimentation was conducted to pinpoint the learning rate that facilitated swift convergence without encountering challenges such as overshooting or convergence to local minima.

Batch Size The choice of mini-batch size during training significantly influences convergence speed and memory usage. Systematic exploration of various batch sizes was undertaken to strike a balance between rapid convergence and manageable memory requirements [143].

Architecture The architectural design of an LSTM network is crucial for its ability to model complex patterns [143]. In our case, we employed the following architecture:

```
    layers = [ ...
        sequenceInputLayer(inputSize,'Normalization','zscore')
        lstmLayer(numHiddenUnits,'OutputMode','last')
        fullyConnectedLayer(numClasses)
        softmaxLayer
        classificationLayer];
giving the options as below:
maxEpochs = 170;
miniBatchSize = 170;

options = trainingOptions('adam', ...
'InitialLearnRate', 1e-4, ...
'MaxEpochs', 170, ...
'MiniBatchSize', 170, ...
'SequenceLength', 'longest', ...
'Shuffle', 'every-epoch', ...
'ValidationData', {TestFeatures, YTest}, ...
'ValidationFrequency', 30, ...
'Verbose', 1, ...
'Plots', 'training-progress', ...
'ExecutionEnvironment', 'gpu');
```

Where the Number of Hidden Unites = 1500;

Max Epochs = 170;

Minimum Batch Size = 170;

and in the other Model we used the below parameters:

```
% Set the training parameters
net.trainParam.epochs = 200; % Set the number of epochs to 100
net.trainParam.max_fail = 20; % Set the maximum number of
validation failures to 20
net.trainParam.lr = 0.01; % Set the learning rate to 0.01
net.trainParam.mc = 0.9; % Set the momentum constant to 0.9
net.trainParam.beta1 = 0.9; % Set the beta1 parameter for Adam
optimization to 0.9
net.trainParam.beta2 = 0.999; % Set the beta2 parameter for Adam
optimization to 0.999
net.trainParam.epsilon = 1e-8; % Set the epsilon parameter for Adam
optimization to 1e-8
```

Training and Ensemble Learning We used an ensemble learning approach by training multiple LSTM models and taking a majority vote for predictions [143]. The code for this ensemble learning is as follows:

```
numModels = 3; % number of models to train
YPredAll = zeros(length(YTest), numModels); % initialize matrix to
store predictions
YTrain=categorical(YTrain);
for i = 1:numModels
    % train a new model
    net = trainNetwork(TrainFeatures,YTrain,layers,options);

    % predict on test set
    YPred = classify(net,TestFeatures);
    YPred =
renamecats(YPred,{'1','2','3','4','5','6','7','8'},catnames);

    % store predictions in YPredAll matrix
    YPredAll(:, i) = grp2idx(YPred);
end

% take the mode prediction across all models
YPredEnsemble = mode(YPredAll, 2);
YPredEnsemble = categorical(YPredEnsemble, 1:8, catnames);
YTest = renamecats(YTest,{'1','2','3','4','5','6','7','8'},catnames);
```

3.7.6.3. Evaluation and Confusion Matrix

To assess the model's performance, we computed accuracy and displayed a confusion matrix:

```
% compute accuracy
accuracy = 100*sum(YPredEnsemble == YTest') /
numel(YTest)

% Plot the confusion matrix for the final model
figure;
cm = confusionchart(YTest, YPredEnsemble);
title('Classification for a faulty and healthy signal for
each road type');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
```

The hyperparameter tuning process encompassed diverse methodologies, ranging from grid search to random search and Bayesian optimization, the selection contingent upon the intricacy of the hyperparameter space. Robust evaluation metrics, encompassing but not limited to accuracy, loss, and domain-specific criteria, were enlisted to gauge the LSTM network's performance [143].

The harmonious amalgamation of the Adam optimization algorithm, ensemble learning, and meticulous hyperparameter tuning culminated in a finely-tuned LSTM architecture and training regimen, delivering optimal outcomes meticulously tailored to the specifics of our task.

3.7.7. SVM Tuning

Tuning SVM parameters is essential to achieve the best possible performance for your specific problem. In MATLAB, we can tune SVM hyperparameters using various techniques, including grid search and cross-validation. Here's a step-by-step on how we tuned SVM in MATLAB

3.7.7.1. Splitting Data for Cross-Validation

To assess the effectiveness of different hyperparameters, it's essential to divide the dataset into training and validation sets. MATLAB offers functions like 'cvpartition' or 'crossval' to create cross-validation folds [144].

3.7.7.2. Choosing the SVM Kernel

SVMs offer various kernels, including linear, polynomial, radial basis function (RBF), or sigmoid. The kernel choice plays a vital role in model's performance, so we should carefully select the most suitable one for your specific problem.

3.7.7.3. Selecting Hyperparameters

Critical SVM hyperparameters to fine-tune include

C This parameter controls the balance between maximizing the margin and minimizing classification errors [145]. In our model, we set the 'BoxConstraint' hyperparameter to the values '0.01,' '0.1,' '1,' and '10.

3.7.7.4. Kernel Parameters

Depending on the selected kernel type, such as the polynomial degree for polynomial kernels or kernel width for RBF kernels, additional parameters may need adjustment [145]. In our model, we specifically adjusted the 'KernelScale' hyperparameter, using the values '0.01,' '0.1,' '1,' and '10.

Grid Search In MATLAB, the 'fitcsvm' function is employed to train SVM classifiers. For multi-class problems, 'fitcecoc' can be utilized. To execute a grid search for hyperparameter optimization, leverage 'fitcecoc' along with the 'HyperparameterOptimizationOptions' parameter [145]. Below is an example:

```
% Create an SVM template
template = templateSVM('KernelFunction', 'rbf');
% Define the hyperparameter search space
parameters = struct('BoxConstraint', [0.01, 0.1, 1, 10], 'KernelScale',
[0.01, 0.1, 1, 10]);
% Set up hyperparameter optimization options
options = struct('Optimizer', 'bayesopt', 'MaxObjectiveEvaluations',
30);
% Conduct hyperparameter optimization using fitcecoc
svm_model = fitcecoc(XTrain, YTrain, 'Learners', template,
'OptimizeHyperparameters', 'auto',...
    'HyperparameterOptimizationOptions', options, 'HyperparameterRange',
parameters);
% Assess the best model's performance on the validation set
predictions = predict(svm_model, XValidation);
```

3.7.7.5. Model Evaluation

Following hyperparameter tuning, evaluate the SVM model's performance using appropriate metrics, such as accuracy, precision, recall, or F1-score, on a test dataset or through cross-validation.

Always keep in mind that the choice of kernel and hyperparameters significantly influences our SVM model's performance. It's crucial to experiment with various configurations while employing cross-validation to prevent overfitting.

IV. THE RESULTS AND DISCUSSION

In this chapter, we fuse the outcomes and insights from our experiments in road obstacle detection and suspension fault classification, providing a comprehensive view of both results and their implications. Our research, powered by an Intel(R) Core(TM) i7-7820HK CPU and NVIDIA GeForce GTX 1080 GPU, spanned several tests, each targeting specific aspects of detection and classification under various environmental conditions.

We focused on the Road Type and Signal Condition Classification, incorporating real-world variables such as noise and denoising processes. This approach allowed us to assess the robustness of our systems under different operational scenarios.

Our discussion goes beyond presenting data, and delving into the performance metrics, challenges, and potential for enhancements in practical applications. We critically analyze the impact of noise on classification accuracy and evaluate the effectiveness of various denoising techniques.

Furthermore, this chapter acknowledges the study's limitations, offering insights into the constraints and practical considerations that shaped our research. We also put forward recommendations for future research, aiming to advance suspension fault detection systems.

Summarizing, this chapter bridges our experimental results with thoughtful analysis, setting the stage for future innovations in automotive safety and performance enhancement.

4.1. First Test Discussion Road obstacles 5 levels of training and testing (With Noise)

In our initial test for detecting road obstacle faults, we examined 15 diverse methods to identify obstacles such as Sinewaves, Grids, Potholes, and Bumps across five different levels of faults. To simulate realistic driving conditions, we introduced various types of noise, such as those resembling grain or engine sounds, which can affect the tie rod during driving. The results in a noisy environment (as shown in Table 1) were enlightening. Notably, the Wavelet Scattering SVM PCA method demonstrated exceptional performance, achieving a mean accuracy of 99.1%.

Table 1 First Test Method Performance Summary (With Noises)

Model Names	Number of Training Cycles	Mean Accuracy (%)	Std Accuracy (%)	Mean Time (sec)	Std Time (sec)
Wavelet Scattering SVM PCA	5 Folds	99,1	0,144	11	1
DWT SVM	1 Iteration	95	2,16	9	4
Wavelet Scattering SVM Tuned	20 Evaluations	95	1,41	403	62
Wavelet Scattering SVM CV SRE	5 Folds	94,9	2,66	11	1
Wavelet Scattering RNN ADAM	170 Epochs	94,8	1,5	53	3
Wavelet Scattering SVM Tuned SRE	20 Evaluations	94,5	1,29	183	75
DWT SVM Tuned	20 Evaluations	94,3	1,26	382	346
Wavelet Scattering SVM CV	5 Folds	94,3	1,55	22	20
Wavelet Scattering RNN	170 Epochs	93,8	2,63	59	4
Wavelet Scattering SVM Tuned RFE	20 Evaluations	93	1,41	123	14
Wavelet Scattering SVM CV RFE	5 Folds	91,1	0,625	13	1
DWT SVM SRE Tuned	20 Evaluations	91	2,94	487	403
Wavelet Scattering SVM CV LDA	5 Folds	77	5,72	9	1
Wavelet Scattering SVM Tuned LDA	20 Evaluations	68	7,16	83	2
DWT Neural Network	200 Epochs	54	6	10	9

4.1. Second Test Discussion Road obstacles 5 levels of training and testing (Denoised)

In the second test, we focused on noise removal from road obstacles, using MATLAB's 'wdenoise' function to eliminate the noises we introduced in the first test. Subsequently, we applied the same 15 methods to detect Sinewave, Grid, Pothole, and Bump obstacles across five varied levels of faults. The results showed that not only did the top-performing method from the first test (referenced in Table 2) maintain its high accuracy post-denoising, but it also confirmed the robustness of this model. In contrast, while methods like the Wavelet Scattering SVM Tuned RFE and Tuned exhibited significant improvements after denoising, they also demonstrated a notable increase in processing time. This highlights the essential balance between detection accuracy and operational efficiency in road safety applications.

Table 2 Second Test Method Performance Summary (Denoised)

Model Names	Number of Training Cycles	Mean Accuracy (%)	Std Accuracy (%)	Mean Time (sec)	Std Time (sec)
Wavelet Scattering SVM PCA	5 Folds	97,9	2,75	11	7
Wavelet Scattering SVM Tuned RFE	20 Evaluations	97,6	1,11	91	16
Wavelet Scattering SVM Tuned	20 Evaluations	97,3	0,655	565	368
Wavelet Scattering SVM Tuned SRE	20 Evaluations	96	0,816	146	25
Wavelet Scattering RNN ADAM	170 Epochs	96	2	53	1
Wavelet Scattering SVM CV RFE	5 Folds	94,8	0,315	13	1
Wavelet Scattering RNN	170 Epochs	94,5	0,577	54	1
DWT SVM	1 Iteration	94,5	1,29	5	2
DWT SVM Tuned	20 Evaluations	93,5	3,7	361	189
Wavelet Scattering SVM CV SRE	5 Folds	93,4	0,903	12	5
Wavelet Scattering SVM CV	5 Folds	93,3	0,772	18	8
DWT SVM SRE Tuned	20 Evaluations	90,5	2,08	561	444
Wavelet Scattering SVM CV LDA	5 Folds	84,9	10,2	8,5	0,5
DWT Neural Network	200 Epochs	82	3,46	247	41
Wavelet Scattering SVM Tuned LDA	20 Evaluations	73,3	10,6	84	2

4.2. Third Test Road Type and Signal Condition Classification Comprehensive Testing (With Noise)

The objective of the third test was to classify road types and signal conditions as either healthy or faulty, with a dual emphasis on maximizing accuracy and minimizing processing time. This comprehensive evaluation included scenarios with noise interference. The results, detailed in Table 3 for conditions with noise, reveal that the Wavelet Scattering RNN ADAM model exhibited high performance. The performance details of the tuning in ADAM's optimization can be found in Figure IV-1. The model maintained a mean accuracy of 97.8% with a relatively low standard deviation, indicating not only its effectiveness but also its consistency under noisy conditions.

Table 3 Third Test Summary of Method Performance Metric (With Noise)

Model Names	Number of Training Cycles	Mean Accuracy (%)	Std Accuracy (%)	Mean Time (sec)	Std Time (sec)
Wavelet Scattering RNN ADAM	170 Epochs	97,8	0,776	360	79
DWT Neural Network	200 Epochs	97,7	0,261	41	6
Wavelet Scattering SVM Tuned SRE	20 Evaluations	97,5	0,304	2970	530
Wavelet Scattering SVM CV	5 Folds	97,2	0,846	86	17
Wavelet Scattering SVM CV SRE	5 Folds	97,1	0,463	91	12
Wavelet Scattering SVM CV RFE	5 Folds	96,5	0,0866	73	13
Wavelet Scattering SVM Tuned RFE	20 Evaluations	96,3	0,135	2010	406
Wavelet Scattering SVM PCA	5 Folds	95,8	0,61	81	14
Wavelet Scattering SVM Tuned LDA	20 Evaluations	95,6	0,367	224	42
Wavelet Scattering RNN	170 Epochs	95,5	0,685	386	56
Wavelet Scattering SVM CV LDA	5 Folds	95,4	0,2	62	12
Wavelet Scattering SVM Tuned	20 Evaluations	95,3	0,735	2630	483
DWT SVM Tuned	20 Evaluations	95	0,598	4590	486
DWT SVM SRE Tuned	20 Evaluations	93,1	0,304	4420	836
DWT SVM	1 Iteration	92,2	0,358	194	38

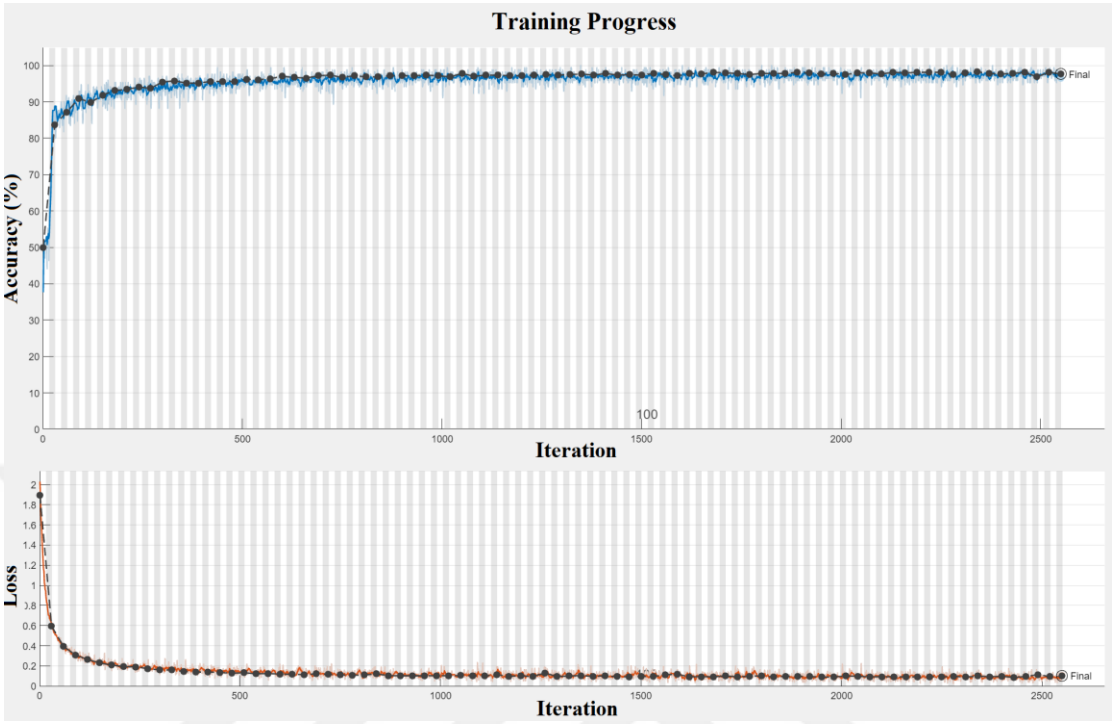

Figure IV-1 Training Progress Road Type and Signal Condition Classification (With Noise) with Wavelet Scattering Neural Network Test Using adam Optimizer

Figure IV-1 depicts the final training progress of the model, concluding at iteration 2550 within three model loops with an impressive accuracy of 97.50%. Notably, this accuracy exhibits significant growth within the first 200 iterations of the training process, followed by a gradual increase until it reaches the final accuracy of 97.50%.

4.3. Fourth Test Road Type and Signal Condition Classification Comprehensive Testing (Denoised)

In the fourth test, we employed the denoising function, as mentioned in the previous test, to assess road type and signal condition classification under denoised conditions. The results, presented in Table 4, showed that the Wavelet Scattering SVM Tuned LDA model was a standout, achieving a mean accuracy of 98.4%. Details of the model's

tuning and evaluation can be found in Figures IV-2 and IV-3. However, it's important to highlight that this high accuracy came at the cost of increased processing time, illustrating the trade-off between accuracy and efficiency. These findings are vital for the advancement of road condition classification systems. They emphasize the significance of choosing the right model for each specific scenario to balance road safety and operational efficiency effectively.

Table 4 Fourth Test Summary of Method Performance Metric (Denoised)

Model Names	Number of Training Cycles	Mean Accuracy (%)	Std Accuracy (%)	Mean Time (sec)	Std Time (sec)
Wavelet Scattering SVM Tuned LDA	20 Evaluations	98,4	0,348	219	46
Wavelet Scattering SVM Tuned	20 Evaluations	98,4	0,481	3580	602
Wavelet Scattering SVM Tuned SRE	20 Evaluations	98,3	0,18	3400	691
Wavelet Scattering SVM Tuned RFE	20 Evaluations	98,3	0,289	3760	569
Wavelet Scattering RNN ADAM	170 Epochs	98,3	0,382	363	18
DWT Neural Network	200 Epochs	98,1	0,499	44	3
Wavelet Scattering SVM CV SRE	5 Folds	97,8	0,45	123	19
Wavelet Scattering SVM CV LDA	5 Folds	97,7	0,456	60	12
Wavelet Scattering RNN	170 Epochs	97,7	0,576	355	14
Wavelet Scattering SVM CV RFE	5 Folds	97,4	0,17	71	11
Wavelet Scattering SVM CV	5 Folds	97,1	0,551	114	22
DWT SVM Tuned	20 Evaluations	97	0,439	4420	861
DWT SVM SRE Tuned	20 Evaluations	95,9	0,421	1560	269
Wavelet Scattering SVM PCA	5 Folds	92,9	0,52	551	101
DWT SVM	1 Iteration	92,5	0,41	190	28

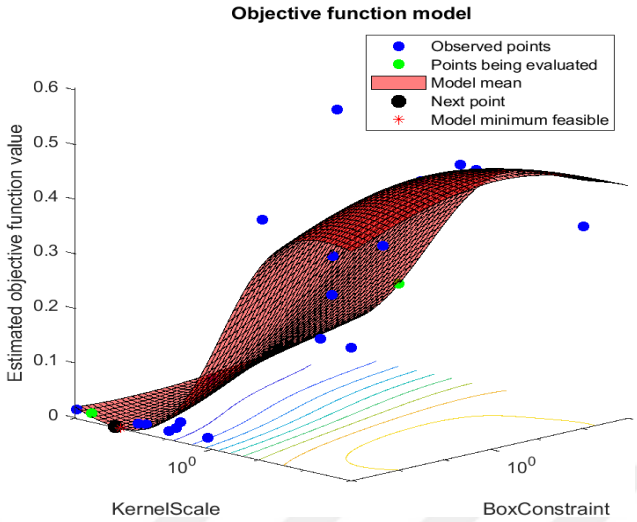

Figure IV-2 Objective Function Modeling of a faulty and healthy signal for each road type (Denoised) Wavelet Scattering and SVM Tuned and Optimized with LDA

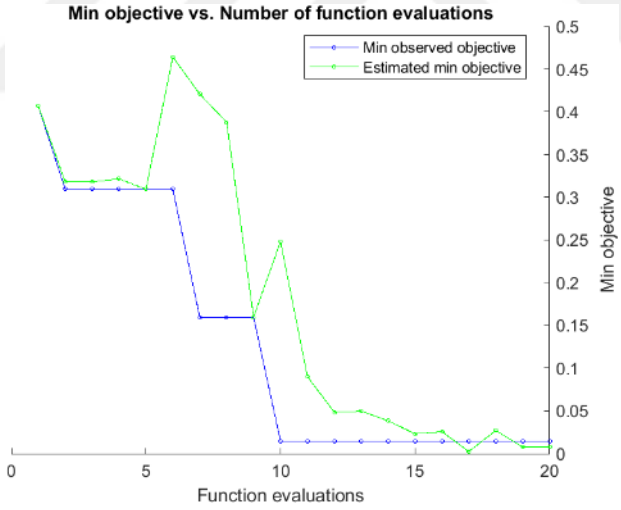

Figure IV-3 Minimum Objective vs. Number of Function Evaluations of a faulty and healthy signal for each road type (Denoised) Wavelet Scattering and SVM Tuned and Optimized with LDA

In Figure IV-2, when we analyze the expected and actual performance of the tuned SVM model, we notice that 'Observed Points' cluster within a range of 0 to 0.6 in the

minimum objective function value. This pattern suggests that the model's performance is within an acceptable range. However, there are occasional instances where the performance falls outside this range. This variability in performance can be attributed to factors such as data characteristics, inherent limitations of the model, or variations in the datasets used for evaluation.

In Figure IV-3, we observe that the hyperparameter optimization process produced promising results. The 'Minimum Observed Objective' achieved a low value of 0.02, signifying strong and robust model performance according to our selected metric. While the 'Estimated Minimum Objective' showed some fluctuations during the optimization process, it converged with the minimum observed objective. Although it did not reach the absolute lowest objective value, this outcome still indicates a level of performance that is both acceptable and effective for our specific task.

Result Conclusion

The extensive evaluation of various methods for detecting and classifying road obstacles under different conditions, including noise and denoising, offers valuable insights for autonomous driving and road safety. The tests demonstrate notable performance variations among methods, underscoring the necessity of choosing the most suitable technique based on application-specific requirements, data characteristics, and the trade-off between accuracy and processing time.

In the first test, which focused on classifying road obstacles in noise-free conditions, the Wavelet Scattering SVM PCA method emerged as the most effective, achieving an exceptional mean accuracy of 99.1%. This result highlights the significance of

sophisticated algorithmic approaches in achieving high accuracy in road obstacle detection.

In the second test, which included scenarios with and without noise, the Wavelet Scattering SVM PCA method maintained its robustness, achieving a high accuracy rate of 97.9% post-denoising. This performance parallels its effectiveness in vibration fault diagnosis for bearings at variable speeds, as noted in reference [146]. However, other methods like the Wavelet Scattering SVM Tuned RFE and the Wavelet Scattering SVM Tuned also showed significant improvements after denoising. These improvements, though, came at the cost of increased processing time. Nonetheless, they demonstrated robustness, especially in detecting unbalanced and bowed rotors, as documented in reference [147]. Specifically, the Wavelet Scattering SVM Tuned RFE achieved an accuracy of 97.6%, while the Wavelet Scattering SVM Tuned reached 97.3%. This test highlighted the critical balance between detection accuracy and operational efficiency in road safety applications.

In the third test, which introduced noise to the road obstacle detection process, the Wavelet Scattering RNN ADAM model stood out. It maintained a mean accuracy of 97.8% with low variability, demonstrating high performance and consistency under noisy conditions. Furthermore, the integration of wavelet scattering with LSTM proved effective in classifying unbalanced and bowed rotors, as indicated in reference [147].

Finally, in the fourth test, incorporating both noise and denoising, the Wavelet Scattering SVM Tuned LDA model was a top performer with a mean accuracy of 98.4%, albeit with increased processing time. This highlights the crucial balance

between accuracy and operational efficiency. It has also showed promise in mixed signal environments, akin to its application in bearing vibration fault detection [148].

In conclusion, the optimal method for road obstacle detection and classification varies depending on data nature, noise presence, and the desired accuracy-efficiency balance. The effectiveness of feature selection, preprocessing, and adaptive techniques in enhancing performance and robustness is evident. These findings provide essential guidance for developing advanced road obstacle detection systems, contributing significantly to the field of autonomous driving technology.

4.4. Limitations of the Study

This research endeavor encountered certain limitations stemming from practical constraints and technical considerations. These limitations prompted the utilization of alternative methodologies to mitigate the challenges posed by the available resources. The scope and constraints of this study are delineated as follows

4.4.1. Data Collection

In this study, the primary method for data acquisition involved the utilization of Adams car simulation software. While Adams car simulation software is well-established for its data generation reliability, it is imperative to underscore that the specific software version employed in this research lacked the intrinsic capacity to directly simulate faults. To address this intrinsic limitation, the simulation methodology introduced variations in angular acceleration predicated on the influence of wear-induced changes. Moreover, in lieu of actual real-world noise, virtual noise components were incorporated. These methodological adaptations were implemented with the explicit objective of facilitating the study's ability to effectively capture and analyze the

manifestations of faults. This strategic approach was necessitated by the resource constraints encountered in this study and the concomitant fiscal implications associated with procuring dedicated sensors and control units for direct fault diagnosis.

In the context of the application of accelerometer data for suspension fault detection, the data collection process necessitates meticulous attention to the following key considerations:

Dataset Diversity and Extensiveness: The dataset employed in this research was diligently constructed to encompass a wide spectrum of real-world road conditions. This comprehensive dataset design aimed to faithfully model suspension behavior under various environmental and road condition scenarios.

Dynamic Road Condition Variations: The data collection strategy incorporated the need to account for dynamic fluctuations in road conditions. This included accommodating transient anomalies such as potholes and other road surface irregularities, which necessitated the collection of data under diverse driving scenarios.

Generalizability of the Model: Ensuring the robust generalizability of the machine learning model across distinct road conditions was a paramount consideration. The model's capacity to remain effective across different environmental contexts and its ability to maintain resilience against undue sensitivity to minor variations were critical for its practical applicability in real-world scenarios.

Vital Information Captured by Accelerometer Data It is crucial to underscore the significance of accelerometer data, which captures pivotal vehicle movements and vibrations. These captured dynamics play an indispensable role in the detection of suspension issues that might not be readily discernible solely from surface

irregularities. Recognizing this pivotal role is imperative for accurate fault diagnosis and classification.

4.4.2. Hardware Performance Evaluation

Another substantial limitation of this study was the unavailability of an array of GPUs, CPUs, or FPGAs with varying processing speeds for the purpose of evaluating their influence on fault detection algorithms. Ideally, a comprehensive assessment would have involved testing diverse hardware speeds to gauge their impact on the effectiveness of the fault detection algorithms. Regrettably, due to resource constraints, this aspect could not be explored in depth. Nonetheless, this study focused on evaluating the overall performance and efficacy of the fault detection algorithms, irrespective of specific hardware variations.

4.4.3. Time Constraints

Time constraints presented a formidable challenge in this research endeavor. These temporal limitations-imposed constraints on the development of a more extensive array of fault detection algorithms and the exploration of a broader spectrum of road types. Although the study aspired to generate a substantial number of fault detection codes and incorporate a wide range of road scenarios, time restrictions curtailed the extent of these efforts. However, within the confines of the available time frame, the study diligently aimed to provide meaningful analysis and insights.

In conclusion, this study acknowledges these limitations and has endeavored to mitigate their impact to the greatest extent possible, given the resources and time available, in pursuit of meaningful contributions to the field of suspension fault detection.

4.5. Recommendations for future research and practical applications

In light of the limitations identified in this study, several compelling avenues for future research and practical applications emerge, each offering unique prospects for enhancing suspension fault detection methodologies.

4.5.1. Integration of LiDAR with Accelerometer Data

Exploration of LiDAR technology integration with accelerometer data is a paramount recommendation for suspension fault detection. This innovative technique utilizes LiDAR's ability to capture large amounts of training data and combines it with the practical testing capabilities of accelerometer data. Previous studies have demonstrated the effectiveness of LiDAR in earthquake detection [149], and this synergy can significantly enhance the accuracy and reliability of suspension fault detection models, particularly in real-world driving scenarios.

4.5.2. Incorporation of Real-Life Data

A key recommendation is to emphasize the incorporation of authentic real-world data in research efforts. While software simulations provide controlled environments, the resource-intensive collection of real-life data can provide invaluable insights into suspension behavior under diverse road conditions. Bridging the gap between laboratory findings and actual road performance augments the practical utility of suspension fault detection systems significantly.

4.5.3. Merging Finite Element Analysis with Vibration Fault Detection

The integration of finite element analysis with vibration fault detection methodologies is a promising approach that can significantly enhance suspension fault detection

accuracy and predictive capabilities, especially when considering the fatigue life cycle. This innovative integration enables the classification of fault severity based on finite element grades. For instance, in the case of tie rod degradation, specific vibrations corresponding to varying degrees of wear and high vibrations can signify different stages of the fatigue life cycle and levels of potential danger. By aligning the severity of detected faults through vibration analysis with fatigue life cycle-linked finite element grades, as done in Vibration Analysis of Defected Ball Bearings [150], this approach facilitates the provision of more precise warnings to drivers.

4.5.4. Investment in Advanced Hardware Resources

Crucial to the advancement of research endeavors is investing in advanced hardware resources, such as high-performance GPUs and processors. These resources facilitate the comprehensive analysis of complex real-life data, enabling the application of sophisticated machine-learning techniques to achieve heightened fault detection accuracy.

4.5.5. Conducting Comparative Studies on Machine-Learning Algorithms

Conducting comparative studies to evaluate different machine-learning algorithms in suspension fault detection is highly advantageous. Systematically comparing classifiers and feature extraction techniques aids in pinpointing the most reliable and efficient approach for practical implementation.

4.5.6. The practical applications of these recommendations span across various sectors

Automotive Industry Integration of suspension fault detection systems in vehicle manufacturing can enhance safety and performance, enabling early identification of suspension issues, reducing maintenance costs, and ensuring safer driving experiences for consumers.

Fleet Management Companies Integration of suspension fault detection into maintenance routines can lead to increased operational efficiency, reduced downtime, and substantial cost savings through timely identification of suspension problems.

Road Safety Authorities Leveraging suspension fault detection systems for continuous monitoring of suspension health can swiftly identify vehicles with potential issues, allowing authorities to take proactive measures to prevent accidents arising from vehicle malfunctions.

Aftermarket Automotive Service Providers Offering suspension fault detection as a value-added service can assist car owners in more effectively maintaining their vehicles, potentially extending their lifespan.

Embracing these recommendations not only propels the field of suspension fault detection forward but also ensures tangible implementation across diverse sectors, promising a safer and more efficient automotive landscape.

4.5.7. The Cost and Application of Accelerometer in Vehicular Vibration Measurement

To effectively measure vibrations in the outer tie rod of a car suspension, an accelerometer with a frequency range of 10 Hz to 1000 Hz and a sensitivity of at least 100 mV/g is required.

The PCB Piezotronics Model 601A01 accelerometer aligns perfectly with these criteria. With a frequency range spanning from 0.27 Hz to 10 kHz and a sensitivity of 100 mV/g, it is specifically designed for high-frequency vibration measurement in automotive and aerospace applications. Priced at $220 per unit, this accelerometer provides a cost-effective solution for precise measurements [1]. To harness the full potential of data processing, customization of a chassis control module or suspension control module is recommended. This tailored modification ensures seamless integration with other functions, thereby enhancing safety and reliability [151]. Additionally, establishing connectivity with other modules, such as the Engine Control Unit (ECU) [152], facilitates not only real-time error display on the dashboard but also error storage for future analysis and reference.

[1] PCB Piezotronics. (2024). Frequency range of 10 Hz to 1000 Hz for measuring vibration in the outer tie rod of a car suspension. Retrieved from [https://www.ni.com/en/shop/data-acquisition/sensor-fundamentals/measuring-vibration-with-accelerometers.html]

V. CONCLUSION

This thesis represents a substantial contribution to the field of fault detection in a simulated car's gear steering system, with a specific focus on suspension failure in vehicles. This issue is critical, as suspension failures pose significant risks on the road and can lead to severe, sometimes fatal, car accidents. The research employed a combination of simulation, data analysis, signal processing, and classification to address this issue thoroughly.

The main findings of the study are centered around simulated wear and tear, data collection, feature extraction methods, classification algorithms, robustness analysis, and identifying the best-performing approaches. The study successfully simulated wear in the outer tie rod, generating faulty signals that mimic real-world conditions. The collected data from these simulations provided a baseline for healthy signals.

Feature extraction was conducted using two distinct methods wavelet scattering and discrete wavelet transform. Their effectiveness in capturing signal characteristics was thoroughly evaluated. Support Vector Machines (SVM) and Neural Networks (NN) were used as classification algorithms to differentiate between normal and faulty signals. The robustness of the system was tested by introducing various types of noise, including rough road, tire, and engine noises. Analyses were performed on signals in conditions without noise, with noise, and after denoising.

The study highlighted "Wavelet Scattering SVM PCA" as the best-performing method in the first test for noise scenarios, achieving an outstanding mean accuracy of 99.1%. In the second test, which incorporated noisy and noise-free scenarios, "Wavelet Scattering SVM PCA" again proved its effectiveness, maintaining a high accuracy rate

post-denoising at 97.9%. The third test, introducing noise into road obstacle detection, saw the "Wavelet Scattering RNN ADAM" model as the standout, with a mean accuracy of 97.8%. In the fourth and final test, incorporating both noise and denoising, the "Wavelet Scattering SVM Tuned LDA" emerged as the top performer, achieving a mean accuracy of 98.4%.

While the research has its limitations, particularly within the simulated environment, it lays a solid foundation for future exploration. Real-world conditions may vary from simulations, and future research should focus on real-world testing, diverse datasets, and the advancement of machine learning techniques to enhance fault detection accuracy.

The significance of this research lies in its potential applications in the automotive industry, offering a pathway to safer and more reliable vehicles. The methodologies and findings presented in this thesis not only contribute to the advancement of automotive safety and steering system fault detection but also play a critical role in enhancing overall road safety.

This thesis, therefore, not only contributes to the technological advancements in the field but also honors the memory of those affected by suspension-related accidents, aiming to make roads safer for everyone.

REFERENCES

[1] A. French, “National Highway Traffic Safety Administration (NHTSA) Notes,” *Ann Emerg Med*, vol. 35, no. 6, pp. 0623–0624, 2000, doi 10.1067/mem.2000.106831.

[2] A. Ohnsman, “Not His Wish Star Of Disney’s ‘Aladdin’ Sues Tesla For Defective Model 3 In Hollywood Crash,” *Forbes*, May 2019.

[3] M. Wozniak, A. Rylski, and K. Siczek, “The Measurement of the Wear of Tie Rod End Components,” *Strojniški vestnik - Journal of Mechanical Engineering*, vol. 68, no. 2, pp. 101–125, 2022, doi 10.5545/sv-jme.2021.7389.

[4] J. C. Watrin, H. Makich, B. Haddag, M. Nouari, and X. Grandjean, “Analytical modelling of the ball pin and plastic socket contact in a ball joint,” in *Congrès français de mécanique*, Lille, France, Aug. 2017. doi hal-03325718.

[5] A. S. Sener, “Fatigue life resolution of the steering wheel tie rod of a LCV with FEA,” *Mechanika*, vol. 23, no. 5, pp. 622–629, 2017, doi 10.5755/j01.mech.23.5.16078.

[6] L. Natrayan, E. Aravindaraj, M. S. Santhosh, and M. S. Kumar, “ANALYSIS AND OPTIMIZATION OF CONNECTING TIE ROD ASSEMBLY IN AGRICULTURE APPLICATION,” *Acta Mechanica Malaysia*, vol. 2, no. 1, pp. 06–10, Jan. 2019, doi 10.26480/amm.01.2019.06.10.

[7] Y. Tomikawa, “PIEZOELECTRIC ANGULAR ACCELERATION SENSOR.” 2003.

[8] S. E. El-Khamy and H. A. Elsayed, “Classification of Multi-User Chirp Modulation Signals Using Wavelet Higher-Order-Statistics Features and Artificial Intelligence Techniques,” *International Journal of Communications, Network and System Sciences*, vol. 05, no. 09, pp. 520–533, 2012, doi 10.4236/ijcns.2012.59063.

[9] K. Daqrouq, T. A. Hilal, M. Sherif, S. El-Hajjar, and A. R. Al-Qawasmi, “Speaker identification system using wavelet transform and neural network,” in *2009 International Conference on Advances in Computational Tools for Engineering Applications, ACTEA 2009*, 2009, pp. 559–564. doi 10.1109/ACTEA.2009.5227953.

[10] M. , N. M. , H. J. , & K. S. H. Heydarzadeh, “Non_invasive_Gearbox_Fault_Diagnosis_Usi,” *IEEE international conference on acoustics, speech and signal processing (ICASSP)* , pp. 371–375, 2017, doi 10.1109/icassp.2017.7952180.

[11] S. Nahak, A. Pathak, and G. Saha, “Fragment-level classification of ECG arrhythmia using wavelet scattering transform,” *Expert Syst Appl*, vol. 224, p. 120019, 2023, doi 10.1016/j.eswa.2023.120019.

[12] S. Wegerich, “Similarity-based modeling of vibration features for fault detection and identification,” *Sensor Review*, vol. 25, no. 2, pp. 114–122, 2005, doi 10.1108/02602280510585691.

[13] P. , & P. V. Khaire, “A smart fault identification system for ball bearing using simulation-driven vibration analysis,” *Archive of Mechanical Engineering*, pp. 247–270, Nov. 2023, doi 10.24425/ame.2023.145583.

[14] C. Pravin and V. Ojha, “A Novel ECG Signal Denoising Filter Selection Algorithm Based on Conventional Neural Networks,” in *Proceedings - 19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020, pp. 1094–1100. doi 10.1109/ICMLA51294.2020.00176.

[15] A. Patil, C. Deshmukh, and A. R. Panat, “Feature extraction of EEG for emotion recognition using Hjorth features and higher order crossings,” in *Conference on Advances in Signal Processing, CASP 2016*, Institute of Electrical and Electronics Engineers Inc., Nov. 2016, pp. 429–434. doi 10.1109/CASP.2016.7746209.

[16] A. Stetco *et al.*, “Machine learning methods for wind turbine condition monitoring A review,” *Renewable Energy*, vol. 133. Elsevier Ltd, pp. 620–635, Nov. 2019. doi 10.1016/j.renene.2018.10.047.

[17] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, “A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals,” *Sensors (Switzerland)*, vol. 17, no. 2, Feb. 2017, doi 10.3390/s17020425.

[18] R. G. T. K. Hazra1, “Comparing Wavelet and Wavelet Packet Image Denoising Using Thresholding Techniques,” *International Journal of Science and Research (IJSR)*, vol. 5, no. 6, pp. 790–796, Nov. 2016, doi 10.21275/v5i6.nov164305.

[19] Dr. S. Kumar, “Image Denoising Technique Using Trimmed Based Median Bilateral Filtering Method,” *International Journal Of Engineering And Computer Science*, Nov. 2017, doi 10.18535/ijecs/v6i6.15.

[20] A. Oppenheim *et al.*, “Digital Signal Processing 24.0 Digital Signal Processing Academic and Research Staff Part-time Assistants/Special Projects.”

[21] J. Guo and Q. Chen, “Image denoising based on nonconvex anisotropic total-variation regularization,” *Signal Processing*, vol. 186, Nov. 2021, doi 10.1016/j.sigpro.2021.108124.

[22] P. Singh, S. Shahnawazuddin, and G. Pradhan, “An Efficient ECG Denoising Technique Based on Non-local Means Estimation and Modified Empirical Mode Decomposition,” *Circuits Syst Signal Process*, vol. 37, no. 10, pp. 4527–4547, Nov. 2018, doi 10.1007/s00034-018-0777-9.

[23] J. Shlens, “A Tutorial on Principal Component Analysis.” 2014. doi 10.48550/arXiv.1404.1100.

[24] S. Huai and S. Zhang, “A novel sparse representation algorithm for AIS real-time signals,” *EURASIP J Wirel Commun Netw*, vol. 2018, no. 1, Nov. 2018, doi 10.1186/s13638-018-1244-9.

[25] Y. Zheng, W. Jiang, and X. Qiu, “A Variable Parameter Method Based on Linear Extended State Observer for Position Tracking,” 2022, doi 10.3390/act110200.

[26] J. B. Tary, R. H. Herrera, and M. Van Der Baan, “Analysis of time-varying signals using continuous wavelet and synchrosqueezed transforms,” *Philosophical Transactions of the Royal Society A Mathematical, Physical and*

Engineering Sciences, vol. 376, no. 2126, Aug. 2018, doi 10.1098/rsta.2017.0254.

[27] B. Belkacemi, S. Saad, Z. Ghemari, F. Zaamouche, and A. Khazzane, "Detection of induction motor improper bearing lubrication by discrete wavelet transforms (DWT) decomposition," *Instrumentation Mesure Metrologie*, vol. 19, no. 5, pp. 347–354, Nov. 2020, doi 10.18280/i2m.190504.

[28] D. Strömbergsson, P. Marklund, K. Berglund, and P. E. Larsson, "Bearing monitoring in the wind turbine drivetrain A comparative study of the FFT and wavelet transforms," *Wind Energy*, vol. 23, no. 6, pp. 1381–1393, Nov. 2020, doi 10.1002/we.2491.

[29] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 8, pp. 1872–1886, 2013, doi 10.1109/TPAMI.2012.230.

[30] S. Zhou, S. Qian, W. Chang, Y. Xiao, and Y. Cheng, "A novel bearing multi-fault diagnosis approach based on weighted permutation entropy and an improved SVM ensemble classifier," *Sensors (Switzerland)*, vol. 18, no. 6, Nov. 2018, doi 10.3390/s18061934.

[31] G. Jain, M. Sharma, and B. Agarwal, "Optimizing semantic LSTM for spam detection," *International Journal of Information Technology (Singapore)*, vol. 11, no. 2, pp. 239–250, Nov. 2019, doi 10.1007/s41870-018-0157-5.

[32] M. Cao, W. Xu, W. Ostachowicz, and Z. Su, "Damage identification for beams in noisy conditions based on Teager energy operator-wavelet transform modal

curvature," *J Sound Vib*, vol. 333, no. 6, pp. 1543–1553, Nov. 2014, doi 10.1016/j.jsv.2013.11.003.

[33] C. Cortes, V. Vapnik, and L. Saitta, "Support-Vector Networks Editor," *Machine Leaming*, vol. 20. Kluwer Academic Publishers, pp. 273–297, 1995. doi 10.1007/BF00994018.

[34] D. R. Cutler *et al.*, "Random forests for classification in ecology," *Ecology*, vol. 88, no. 11, pp. 2783–2792, Nov. 2007, doi 10.1890/07-0539.1.

[35] G. C, J. A. Mangai, and M. Bansal, "An Investigation of Ensemble Learning Algorithms for Fault Diagnosis of Roller Bearing," 2022. doi 10.3233/apc220016.

[36] C. Li, R. V. Sánchez, G. Zurita, M. Cerrada, and D. Cabrera, "Fault diagnosis for rotating machinery using vibration measurement deep statistical feature learning," *Sensors (Switzerland)*, vol. 16, no. 6, Nov. 2016, doi 10.3390/s16060895.

[37] Z. G. Su, Q. Hu, and T. Denoeux, "A Distributed Rough Evidential K-NN Classifier Integrating Feature Reduction and Classification," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 8, pp. 2322–2335, Nov. 2021, doi 10.1109/TFUZZ.2020.2998502.

[38] E. Zio, P. Baraldi, and I. C. Popescu, "A fuzzy decision tree for fault classification," *Risk Analysis*, vol. 28, no. 1, pp. 49–67, Nov. 2008, doi 10.1111/j.1539-6924.2008.01002.x.

[39] A. Spyros *et al.*, "Towards Continuous Enrichment of Cyber Threat Intelligence A Study on a Honeypot Dataset," in *Proceedings of the 2022 IEEE International*

Conference on Cyber Security and Resilience, CSR 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 267–272. doi 10.1109/CSR54599.2022.9850295.

[40] K. K. Verma, B. M. Singh, and A. Dixit, "A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system," *International Journal of Information Technology (Singapore)*, vol. 14, no. 1, pp. 397–410, Feb. 2022, doi 10.1007/s41870-019-00364-0.

[41] D. Dey, B. Chatterjee, S. Dalai, S. Munshi, and S. Chakravorti, "A deep learning framework using convolution neural network for classification of impulse fault patterns in transformers with increased accuracy," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 24, no. 6, pp. 3894–3897, Dec. 2017, doi 10.1109/TDEI.2017.006793.

[42] Z. Yu, L. Zhang, and J. Kim, "The Performance Analysis of PSO-ResNet for the Fault Diagnosis of Vibration Signals Based on the Pipeline Robot," *Sensors (Basel)*, vol. 23, no. 9, May 2023, doi 10.3390/s23094289.

[43] H. Li, X. Yue, Z. Wang, W. Wang, H. Tomiyama, and L. Meng, "A survey of Convolutional Neural Networks —From software to hardware and the applications in measurement," in *Measurement Sensors*, Elsevier Ltd, Dec. 2021. doi 10.1016/j.measen.2021.100080.

[44] M. A. Atoui and A. Cohen, "Coupling data-driven and model-based methods to improve fault diagnosis," *Comput Ind*, vol. 128, Jun. 2021, doi 10.1016/j.compind.2021.103401.

[45] J. Yan and X. Wang, “Unsupervised and semi-supervised learning the next frontier in machine learning for plant systems biology,” *Plant Journal*. John Wiley and Sons Inc, 2022. doi 10.1111/tpj.15905.

[46] H. Zhu, Z. He, J. Wei, J. Wang, and H. Zhou, “Bearing fault feature extraction and fault diagnosis method based on feature fusion,” *Sensors*, vol. 21, no. 7, Apr. 2021, doi 10.3390/s21072524.

[47] J. Peffers, “The Design Science Research Process A Model for Producing and Presenting Information Systems Research,” 2006. [Online]. Available http//rightsstatements.org/page/InC/1.0/?language=en

[48] A. Aboazoum, “An Overview of the most Common Vehicle Suspension Problems,” *Brilliance Research of Artificial Intelligence*, vol. 2, no. 3, pp. 120–124, Nov. 2022, doi 10.47709/brilliance.v2i3.1655.

[49] R. Kumar, T. Sharma, A. Shekhar, and N. S. Vyas, “Primary suspension failure analysis in FIAT type LHB bogies and life estimation,” *Eng Fail Anal*, vol. 138, Nov. 2022, doi 10.1016/j.engfailanal.2022.106320.

[50] S. D. Shinde, S. Maheshwari, and S. Kumar, “Literature review on analysis of various Components of McPherson suspension,” *Materials Today Proceedings*, vol. 5. pp. 19102–19108, 2018. doi 10.18280/mmep.070411.

[51] M. Hamed, B. Tesfa, F. Gu, and A. D. Ball, “Effects of Tyre Pressure on Vehicle Suspension Performance,” *International Letters of Chemistry, Physics and Astronomy*, vol. 55, pp. 102–111, Nov. 2015, doi 10.56431/p-l6t1lc.

[52] P. B. Patil and P. D. Darade, “Vibrational Analysis, Life Prediction and Optimization of Pitman Arm Using FEM,” *International Journal of*

Computational Engineering Research, vol. 8. pp. 2250–3005, 2018. doi 10.18280/mmep.070411.

[53] K. Reza Kashyzadeh, M. Jafar Ostad-Ahmad-Ghorabi, and A. Arghavan, "Mediterranean Journal of Modeling and Simulation Fatigue life prediction of package of suspension automotive under random vibration based on road roughness," 2015. Accessed Nov. 26, 2023. [Online]. Available https//www.asjp.cerist.dz/en/downArticle/10/4/1/674

[54] W. Xiukun, G. Kun, and J. Limin, "Fault Isolation of Light Rail Vehicle Suspension System Based on D-S Evidence Theory," in *the 32nd Chinese Control Conference*, Xi'an,China IEEE, 2013, pp. 6116–6121.

[55] P. Jin, W. Xue, and K. Li, "Actuator Fault Estimation for Vehicle Active Suspensions Based on Adaptive Observer and Genetic Algorithm," *Shock and Vibration*, vol. 2019, 2019, doi 10.1155/2019/1783850.

[56] X. Zhu, Y. Xia, S. Chai, and P. Shi, "Fault detection for vehicle active suspension systems in finite-frequency domain," *IET Control Theory and Applications*, vol. 13, no. 3, pp. 387–394, Nov. 2019, doi 10.1049/iet-cta.2018.5922.

[57] Z. Mao, Y. Zhan, G. Tao, B. Jiang, and X. G. Yan, "Sensor Fault Detection for Rail Vehicle Suspension Systems with Disturbances and Stochastic Noises," *IEEE Trans Veh Technol*, vol. 66, no. 6, pp. 4691–4705, Nov. 2017, doi 10.1109/TVT.2016.2628054.

[58] Y. S. Takashi KOJIMA, "Fault Detection of Vertical Dampers of Railway Vehicle Based on Phase Difference of Vibrations," vol. 54, no. 3. 2013. doi 10.2219/rtriqr.54.139.

[59] S. Yin and Z. Huang, “Performance Monitoring for Vehicle Suspension System via Fuzzy Positivistic C-Means Clustering Based on Accelerometer Measurements,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2613–2620, Nov. 2015, doi 10.1109/TMECH.2014.2358674.

[60] X. Wei, L. Jia, and H. Liu, “A comparative study on fault detection methods of rail vehicle suspension systems based on acceleration measurements,” *Vehicle System Dynamics*, vol. 51, no. 5, pp. 700–720, 2013, doi 10.1080/00423114.2013.767464.

[61] J. S. Sakellariou, K. A. Petsounis, and S. D. Fassois, “Vibration based fault diagnosis for railway vehicle suspensions via a functional model based method A feasibility study,” *Journal of Mechanical Science and Technology*, vol. 29, no. 2, pp. 471–484, Nov. 2015, doi 10.1007/s12206-015-0107-0.

[62] T. C. I. Aravanis, J. S. Sakellariou, and S. D. Fassois, “A stochastic Functional Model based method for random vibration based robust fault detection under variable non–measurable operating conditions with application to railway vehicle suspensions,” *J Sound Vib*, vol. 466, Feb. 2020, doi 10.1016/j.jsv.2019.115006.

[63] A. A. A. Rahim, S. Abdullah, S. S. K. Singh, and M. Z. Nuawi, “Selection of the optimum decomposition level using the discrete wavelet transform for automobile suspension system,” *Journal of Mechanical Science and Technology*, vol. 34, no. 1, pp. 137–142, Jan. 2020, doi 10.1007/s12206-019-1213-1.

[64] S. Azadi and A. Soltani, “Fault detection of vehicle suspension system using wavelet analysis,” *Vehicle System Dynamics*, vol. 47, no. 4, pp. 403–418, Apr. 2009, doi 10.1080/00423110802094298.

[65] N. Khan, “Vibration Response of a Gearbox having Gear with Teeth Root Cracks,” *Int J Res Appl Sci Eng Technol*, vol. 7, no. 6, pp. 2015–2020, Nov. 2019, doi 10.22214/ijraset.2019.6339.

[66] H. Ahmed and A. K. Nandi, “Compressive Sampling and Feature Ranking Framework for Bearing Fault Classification With Vibration Signals,” *IEEE Access*, vol. 6, pp. 44731–44746, Nov. 2018, doi 10.1109/ACCESS.2018.2865116.

[67] T. Plante, A. Nejadpak, and C. X. Yang, “Faults detection and failures prediction using vibration analysis,” in *AUTOTESTCON (Proceedings)*, Institute of Electrical and Electronics Engineers Inc., Nov. 2015, pp. 227–231. doi 10.1109/AUTEST.2015.7356493.

[68] S. Fu, K. Liu, Y. Xu, and Y. Liu, “Rolling bearing diagnosing method based on time domain analysis and adaptive fuzzy C -means clustering,” *Shock and Vibration*, vol. 2016, 2016, doi 10.1155/2016/9412787.

[69] G. Dong, J. Chen, and F. Zhao, “Monitoring of the Looseness in Cargo Bolts under Random Excitation Based on Vibration Transmissibility,” *Shock and Vibration*, vol. 2021, 2021, doi 10.1155/2021/8841940.

[70] M. Parzinger, L. Hanfstaengl, F. Sigg, U. Spindler, U. Wellisch, and M. Wirnsberger, “Residual analysis of predictive modelling data for automated fault detection in building’s heating, ventilation and air conditioning systems,”

Sustainability (Switzerland), vol. 12, no. 17, Nov. 2020, doi 10.3390/SU12176758.

[71] Y. Huang, C. H. Chen, and C. J. Huang, "Motor fault detection and feature extraction using rnn-based variational autoencoder," *IEEE Access*, vol. 7, pp. 139086–139096, 2019, doi 10.1109/ACCESS.2019.2940769.

[72] M. Hashemi and M. S. Safizadeh, "Design of a fuzzy model based on vibration signal analysis to auto-detect the gear faults," *Industrial Lubrication and Tribology*, vol. 65, no. 3, pp. 194–201, 2013, doi 10.1108/00368791311311196.

[73] Z. Wang, J. Yang, H. Li, D. Zhen, Y. Xu, and F. Gu, "Fault identification of broken rotor bars in induction motors using an improved cyclic modulation spectral analysis," *Energies (Basel)*, vol. 12, no. 17, Nov. 2019, doi 10.3390/en12173279.

[74] J. Guo, Z. Shi, H. Li, D. Zhen, F. Gu, and A. D. Ball, "Early fault diagnosis for planetary gearbox based wavelet packet energy and modulation signal bispectrum analysis," *Sensors (Switzerland)*, vol. 18, no. 9, Nov. 2018, doi 10.3390/s18092908.

[75] Y. Li, X. Liang, Y. Chen, Z. Chen, and J. Lin, "Wheelset bearing fault detection using morphological signal and image analysis," *Struct Control Health Monit*, vol. 27, no. 11, Nov. 2020, doi 10.1002/stc.2619.

[76] T. D. Popescu and D. Aiordachioaie, "Rolling Element Bearing Fault Detection Using Vibrating Signals Segmentation," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Institute of Electrical

and Electronics Engineers Inc., Nov. 2018, pp. 940–947. doi 10.1109/ETFA.2018.8502555.

[77] H. Zhu, Z. He, J. Wei, J. Wang, and H. Zhou, “Bearing fault feature extraction and fault diagnosis method based on feature fusion,” *Sensors*, vol. 21, no. 7, Nov. 2021, doi 10.3390/s21072524.

[78] H. Liang, Y. Chen, S. Liang, and C. Wang, “Fault detection of stator inter-turn short-circuit in pmsm on stator current and vibration signal,” *Applied Sciences (Switzerland)*, vol. 8, no. 9, Nov. 2018, doi 10.3390/app8091677.

[79] M. Rahnama and A. Vahedi, “Application of acoustic signals for rectifier fault detection in brushless synchronous generator,” *Archives of Acoustics*, vol. 44, no. 2, pp. 267–276, 2019, doi 10.24425/aoa.2019.128490.

[80] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, “A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals,” *Sensors (Switzerland)*, vol. 17, no. 2, Nov. 2017, doi 10.3390/s17020425.

[81] A. Shaheryar, X.-C. Yin, and W. Y. Ramay, “Robust Feature Extraction on Vibration Data under Deep-Learning Framework An Application for Fault Identification in Rotary Machines,” *International Journal of Computer Applications*, vol. 167, no. 4. pp. 975–8887, 2017.

[82] S. E. Pandarakone, Y. Mizuno, and H. Nakamura, “A comparative study between machine learning algorithm and artificial intelligence neural network in detecting minor bearing fault of induction motors,” *Energies (Basel)*, vol. 12, no. 11, 2019, doi 10.3390/en12112105.

[83] Hamilton and James D, “Time Series Analysis,” 2021. doi 10.1515/9780691218632.

[84] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde, “Bayesian measures of model complexity and fit,” *J. R. Statist. Soc. B*, vol. 64. pp. 583–639, 2002. doi 10.1111/1467-9868.00353.

[85] P. Mayo, O. Karakus, R. Holmes, and A. Achim, “Representation learning via cauchy convolutional sparse coding,” *IEEE Access*, vol. 9, pp. 100447–100459, 2021, doi 10.1109/ACCESS.2021.3096643.

[86] P. C. Nugroho, R. Widadi, and D. Zulherman, “Hand and Foot Movement of Motor Imagery Classification Using Wavelet Packet Decomposition and Multilayer Perceptron Backpropagation.” 2021. doi 10.2991/aer.k.210810.042.

[87] I. M. Johnstone and B. W. Silverman, “Empirical bayes selection of wavelet thresholds,” *Ann Stat*, vol. 33, no. 4, pp. 1700–1752, Aug. 2005, doi 10.1214/009053605000000345.

[88] S. G. (Stéphane G.) Mallat and Gabriel. Peyré, *A wavelet tour of signal processing the sparse way*. 1999. doi 10.1016/B978-0-12-374370-1.X0001-8.

[89] I. Daubechies, *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992. doi 10.1137/1.9781611970104.

[90] V. J. Samar, A. Bopardikar, R. Rao, and K. Swartz, “Wavelet Analysis of Neuroelectric Waveforms A Conceptual Tutorial,” *Brain and Language*, vol. 66. Brain and Language, pp. 7–60, 1999. doi 10.1006/brln.1998.2024.

[91] C. S. Burrus, R. Gopinath, and H. Guo, “Wavelets and Wavelet Transforms.” OpenStax-CNX, 2022. doi cnx-org-col11454.

[92] S. Sridhar, P. Rajesh Kumar, and K. V. Ramanaiah, “Wavelet Transform Techniques for Image Compression – An Evaluation,” *International Journal of Image, Graphics and Signal Processing*, vol. 6, no. 2, pp. 54–67, Jan. 2014, doi 10.5815/ijigsp.2014.02.07.

[93] J. C. i Roura and J. L. R. Martínez, *Transient Analysis and Motor Fault Detection using the Wavelet Transform*. INTECH Open Access Publisher, 2011. doi 10.5772/15377.

[94] G. C. (Geoffrey C. Green, *Wavelet-based denoising of cardiac PET data*. Library and Archives Canada = Bibliothèque et Archives Canada, 2005. doi 10.22215/etd/2005-07963.

[95] V. S. Chourasia and A. K. Tiwari, “Design methodology of a new wavelet basis function for fetal phonocardiographic signals,” *The Scientific World Journal*, vol. 2013, 2013, doi 10.1155/2013/505840.

[96] M. S. Mechee, Z. M. Hussain, and Z. I. Salman, “Wavelet Theory Applications of the Wavelet.” IntechOpen, 2021. doi 10.5772/intechopen.94911.

[97] S. G. Mallat, “A Theory for Multiresolution Signal Decomposition The Wavelet Representation,” *IEEE Trans Pattern Anal Mach Intell*, vol. I, no. 7, 1989, doi 10.1109/34.192463.

[98] M. Kobayashi and K. Nakano, “Development of Quasi-Shift-Invariant Complex Discrete Wavelet Transform,” *Journal of Signal Processing*. pp. 211–244, 2017. doi 10.2299/jsp.21.211.

[99] X. He and H. Guan, “Multiresolution analysis of precipitation teleconnections with large-scale climate signals A case study in South Australia,” *Water Resour Res*, vol. 49, no. 10, pp. 6995–7008, Oct. 2013, doi 10.1002/wrcr.20560.

[100] J. Andén and S. Mallat, “Deep scattering spectrum,” *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, Aug. 2014, doi 10.1109/TSP.2014.2326991.

[101] L. Sifre and S. Mallat, “Rotation, scaling and deformation invariant scattering for texture discrimination,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1233–1240. doi 10.1109/CVPR.2013.163.

[102] S. Mallat, “Group Invariant Scattering,” *Commun Pure Appl Math*, vol. 65, no. 10, pp. 1331–1398, Oct. 2012, doi 10.1002/cpa.21413.

[103] C. Szegedy *et al.*, “2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),” IEEE Computer Society, 2015. doi 10.1109/CVPR.2015.7298594.

[104] J. Bruna and S. Mallat, “Invariant scattering convolution networks,” *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 8, pp. 1872–1886, 2013, doi 10.1109/TPAMI.2012.230.

[105] Z. Li, “An Explicit Algorithm for the Construction of 3-Band Wavelet Frames Based on FMRA,” *Applied and Computational Mathematics*, vol. 7, no. 3, p. 155, 2018, doi 10.11648/j.acm.20180703.21.

[106] H. Zhou *et al.*, “Weight-Variable Scattering Convolution Networks and Its Application in Electromagnetic Signal Classification,” *IEEE Access*, vol. 7, pp. 175889–175896, 2019, doi 10.1109/ACCESS.2019.2957519.

[107] I. Waldspurger, “Exponential decay of scattering coefficients.” IEEE, 2017. doi 10.1109/SAMPTA.2017.8024473.

[108] T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning.” Springer New York, pp. 225–256, 2001. doi 10.1007/978-0-387-21606-5.

[109] S. M. Holland, “PRINCIPAL COMPONENTS A N ALYSI S (PCA).” Department of Geology, University of Georgia, Athens, GA 30602-2501, 2019. Accessed Nov. 20, 2023. [Online]. Available http//strata.uga.edu/8370/handouts/pcaTutorial.pdf

[110] A. E. Maxwell, T. A. Warner, and F. Fang, “Implementation of machine-learning classification in remote sensing An applied review,” *International Journal of Remote Sensing*, vol. 39, no. 9. Taylor and Francis Ltd., pp. 2784–2817, Nov. 2018. doi 10.1080/01431161.2018.1433343.

[111] A. Khelifi, N. M. Ben Lakhal, H. Gharsallaoui, and O. Nasri, “Artificial Neural Network-based Fault Detection,” in *2018 5th International Conference on Control, Decision and Information Technologies, CoDIT 2018*, Institute of Electrical and Electronics Engineers Inc., Jun. 2018, pp. 1017–1022. doi 10.1109/CoDIT.2018.8394963.

[112] Y. Zhang and L. Wu, “Classification of fruits using computer vision and a multiclass support vector machine,” *Sensors (Switzerland)*, vol. 12, no. 9, pp. 12489–12505, Nov. 2012, doi 10.3390/s120912489.

[113] Í. Santana, B. Serrano, M. Schiffer, and T. Vidal, “Support Vector Machines with the Hard-Margin Loss Optimal Training via Combinatorial Benders’ Cuts,” Nov. 2022, doi 10.48550/arXiv.2207.07690.

[114] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, “Reproducing Kernel Hilbert Space, Mercer’s Theorem, Eigenfunctions, Nystr\"om Method, and Use of Kernels in Machine Learning Tutorial and Survey,” Jun. 2021, [Online]. Available http//arxiv.org/abs/2106.08443

[115] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, “Reproducing Kernel Hilbert Space, Mercer’s Theorem, Eigenfunctions, Nyström Method, and Use of Kernels in Machine Learning Tutorial and Survey,” Nov. 2021, doi 10.48550/arXiv.2106.08443.

[116] R. G. Brereton and G. R. Lloyd, “Support Vector Machines for classification and regression,” *Analyst*, vol. 135, no. 2. Royal Society of Chemistry, pp. 230–267, 2010. doi 10.1039/b918972f.

[117] J. Hajewski, S. Oliveira, and D. Stewart, “Smoothed hinge loss and ℓ1 support vector machines,” in *IEEE International Conference on Data Mining Workshops, ICDMW*, IEEE Computer Society, Nov. 2019, pp. 1217–1223. doi 10.1109/ICDMW.2018.00174.

[118] R. Hammoud, “MATHEMATICS BEHIND MACHINE LEARNING,” 2023. [Online]. Available

https//scholarworks.lib.csusb.edu/etdDissertations.1778.https//scholarworks.lib.csusb.edu/etd/1778

[119] M. Shamsi and S. Beheshti, “Separability and Scatteredness (S&S) Ratio-Based Efficient SVM Regularization Parameter, Kernel, and Kernel Parameter Selection,” Nov. 2023, doi 10.48550/arXiv.2305.10219.

[120] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, “A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data,” *Frontiers in Energy Research*, vol. 9. Frontiers Media S.A., Nov. 2021. doi 10.3389/fenrg.2021.652801.

[121] Y. Xu and R. Goodacre, “On Splitting Training and Validation Set A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning,” *J Anal Test*, vol. 2, no. 3, pp. 249–262, Jul. 2018, doi 10.1007/s41664-018-0068-2.

[122] D. M. Belete and M. D. Huchaiah, “Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results,” *International Journal of Computers and Applications*, vol. 44, no. 9, pp. 875–886, 2022, doi 10.1080/1206212X.2021.1974663.

[123] Reda Yacouby and Dustin Axman, “Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models.” Association for Computational Linguistics, 2020. doi 10.18653/v1/2020.eval4nlp-1.9.

[124] B. Choi and S. Jo, “A Low-Cost EEG System-Based Hybrid Brain-Computer Interface for Humanoid Robot Navigation and Recognition,” *PLoS One*, vol. 8, no. 9, Sep. 2013, doi 10.1371/journal.pone.0074583.

[125] B. Kwon, H. Song, and S. Lee, “Accurate Blind Lempel-Ziv-77 Parameter Estimation via 1-D to 2-D Data Conversion over Convolutional Neural Network,” *IEEE Access*, vol. 8, pp. 43965–43979, 2020, doi 10.1109/ACCESS.2020.2977827.

[126] F. Iglesias and T. Zseby, “Analysis of network traffic features for anomaly detection,” *Mach Learn*, vol. 101, no. 1–3, pp. 59–84, Nov. 2015, doi 10.1007/s10994-014-5473-9.

[127] A. Shewalkar, D. nyavanandi, and S. A. Ludwig, “Performance Evaluation of Deep neural networks Applied to Speech Recognition Rnn, LSTM and GRU,” *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 235–245, Nov. 2019, doi 10.2478/jaiscr-2019-0006.

[128] A. Shewalkar, D. nyavanandi, and S. A. Ludwig, “Performance Evaluation of Deep neural networks Applied to Speech Recognition Rnn, LSTM and GRU,” *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 235–245, Oct. 2019, doi 10.2478/jaiscr-2019-0006.

[129] H. Sak, A. Senior, and F. Beaufays, “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition,” Nov. 2014, doi 10.48550/arXiv.1402.1128.

[130] G. Jain, M. Sharma, and B. Agarwal, “Optimizing semantic LSTM for spam detection,” *International Journal of Information Technology (Singapore)*, vol. 11, no. 2, pp. 239–250, Jun. 2019, doi 10.1007/s41870-018-0157-5.

[131] M. U. Abbasi, A. Rashad, A. Basalamah, and M. Tariq, “Detection of Epilepsy Seizures in Neo-Natal EEG Using LSTM Architecture,” *IEEE Access*, vol. 7, pp. 179074–179085, 2019, doi 10.1109/ACCESS.2019.2959234.

[132] Y. and S. Q. Hao, “The implementation of a Deep Recurrent Neural Network Language Model on a Xilinx FPGA 1.” 2017. doi 10.48550/arXiv.1710.10296.

[133] M. Tan, S. Yuan, S. Li, Y. Su, H. Li, and F. H. He, “Ultra-Short-Term Industrial Power Demand Forecasting Using LSTM Based Hybrid Ensemble Learning,” *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 2937–2948, Nov. 2020, doi 10.1109/TPWRS.2019.2963109.

[134] S. Ruder, “An overview of gradient descent optimization algorithms,” Nov. 2016, doi 10.48550/arXiv.1609.04747.

[135] A. Napieralski and R. Nowak, “Basecalling Using Joint Raw and Event Nanopore Data Sequence-to-Sequence Processing,” *Sensors*, vol. 22, no. 6, Mar. 2022, doi 10.3390/s22062275.

[136] Y. Lu and F. M. Salem, “Simplified Gating in Long Short-term Memory (LSTM) Recurrent Neural Networks Index Terms-Recurrent Neural Networks (RNN), Long Short-term Memory (LSTM), Stochastic Gradient Descent.” IEEE, 2017. doi 10.1109/MWSCAS.2017.8053244.

[137] G. Van Houdt, C. Mosquera, and G. Nápoles, “A review on the long short-term memory model,” *Artif Intell Rev*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, doi 10.1007/s10462-020-09838-1.

[138] Z. Huang, H. Chen, C. J. Hsu, W. H. Chen, and S. Wu, “Credit rating analysis with support vector machines and neural networks A market comparative study,” *Decis Support Syst*, vol. 37, no. 4, pp. 543–558, Nov. 2004, doi 10.1016/S0167-9236(03)00086-1.

[139] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘Why should i trust you?’ Explaining the predictions of any classifier,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Nov. 2016, pp. 1135–1144. doi 10.1145/2939672.2939778.

[140] L. Vanajakshi and L. R. Rilett, *A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed*. IEEE Intelligent Vehicles Symposium, 2004. doi 10.1109/IVS.2004.1336380.

[141] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” Nov. 2019, doi 10.48550/arXiv.1904.09237.

[142] Y. Xi, C. Ren, Q. Tian, Y. Ren, X. Dong, and Z. Zhang, “Exploitation of Time Series Sentinel-2 Data and Different Machine Learning Algorithms for Detailed Tree Species Classification,” *IEEE J Sel Top Appl Earth Obs Remote Sens*, vol. 14, pp. 7589–7603, 2021, doi 10.1109/JSTARS.2021.3098817.

[143] L. Yang and A. Shami, “On Hyperparameter Optimization of Machine Learning Algorithms Theory and Practice,” Jul. 2020, doi 10.1016/j.neucom.2020.07.061.

[144] Farhan Ahnaf Rashid and F. A. Rashid, “Development of a Machine Learning Based Fall Detection System for the Elderly and Disabled CERTIFICATE OF ORIGINAL AUTHORSHIP.” 2021. Accessed Nov. 20, 2023. [Online]. Available http//hdl.handle.net/10453/162783

[145] A. R. Mello, J. de Matos, M. R. Stemmer, A. de Souza Britto, and A. L. Koerich, “A Novel Orthogonal Direction Mesh Adaptive Direct Search Approach for SVM Hyperparameter Tuning,” Nov. 2019, doi 10.48550/arXiv.1904.11649.

[146] M. Pule, O. Matsebe, and R. Samikannu, “Application of PCA and SVM in Fault Detection and Diagnosis of Bearings with Varying Speed,” *Math Probl Eng*, vol. 2022, p. 5266054, 2022, doi 10.1155/2022/5266054.

[147] N. Rezazadeh, M. de Oliveira, D. Perfetto, A. De Luca, and F. Caputo, “Classification of Unbalanced and Bowed Rotors under Uncertainty Using Wavelet Time Scattering, LSTM, and SVM,” *Applied Sciences (Switzerland)*, vol. 13, no. 12, Jun. 2023, doi 10.3390/app13126861.

[148] J. Pacheco-Chérrez, J. A. Fortoul-Díaz, F. Cortés-Santacruz, L. María Aloso-Valerdi, and D. I. Ibarra-Zarate, “Bearing fault detection with vibration and acoustic signals Comparison among different machine leaning classification methods,” *Eng Fail Anal*, vol. 139, p. 106515, 2022, doi https//doi.org/10.1016/j.engfailanal.2022.106515.

[149] A. K. Krishnan, E. Nissen, S. Saripalli, R. Arrowsmith, and A. H. Corona, “Change Detection Using Airborne LiDAR Applications to Earthquakes,” in *Experimental Robotics The 13th International Symposium on Experimental Robotics*, J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, Eds., Heidelberg

Springer International Publishing, 2013, pp. 733–743. doi 10.1007/978-3-319-00065-7_49.

[150] V. G. Salunkhe, R. G. Desavale, and S. G. Kumbhar, “Vibration Analysis of Deep Groove Ball Bearing Using Finite Element Analysis and Dimension Analysis,” *J Tribol*, vol. 144, no. 8, Feb. 2022, doi 10.1115/1.4053262.

[151] A. Yang, Y. Zang, L. Xu, L. Li, and D. Tan, “A Systematic Review and Future Development of Automotive Chassis Control Technology,” *Applied Sciences*, vol. 13, no. 21, p. 11859, Oct. 2023, doi 10.3390/app132111859.

[152] J. Schuller, “CHASSIS ARCHITECTURES – Electronic chassis platform – highly integrated ECU for chassis control functions,” 2017, pp. 349–365. doi 10.1007/978-3-658-14219-3_25.

APPENDIX A

5.1. MATLAB Code to Generate Signals for Four Types Road obstacles 5 levels training and testing (With Noise, Denoised)

```
% Load the data
Sivewave = importdata('Sinewave.tab', '\t').data;
Roughness = importdata('Roughness.tab', '\t').data;
Pothole = importdata('Pothole.tab', '\t').data;
Bump = importdata('bump.tab', '\t').data;
% Extract the time vector and acceleration data
t = Sivewave(,1);
accsv = Sivewave(,17);
accrg = Roughness(,17);
accpthl = Pothole(,17);
accbmp = Bump(,17);
% Define the cell array of workspace names
workspaceNames = {'accsv', 'accrg', 'accpthl','accbmp'};
% Loop through the workspace names
for i = 1length(workspaceNames)
   % Get the data for this workspace
   data = evalin('base', workspaceNames{i});

   % Loop through the time points and frequency indices for 1000
   for m = 1length(t)
      for n = 199
         % Compute the new value based on the time and frequency
         if t(m) == 0 % Modified condition
            data(m,n) = 0;
         elseif data(m,1) > 0
            data(m,n+1) = ((data(m,1)*t(m)^2) + (n*2/100000)) / (t(m)^2);
         elseif data(m,1) < 0
            data(m,n+1) = ((data(m,1)*t(m)^2) - (n*2/100000)) / (t(m)^2);
         end
      end
   end

   % Loop through the time points and frequency indices for 100
   for m = 1length(t)
      for n = 1400
```

```
            % Compute the new value based on the time and frequency
            if t(m) == 0 % Modified condition
                data(m,n+100) = 0;
            elseif data(m,1) > 0
                data(m,n+100) = ((data(m,1)*t(m)^2) + (n*2/400)) / (t(m)^2);
            elseif data(m,1) < 0
                data(m,n+100) = ((data(m,1)*t(m)^2) - (n*2/400)) / (t(m)^2);
            end
        end
    end

    % Assign the modified data back to the workspace
    assignin('base', workspaceNames{i}, data);
end
% Delete the first three rows from each dataset
accsv(13,) = [];
accrg(13,) = [];
accpthl(13,) = [];
accbmp(13,) = [];
t(13,)=[];
%labeling my columns
x=ones(1,100);
x2=repmat(2,1,100);
x3=repmat(3,1,100);
x4=repmat(4,1,100);
x5=repmat(5,1,100);
label=([x,x2,x3,x4,x5]);
accsv=[label;accsv];
accrg=[label;accrg];
accpthl=[label;accpthl];
accbmp=[label;accbmp];
AccData=accrg;
noiser = 30 * randn(size(AccData(2end,))); % Gaussian noise with standard deviation
0.1
noisy_signal = AccData(2end,) + noiser;
%AccData= [[accsv1, accrg1, accpthl1,accbmp1];noisy_signal];
% Perform wavelet denoising on each column of the matrix
denoisedMatrix = zeros(size(noisy_signal));
for col = 1size(noisy_signal, 2)
    signal = noisy_signal(, col);
```

```
    % Perform wavelet denoising on the individual signal using wdenoise
    denoisedSignal = wdenoise(signal);
        % Store the denoised signal in the denoised matrix
    denoisedMatrix(, col) = denoisedSignal;
end
AccData=[label;denoisedMatrix];
```

5.2. First Labelling Code for Four Types Road obstacles 5 levels training and testing (With Noise, Denoised)

```
Ts=mean(diff(t));
rng default %fix the random
AccData=AccData(,randperm(size(AccData,2)));
%shuffle the columns
traindata=AccData(2end,);
trainlabel=categorical(AccData(1,));
numClasses = numel(trainlabel);
CT = countlabels(trainlabel);
tbl = table2array(CT(,2));
% Use cvpartition to split data into training and validation sets
c = cvpartition(trainlabel, 'HoldOut', 0.2);
trainIdx = training(c);
testIdx = test(c);
Ptrain = traindata(,trainIdx);
Ttrain = trainlabel(trainIdx);
Ptest = traindata(,testIdx);
Ttest = trainlabel(testIdx);
uniqueLabels = unique(trainlabel);
catnames = {'Normal','Level 1','Level 2','Level 3','Level 4'};
uniqueLabels = renamecats(uniqueLabels,{'1','2','3','4','5'},catnames);
bar(uniqueLabels, tbl);
CTtrain=countlabels(Ttrain);
CTtest=countlabels(Ttest);
tblTrain = table2array(CTtrain(,2));
tblTest = table2array(CTtest(,2));
H = bar(uniqueLabels,[tblTrain, tblTest],'stacked');
legend(H,["Training Set","Test Set"],'Location','NorthEastOutside')
LPtrain=[double(Ttrain);Ptrain];
[ii,jj,kk]=unique(LPtrain(1,));
m=accumarray(kk,(1numel(kk)),[],@(x) {x'});
```

```
out=cell2mat(cellfun(@(x)
LPtrain(,x(randperm(numel(x),1))),m','UniformOutput',false));
%%idx = randperm(size(Ptrain,2),4);
%Fs = 1/Ts;
parfor n = 1numel(out(1,))
   x=out(2end,n);
   subplot(4,2,n);
   plot(t,x);
   if n == 4 || n == 5
      xlabel('Seconds');
   end
   d=dictionary([1,2,3,4,5],catnames);
   title(d(n));
end
```

5.2.1. Codes For All Models Used for First Tests.

5.2.1.1. MATLAB Code Wavelete Scattering and Neural Network Using Adam Optimizer

```
N = length(t(,1));
Fs = 1/Ts;
tic;
sn = waveletScattering('SignalLength',N,'SamplingFrequency',Fs,...
   'InvarianceScale',20);
[~,numpaths] = paths(sn);
Ncfs = numCoefficients(sn);
sum(numpaths)
scTrain=gpuArray([ ]);
for n=1size(Ptrain,2)
   feat=featureMatrix(sn,gather(Ptrain(,n)));
   u=feat(,12end,);
   %Define the Kalman filter parameters
A = eye(3); % state transition matrix
H = eye(3); % observation matrix
Q = 1e-5*eye(3); % state noise covariance
R = 1e-3*eye(3); % observation noise covariance
x0 = zeros(3,1); % initial state
P0 = eye(3); % initial state covariance
% Smooth each signal using the Kalman filter
for i = 1size(u, 1)/3
```

```
    % Extract the signal
    signal = u((i-1)*3+1i*3,);

    % Initialize the Kalman filter
    x = x0;
    P = P0;

    % Smooth the signal
    for j = 1size(signal, 2)
        y = signal(,j);

        % Predict
        x_ = A*x;
        P_ = A*P*A' + Q;

        % Update
        K = P_*H'/(H*P_*H' + R);
        x = x_ + K*(y - H*x_);
        P = (eye(3) - K*H)*P_;

        % Store the smoothed signal
        signal(,j) = x;
    end

    % Replace the original signal with the smoothed signal
    u((i-1)*3+1i*3,) = signal;
end
    scTrain=cat(3,scTrain,u);
end
scTest=gpuArray([ ]);
for n=1size(Ptest,2)
    feat=featureMatrix(sn,gather(Ptest(,n)));
    u=feat(,12end,);
    %Define the Kalman filter parameters
A = eye(3); % state transition matrix
H = eye(3); % observation matrix
Q = 1e-5*eye(3); % state noise covariance
R = 1e-3*eye(3); % observation noise covariance
x0 = zeros(3,1); % initial state
P0 = eye(3); % initial state covariance
```

```
% Smooth each signal using the Kalman filter
for i = 1size(u, 1)/3
    % Extract the signal
    signal = u((i-1)*3+1i*3,);

    % Initialize the Kalman filter
    x = x0;
    P = P0;

    % Smooth the signal
    for j = 1size(signal, 2)
        y = signal(,j);

        % Predict
        x_ = A*x;
        P_ = A*P*A' + Q;

        % Update
        K = P_*H'/(H*P_*H' + R);
        x = x_ + K*(y - H*x_);
        P = (eye(3) - K*H)*P_;

        % Store the smoothed signal
        signal(,j) = x;
    end

    % Replace the original signal with the smoothed signal
    u((i-1)*3+1i*3,) = signal;
end
    scTest=cat(3,scTest,u);
end
TrainFeatures = scTrain;
TrainFeatures = squeeze(num2cell(TrainFeatures,[1 2]));
YTrain = Ttrain;
TestFeatures = scTest;
TestFeatures = squeeze(num2cell(TestFeatures,[1 2]));
YTest = Ttest;
[inputSize, ~] = size(TrainFeatures{1});
numHiddenUnits = 1500;
numClasses = numel(unique(YTrain));
```

```
layers = [ ...
    sequenceInputLayer(inputSize,'Normalization','zscore')
    lstmLayer(numHiddenUnits,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
maxEpochs = 170;
miniBatchSize = 170;
options = trainingOptions('adam', ...
    'InitialLearnRate', 1e-4, ...
    'MaxEpochs', 170, ...
    'MiniBatchSize', 170, ...
    'SequenceLength', 'longest', ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', {TestFeatures, YTest}, ...
    'ValidationFrequency', 30, ...
    'Verbose', 1, ...
    'Plots', 'training-progress', ...
    'ExecutionEnvironment', 'gpu');
numModels = 3; % number of models to train
YPredAll = zeros(length(YTest), numModels); % initialize matrix to store predictions
YTrain=categorical(YTrain);
for i = 1numModels
    % train a new model
    net = trainNetwork(TrainFeatures,YTrain,layers,options);

    % predict on test set
    YPred = classify(net,TestFeatures);
    YPred = renamecats(YPred,{'1','2','3','4','5'},catnames);

    % store predictions in YPredAll matrix
    YPredAll(, i) = grp2idx(YPred);
end
% take the mode prediction across all models
YPredEnsemble = mode(YPredAll, 2);
YPredEnsemble = categorical(YPredEnsemble, 15, catnames);
 YTest = renamecats(YTest,{'1','2','3','4','5'},catnames);
% compute accuracy and display confusion matrix
accuracy = 100*sum(YPredEnsemble == YTest') / numel(YTest)
% Plot the confusion matrix for the final model
```

```
figure;
cm = confusionchart(YTest, YPred);
title('Classification of faulty signals based on Levels');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
elapsedTime = toc;
fprintf('Elapsed time %.2f seconds\n', elapsedTime);
```

5.2.1.2. MATLAB Code For Wavelet Scattering and SVM with PCA

```
N = length(t(,1));
Fs = 1/Ts;
tic;
% Set wavelet scattering parameters
sn =
waveletScattering('SignalLength',N,'SamplingFrequency',Fs,'InvarianceScale',20);
[~,numpaths] = paths(sn);
Ncfs = numCoefficients(sn);
% Extract scattering coefficients for training set
scTrain = gpuArray([]);
for n = 1size(Ptrain,2)
   feat = featureMatrix(sn,gather(Ptrain(,n)));
   u = feat;
   scTrain = cat(3,scTrain,u);
end
% Extract scattering coefficients for test set
scTest = gpuArray([]);
for n = 1size(Ptest,2)
   feat = featureMatrix(sn,gather(Ptest(,n)));
   u = feat;
   scTest = cat(3,scTest,u);
end
% Prepare the data for training and testing
TrainFeatures = scTrain;
TrainFeatures = reshape(TrainFeatures, [], size(TrainFeatures, 3));
TrainFeatures = TrainFeatures';
YTrain = Ttrain;
TestFeatures = scTest;
TestFeatures = reshape(TestFeatures, [], size(TestFeatures, 3));
TestFeatures = TestFeatures';
YTest = Ttest;
```

```
% Perform PCA to reduce the dimensionality of the feature matrix
[coeff, score, latent] = pca(TrainFeatures);
% Determine the number of principal components to use
total_var = sum(latent);
var_explained = cumsum(latent) / total_var;
% Choose the number of principal components that explain at least 95% of the
variance
num_components = find(var_explained >= 0.95, 1);
% Project the training and testing features onto the selected principal components
TrainFeatures = TrainFeatures * coeff(, 1num_components);
TestFeatures = TestFeatures * coeff(, 1num_components);
% Set the number of folds for cross-validation
numFolds = 5;
% Define a cross-validation partition
cvp = cvpartition(YTrain, 'KFold', numFolds);
% Create a cell array to store the accuracy of each fold
accuracyArray = cell(numFolds, 1);
% Loop over each fold
for fold = 1numFolds

    % Get the training and validation indices for this fold
    trainIdx = cvp.training(fold);
    testIdx = cvp.test(fold);

    % Extract the features and labels for this fold
    FoldTrainFeatures = TrainFeatures(trainIdx, );
    FoldTrainLabels = YTrain(trainIdx);
    FoldTestFeatures = TrainFeatures(testIdx, );
    FoldTestLabels = YTrain(testIdx);

    % Train the SVM model on the training data for this fold

model = fitcecoc(FoldTrainFeatures, FoldTrainLabels);

    % Predict the labels of the test data for this fold
    YPred = predict(model, FoldTestFeatures);

    % Calculate the accuracy for this fold and store it in the accuracy array
    accuracyArray{fold} = 100 * sum(YPred == FoldTestLabels') /
numel(FoldTestLabels);
```

```
end
% Calculate the mean accuracy over all folds
meanAccuracy = mean(cell2mat(accuracyArray));
% Display the mean accuracy
fprintf('Mean cross-validation accuracy %.2f%%\n', meanAccuracy);
% Train the final model on all of the training data
finalModel = fitcecoc(TrainFeatures, YTrain);
% Predict the labels of the test data using the final model
YPred = predict(finalModel, TestFeatures);
% Calculate the accuracy of the final model
finalAccuracy = 100 * sum(YPred == YTest') / numel(YTest);
% Display the final accuracy
fprintf('Final test set accuracy %.2f%%\n', finalAccuracy);
YPred = categorical(YPred, {'1','2','3','4','5'}, catnames);
 YTest = renamecats(YTest,{'1','2','3','4','5'},catnames);
% Plot the confusion matrix for the final model
figure;
cm = confusionchart(YTest, YPred);
title('Classification of faulty signals based on Levels');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
elapsedTime = toc;
fprintf('Elapsed time %.2f seconds\n', elapsedTime);
```

5.2.1.3. MATLAB Code for WaveletScattering and SVM TUNED LDA

```
N = length(t(,1));
Fs = 1/Ts;
tic;
% Set wavelet scattering parameters
sn =
waveletScattering('SignalLength',N,'SamplingFrequency',Fs,'InvarianceScale',20);
% Extract scattering coefficients for training set
scTrain = [];
for n = 1size(Ptrain,2)
   feat = featureMatrix(sn, Ptrain(,n));
   scTrain = cat(3, scTrain, feat);
end
% Extract scattering coefficients for test set
scTest = [];
for n = 1size(Ptest,2)
```

```
    feat = featureMatrix(sn, Ptest(,n));
    scTest = cat(3, scTest, feat);
end
% Prepare the data for training and testing
TrainFeatures = scTrain;
TrainFeatures = reshape(TrainFeatures, [], size(TrainFeatures, 3));
TrainFeatures = TrainFeatures';
YTrain = Ttrain;
TestFeatures = scTest;
TestFeatures = reshape(TestFeatures, [], size(TestFeatures, 3));
TestFeatures = TestFeatures';
YTest = Ttest;
% Apply Linear Discriminant Analysis (LDA) on the training data for this fold
    ldaModel = fitcdiscr(TrainFeatures, YTrain);

    % Apply LDA to the training and test data for this fold
    TrainFeaturesLDA = predict(ldaModel, TrainFeatures);
    TestFeaturesLDA = predict(ldaModel, TestFeatures);

    % Convert LDA-transformed features to a numeric matrix
    TrainFeaturesLDA = double(TrainFeaturesLDA);
    TestFeaturesLDA = double(TestFeaturesLDA);
% Define hyperparameters to tune
hyperparameters = struct();
hyperparameters.BoxConstraint = {'0.01', '0.1', '1', '10'};
hyperparameters.KernelScale = {'0.01', '0.1', '1', '10'};
% Define the search space for each hyperparameter
params = struct();
params.BoxConstraint = optimizableVariable('BoxConstraint', [1, 10], 'Type',
'integer');
hyperparameters.KernelScale = [0.1, 1];
params.KernelScale = optimizableVariable('KernelScale',
hyperparameters.KernelScale, 'Type', 'real');
% Define the optimization options
optimizationOptions = struct();
optimizationOptions.MaxObjectiveEvaluations = 100;
optimizationOptions.AcquisitionFunctionName = 'expected-improvement-plus';
optimizationOptions.UseParallel = true;
% Train and tune the SVM model using the training data
temp = templateSVM('Standardize',true);
```

```
SVMModel = fitcecoc(TrainFeaturesLDA, YTrain, 'Learners', temp,
'FitPosterior',true,...
   'OptimizeHyperparameters', {'BoxConstraint','KernelScale'},...
   'HyperparameterOptimizationOptions', optimizationOptions, 'Verbose', 1,...
   'HyperparameterOptimizationOptions', struct('AcquisitionFunctionName',...
   'expected-improvement-plus', 'MaxObjectiveEvaluations', 20, 'UseParallel',true),
'Options', statset('UseParallel',true));
% Predict the labels of the test data using the tuned model
YPred = predict(SVMModel, TestFeaturesLDA);
% Calculate the accuracy of the model
accuracy = 100 * sum(YPred == YTest') / numel(YTest);
% Display the accuracy
fprintf('Test set accuracy %.2f%%\n', accuracy);
figure;
YPred = categorical(YPred, {'1','2','3','4','5'}, catnames);
 YTest = renamecats(YTest, {'1','2','3','4','5'},catnames);
  % Plot the confusion matrix in a new figure

     figure;
cm = confusionchart(YTest, YPred);
title('Classification of faulty signals based on Levels');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
   elapsedTime = toc;
   fprintf('Elapsed time %.2f seconds\n', elapsedTime);
```

5.3. MATLAB Code to Generate Road Type and Signal Condition Classification Comprehensive Testing (Three Scenarios)

```
% Load the data
Sivewave = importdata('Sinewave.tab', '\t').data;
Roughness = importdata('Roughness.tab', '\t').data;
Pothole = importdata('Pothole.tab', '\t').data;
Bump = importdata('bump.tab', '\t').data;
% Extract the time vector and acceleration data
t = Sivewave(,1);
accsv = Sivewave(,17);
accrg = Roughness(,17);
accpthl = Pothole(,17);
accbmp = Bump(,17);
% Define the cell array of workspace names
```

```
workspaceNames = {'accsv', 'accrg', 'accpthl','accbmp'};
% Loop through the workspace names
for i = 1length(workspaceNames)
    % Get the data for this workspace
    data = evalin('base', workspaceNames{i});

    % Loop through the time points and frequency indices for 1000
    for m = 1length(t)
        for n = 1399
            % Compute the new value based on the time and frequency
            if t(m) == 0 % Modified condition
                data(m,n) = 0;
            elseif data(m,1) > 0
                data(m,n+1) = ((data(m,1)*t(m)^2) + (n*2/798000)) / (t(m)^2);
            elseif data(m,1) < 0
                data(m,n+1) = ((data(m,1)*t(m)^2) - (n*2/798000)) / (t(m)^2);
            end
        end
    end

    % Loop through the time points and frequency indices for 100
    for m = 1length(t)
        for n = 1400
            % Compute the new value based on the time and frequency
            if t(m) == 0 % Modified condition
                data(m,n+400) = 0;
            elseif data(m,1) > 0
                data(m,n+400) = ((data(m,1)*t(m)^2) + (n*2/400)) / (t(m)^2);
            elseif data(m,1) < 0
                data(m,n+400) = ((data(m,1)*t(m)^2) - (n*2/400)) / (t(m)^2);
            end
        end
    end

    % Assign the modified data back to the workspace
    assignin('base', workspaceNames{i}, data);
end
% Delete the first three rows from each dataset
accsv(13,) = [];
accrg(13,) = [];
```

```
accpthl(13,) = [];
accbmp(13,) = [];
t(13,)=[];
x=ones(1,400);
x2=repmat(2,1,400);
accsv1=([x,x2]);
accsv=[accsv1;accsv];
x=repmat(3,1,400);
x2=repmat(4,1,400);
accrg1=([x,x2]);
accrg=[accrg1;accrg];
x=repmat(5,1,400);
x2=repmat(6,1,400);
accpthl1=([x,x2]);
accpthl=[accpthl1;accpthl];
x=repmat(7,1,400);
x2=repmat(8,1,400);
accbmp1=([x,x2]);
accbmp=[accbmp1;accbmp];
AccData=[accsv, accrg, accpthl,accbmp];
%{
noiser = 30 * randn(size(AccData(2end,))); % Gaussian noise with standard deviation
0.1
noisy_signal = AccData(2end,) + noiser;
AccData= [[accsv1, accrg1, accpthl1,accbmp1];noisy_signal];
% Perform wavelet denoising on each column of the matrix
denoisedMatrix = zeros(size(noisy_signal));
for col = 1size(noisy_signal, 2)
   signal = noisy_signal(, col);

   % Perform wavelet denoising on the individual signal using wdenoise
   denoisedSignal = wdenoise(signal);

   % Store the denoised signal in the denoised matrix
   denoisedMatrix(, col) = denoisedSignal;
end
% Perform wavelet denoising
AccData = [[accsv1, accrg1, accpthl1,accbmp1];denoisedMatrix];
%}
%AccData=accsv;
```

5.4. Second Labeling Code For Road Type and Signal Condition Classification Comprehensive Testing

```
Ts = mean(diff(t));
rng default % Fix the random seed
AccData = AccData(, randperm(size(AccData, 2)));
% Shuffle the columns
traindata = AccData(2end, );
trainlabel = categorical(AccData(1, ));
numClasses = numel(trainlabel);
CT = countlabels(trainlabel);
tbl = table2array(CT(, 2));
% Use cvpartition to split data into training and validation sets
c = cvpartition(trainlabel, 'HoldOut', 0.2);
trainIdx = training(c);
testIdx = test(c);
Ptrain = traindata(, trainIdx);
Ttrain = trainlabel(trainIdx);
Ptest = traindata(, testIdx);
Ttest = trainlabel(testIdx);
uniqueLabels = unique(trainlabel);
catnames = {'Normal Sinewave Road Obstacle Signals','Faulty Sinewave Road
Obstacle Signals','Normal Roughness Road Obstacle Signals','Faulty Roughness Road
Obstacle Signals','Normal Pothole Road Obstacle Signals','Faulty Pothole Road
Obstacle Signals','Normal Bump Road Obstacle Signals','Faulty Bump Road Obstacle
Signals'};
uniqueLabels = renamecats(uniqueLabels,{'1','2','3','4','5','6','7','8'},catnames);
bar(uniqueLabels, tbl);
CTtrain = countlabels(Ttrain);
CTtest = countlabels(Ttest);
tblTrain = table2array(CTtrain(, 2));
tblTest = table2array(CTtest(, 2));
H = bar(uniqueLabels, [tblTrain, tblTest], 'stacked');
legend(H, ["Training Set", "Test Set"], 'Location', 'NorthEastOutside')
LPtrain = [double(Ttrain); Ptrain];
[ii, jj, kk] = unique(LPtrain(1, ));
m = accumarray(kk, (1numel(kk)), [], @(x) {x'});
out = cell2mat(cellfun(@(x) LPtrain(, x(randperm(numel(x), 1))), m',
'UniformOutput', false));
%%idx = randperm(size(Ptrain,2),4);
```

```
%Fs = 1/Ts;
parfor n = 1numel(out(1,))
   x=out(2end,n);
   subplot(8,2,n);
   plot(t,x);
   if n == 7 || n == 8
      xlabel('Seconds');
   end
   d=dictionary([1,2,3,4,5,6,7,8],catnames);
   title(d(n));
end
```

5.4.1. Codes For All Models Used for Second Tests.

This test has the same codes as the previous test, with only differences in labeling and data. Therefore, you can replace, for example, the below

```
code% Convert labels to categorical and rename categories
YPred = categorical(YPred, {'1', '2', '3', '4', '5'}, catnames);
YTest = renamecats(YTest, {'1', '2', '3', '4', '5'}, catnames);
% Plot the confusion matrix
figure;
cm = confusionchart(YTest, YPred);
title('Classification of faulty signals based on Levels');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
to
YPred = categorical(YPred, {'1','2','3','4','5','6','7','8'}, catnames);
 YTest = renamecats(YTest,{'1','2','3','4','5','6','7','8'},catnames);
% Plot the confusion matrix for the final model
figure;
cm = confusionchart(YTest, YPred);
title('Classification for a faulty and healthy signal for each road type');
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
```

so the title and the number of mentioned label are differ.