A DEEP LEARNING SOLUTION FOR FAULT DETECTION AND DIAGNOSIS APPLIED TO INTERNAL COMBUSTION ENGINES

A Deep Learning Solution for Fault Detection and Diagnosis Applied to Internal Combustion Engines

By

Christian Brice Tongkoua Bangmi, BEng (Hons)

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

Master's of Applied Science

McMaster University

Master of Applied Science — McMaster University

(2024)

(Mechanical Engineering) — Hamilton, Ontario, Canada

TITLE: A Deep Learning Solution for Fault Detection and Diagnosis Applied to Internal Combustion Engines

AUTHOR: Christian Brice Tongkoua Bangmi

BEng (Hons), Mechanical Engineering

Coventry University, United Kingdom

SUPERVISOR: Professor Saeid Habibi

NUMBER OF PAGES: xiv, 174

This work is dedicated to my beloved mother, my dear sisters, and to the loving memory of my late father.

Abstract

In today's competitive manufacturing environment, special attention is given to the quality and reliability of manufactured products. Condition monitoring and more precisely Fault Detection and Diagnosis (FDD) are aimed at addressing that attention for increased customer satisfaction. The economic implications of FDD are highly valued in the industry, and academia is leveraged to provide smart responses. The focus of this research is the development of an FDD algorithm for internal combustion engine faults via engine block vibration using deep learning. The FDD solution would have to be implemented in software where it could operate in the absence of human intervention. The proposed solution includes two elements namely: input feature construction and fault classification.

Short-time Fourier Transform (STFT) and Convolutional Neural Networks (CNNs) perform the aforementioned elements. The FDD solution detects and diagnoses fault signatures from 4 different knock sensors mounted on a V8-type Ford engine. The solution comprises the STFT which converts the knock sensors' signal from the time domain to the crank angle-frequency domain, hence providing features to be used for diagnosis. These features are then used as input to a CNN, which can learn the crank angle-frequency patterns found in the input data and subsequently perform classification. Transfer learning is used in the proposed solution to circumvent domain shift and improve generalization. This gives the FDD solution advantages such as high diagnosis accuracy, robustness against perturbations in data quality and no need for human intervention.

Acknowledgments

I would like to thank my supervisor Professor Saeid Habibi for his ongoing support throughout my research and in the completion of my thesis. I am grateful for the invaluable opportunity I had to work at the Center for Mechatronics and Hybrid Technology (CMHT) at McMaster University where I had access to state-of-the-art research facilities. I met a fantastic group of colleagues, and the synergy we developed enriched both my professional and personal experience. I would like to also thank Cam Fisher for his technical support.

To my mother, your love, strength, and unwavering support have been the foundation of my achievements. To my three wonderful sisters, each of you has been a source of inspiration and joy in my life. To my late father, your wisdom and guidance have left an indelible mark on my journey.

Notation and abbreviations

Abbreviations

ICE	Internal Combustion Engine
FDD	Fault Detection and Diagnosis
FT	Fourier Transform
STFT	Short Time Fourier Transform
PCA	Principal Component Analysis
PC	Principal Components
WT	Wavelet Transform
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
WPT	Wavelet Packet Transform
CI	Compression-Ignition
SI	Spark Ignition
CMHT	Centre for Mechatronics and Hybrid Technologies
TDC	Top Dead Centre
NO_x	Nitrogen Oxides
CO_2	Carbon Dioxide
CO	Carbon Monoxide

EMS	Engine Management System
WVD	Wigner-Ville Distribution
CWD	Choi-Williams Distribution
SVM	Support Vector Machines
SVD	Singular Value Decomposition
SPE	Squared Prediction Error
RBC	Reconstruction Based Chart
MSPCA	Multiscale Principal Component Analysis
Mod-MSPCA	Modified Multiscale Principal Component Analysis
EMSPCA	Extended Multiscale Principal Component Analysis
GMM	Gaussian Mixed Mode
AI	Artificial Intelligence
KNN	K-Nearest Neighbour
ANN	Artificial Neural Network
DL	Deep Learning
HHT	Hilbert–Huang transform
RFE	Recursive Feature Elimination
NRS	Neighborhood Rough Set
ICS	Instantaneous Crankshaft Speed
BPNN	Back Propagation Neural Network
OLSMO	Optimized Luenberger Sliding Mode Observer

ENN	Elman Neural Network
AE	Auto Encoders
DBN	Deep Belief Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
ECU	Engine Control Unit
ECT	ECU Configuration Tool
PID	Proportional Integral Derivative
OEM	Original Equipment Manufacturer
DANN	Domain Adversarial Neural Network
GAN	Generative Adversarial Network

Notations

x	Vector x
X	Matrix X
X^t	Transpose of matrix X
$X \circ Y$	Hadamard product of matrices X and Y
$*$	Convolution product
T	Principle Components or Principle Components scores
P	PCA transformation matrix
x_{new}	New observation measurements
Σ_T	PC Covariance
Σ	SVD middle singular values matrix
V	SVD right singular values matrix
$\widehat{(\ )}$	Features subspace
$\widetilde{(\ )}$	Residual subspace
$\boldsymbol{T}^2$	Hotteling's T-squared index
Q	SPE index
φ	Combined index
ψ	Mother wavelet function
ϕ	Scaling function

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview

The condition monitoring of an industrial product is of prime importance in today's highly competitive manufacturing environment, as it is capable of discerning the product's behaviour. It is an essential part of predictive maintenance and is capable of identifying deviations from a product's operation, caused by an incipient fault. The data it generates can be used to improve product design and can be examined to correlate operating loads, life cycle and failure types. A key benefit of condition monitoring is enhanced reliability. High reliability assures a strong engineering design process, and most importantly customer loyalty. The direct consequences are a reduction in operating costs and an increase in market share and profit. However, even the best-engineered product is not perfect and malfunctions can occur over an evolutionary process, leading to failure. Hence, condition monitoring raises three questions: Has a fault occurred?', 'Where did it occur and how large?', and 'How will it progress in the future?'. These questions are answered by fault detection, fault diagnosis, and fault prognosis respectively.

In the case of complex machinery such as Internal Combustion Engines (ICEs), where many components are rotating or reciprocating, a data-driven approach offers a suitable engineering starting point to help answer the three big questions mentioned

above. To this end, vibroacoustic signals provide a fertile ground in the field of condition monitoring.

Mechanical, aerodynamic and combustion sources are the main causes of noise generation in ICEs. Noise caused by the effects of airflow perturbation can be attributed to an aerodynamic source. The turbulence phenomena occurring in the intake and exhaust ducts of an engine in addition to the turbocharger and cooling fan are good examples of aerodynamic sources. Shock between surfaces, vibrations, contacts and impacts generate noise which can be attributed to a mechanical source. Good examples are belt/chain vibration, gear vibration, and impacts between pistons and cylinders. The process governing the combustion source of noise lies in the (high) rate of cylinder pressure increase, which follows primarily the ignition delay period. Cylinder pressure frequency spectrum discontinuity and increase in the level of the high-frequency region are caused by this pressure rise. The result is engine block vibration and finally combustion noise radiation [1]. Both mechanical and combustion forces cause vibration of the engine structure, translating into the emission of noise. This interconnection between noise and vibration generation in ICEs is the reason why they can be studied under the umbrella name of vibro-acoustics. Hence vibro-acoustic signals are capable of furnishing valuable information on engine fault conditions that affect combustion, mechanics and aerodynamics as shown in Figure 1.1, which illustrates the most relevant sound and vibration sources in ICEs.

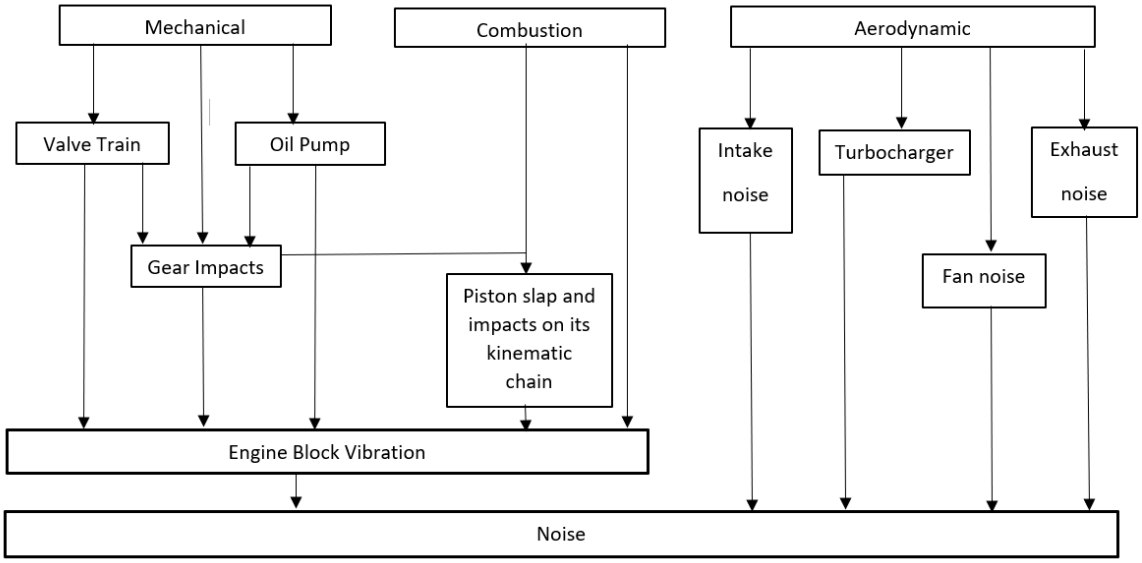


Figure 1.1 Engine noise generation

A typical way of studying faults is to consider the ICE in different sub-systems/components. The knowledge of the transmission path of the vibroacoustic phenomena, which emanate from the manifestation of these faults can be superimposed with the sub-system division of the engine to have a better picture of its state. Below is a list of typical faults and their vibroacoustic transmission path impacting the whole ICE-subsystems:

- Injection system → faults concerning injectors and the fuel pump are transmitted from the engine block to the cylinder covers.

- Ignition system → faults concerning spark plugs fall under the combustion source of noise. This noise is transmitted to the engine block as vibration and also to the manifolds.
- Pistons and Timing system → piston clearances, piston slap, clearance inside the cam-timing system (phasers, lash adjusters), slack belt/chain have a mechanical source of noise. This can be transmitted via engine block vibration and also through acoustic radiation from the chain cover and cylinder head.
- Pistons → knock and misfire fall under the combustion source of noise. This noise is transmitted to the engine block as vibration and also to the manifolds.
- Lubricating system → a fault involving the oil pump is transmitted to the oil sump by a dynamic connection between the oil and the oil sump.
- Intake and exhaust pipes → losses in the intake/exhaust manifold fall under the aerodynamic source of noise, and is transmitted by air
- Cooling system → a faulty fan falls under an aerodynamic source and is transmitted by air
- Alternator → whistle noise, interacting with the system resonance and is relevant to the aerodynamic source.
- Turbocharger → a faulty blade implies an aerodynamic source of noise and is transmitted by air. On the other hand, shaft imbalance is a mechanical source, transmitted structurally via the bearings to the compressor casing through vibration.

Engines' vibroacoustic signals are nonstationary signals, whose properties change over time. Hence one of the functions of a fault detection and diagnosis strategy is to process the signals to decipher their time-varying nature. It should also be noted that signals from rotating machinery usually exhibit cyclic characteristics. Modern engines usually have multiple sensors providing vibroacoustic information. Therefore, another function of the strategy is the utilization of multivariate statistical methods which explain the fault diagnosis as a pattern recognition problem. To conclude, the last function of the strategy is to solve the pattern recognition problem to perform classification. Artificial intelligence has proven to be a great tool to solve such problems via deep learning.

1.2 Research Motivation

The research is aimed at the development of a condition monitoring software for ICEs in the absence of human intervention. It would use only already existing engine sensors, learn from previously acquired data and would detect and localize engine faults with an accuracy higher than that of a human. In addition, the software would also be able to perform in an environment with perturbations in data quality.

1.3 Research Objective

The objective of this research is to employ deep learning in the development of a fault detection and diagnosis algorithm for internal combustion engine faults via engine block vibration. The algorithm is required to perform the following tasks:

- Use knock sensors only.
- Train using labelled data.
- Account for perturbations in data quality caused by day-to-day variations in data collection.
- Detect and localize known fault conditions.

1.4 Proposed Solution

The proposed solution satisfies the objectives via two principal steps as shown in Figure 1.2.

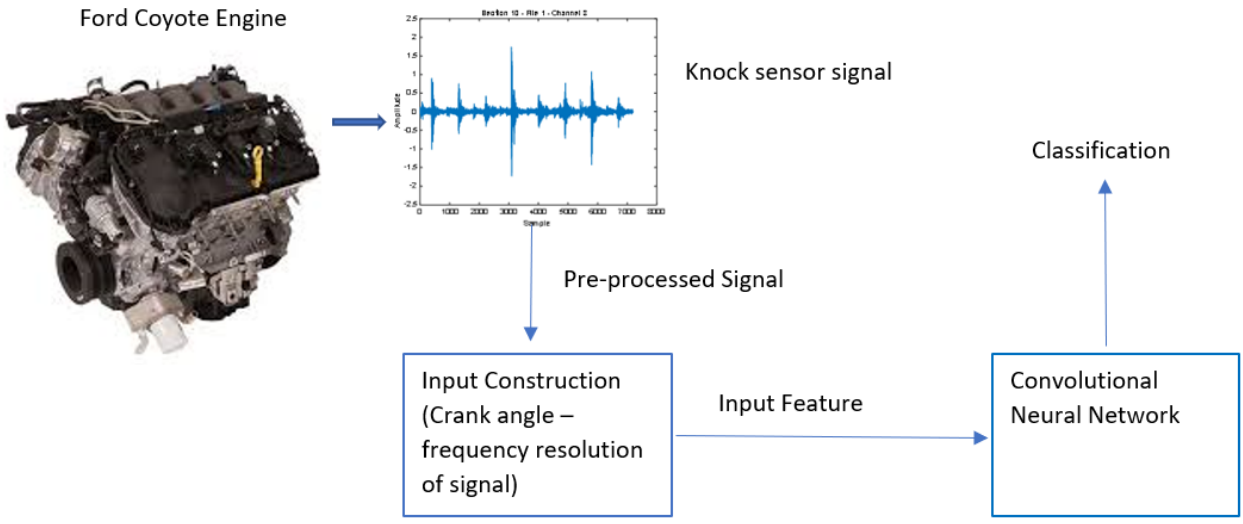


Figure 1.2 Fault Detection and Diagnosis Solution

1.5 Research Contribution and Novelty

There has been research made where knock sensors were used to perform combustion diagnosis. For example, in [2], a knock sensor was used for the diagnosis of phasing, combustion duration, and maximum pressure location. In [3], four knock sensors were used for knock detection in a 4-cylinder engine. Knock sensors in research are mostly used for knock detection, and mostly performed on straight-line engines like in [4] and [2]. The novelty of this research relies on the combination of 2 points:

- Vibration signals used for FDD come from 4 knock sensors mounted on a V-type engine, where a fault can come from any of the two banks of the engine, thereby increasing the complexity of localization by the number of cylinders on each bank
- The algorithm applied uses Short Time Fourier Transform to resolve signals in the crank angle frequency domain in combination with Convolutional Neural Network (CNN) for classification

1.6 Thesis Structure

This thesis is organized as follows: Chapter 2 presents a literature review on Fault Detection and Diagnosis (FDD) focused on internal combustion engines. It starts by covering the ICE and its mechatronic evolution up to the present day, signal processing techniques, principal component analysis and artificial intelligence. Chapter 3 covers the experimental phase of the research. It goes through the Ford

Coyote Engine specifications, the dynamometer specifications, the combustion analyzer, data logging and management and dataset construction. Chapter 4 presents the methodology that was used throughout the thesis, covering the construction of the input for classification using deep learning. Chapter 5 documents the results obtained and the discussions that follow. The conclusions of the thesis and future recommendations are provided in chapter 6.

Chapter 2: Literature Review

2.1 The Internal Combustion Engine

In the mid-1800s, the concept of heat engines using internal combustion was first developed by Nicholas Otto, Rudolph Diesel and Jean Lenoir. Since that period, internal combustion engines (ICEs) have become the primary transportation mover, and this will probably continue for decades. Fundamentally, an ICE converts chemical energy stored in fuel (mostly petroleum-based products) into thermal energy through combustion, and the subsequent expansion of the working fluid converts the thermal energy into mechanical work thanks to the crank-slider mechanism, converting linear force/work to rotational torque/work to the output shaft [5]. Discrete volumes of air and fuel are processed cyclically, with combustion occurring in a closed chamber. ICEs can be divided into two main groups namely: Compression Ignition (CI) engines which run on Diesel, and Spark Ignition (SI) engines which run mostly on Gasoline. In Canada, 90% of light-duty vehicles are powered by SI gasoline engines as of 2022 [6], while the greatest proportion of medium and heavy-duty vehicles have diesel engines.

Since the advent of ICEs, tremendous improvements in their design have been made over the years to increase performance. The global warming concerns introduced a novel paradigm in ICE design, which is emissions reduction. Modern-day engines have to perform well, be fuel efficient and achieve low emissions of greenhouse gases

such as NO_x, CO_2 and CO. These complex constraints made modern ICEs embrace the concept of mechatronic systems, involving an abundant utilization of numerous electronics, sensors and microprocessor-based control strategies, to remain competitive. Figure 2.1 shows the timeline of the adoption of engine sensors and actuators. Nowadays, engines are controlled by an Engine Management System (EMS) which regulates cylinder deactivation, idle speed, air-fuel ratio, and ignition, to reduce fuel consumption and emission levels [7]. The numerous sensors available, help the EMS in real-time condition monitoring of the engine and actuators to control the camshaft, spark plug timing, injector, throttle camshaft phasing, and cylinder deactivation. The architecture of the EMS control system is composed of various control loops (also called modules). Figure 2.2 depicts a schematic representation of a control system architecture.

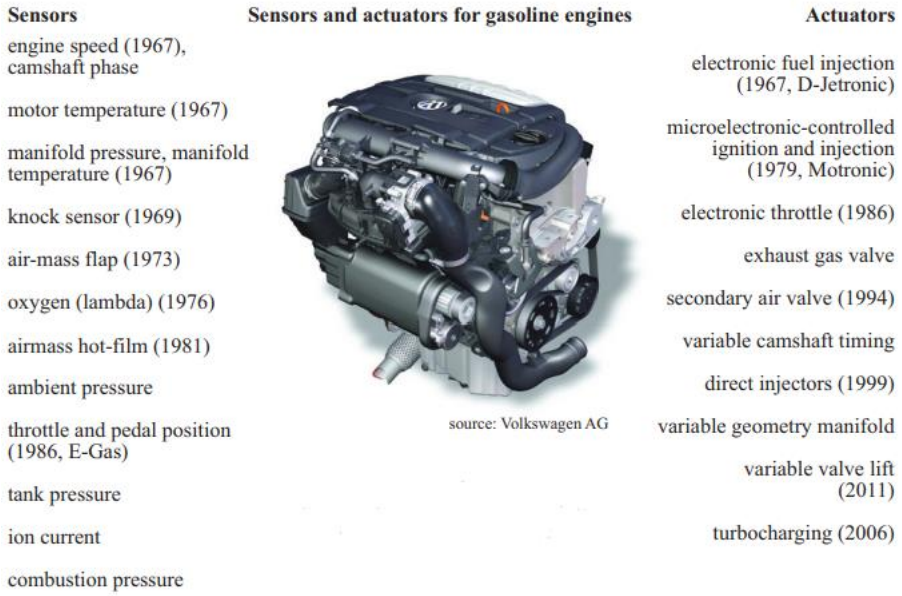


Figure 2.1 Gasoline engine sensors and actuators timeline [8]

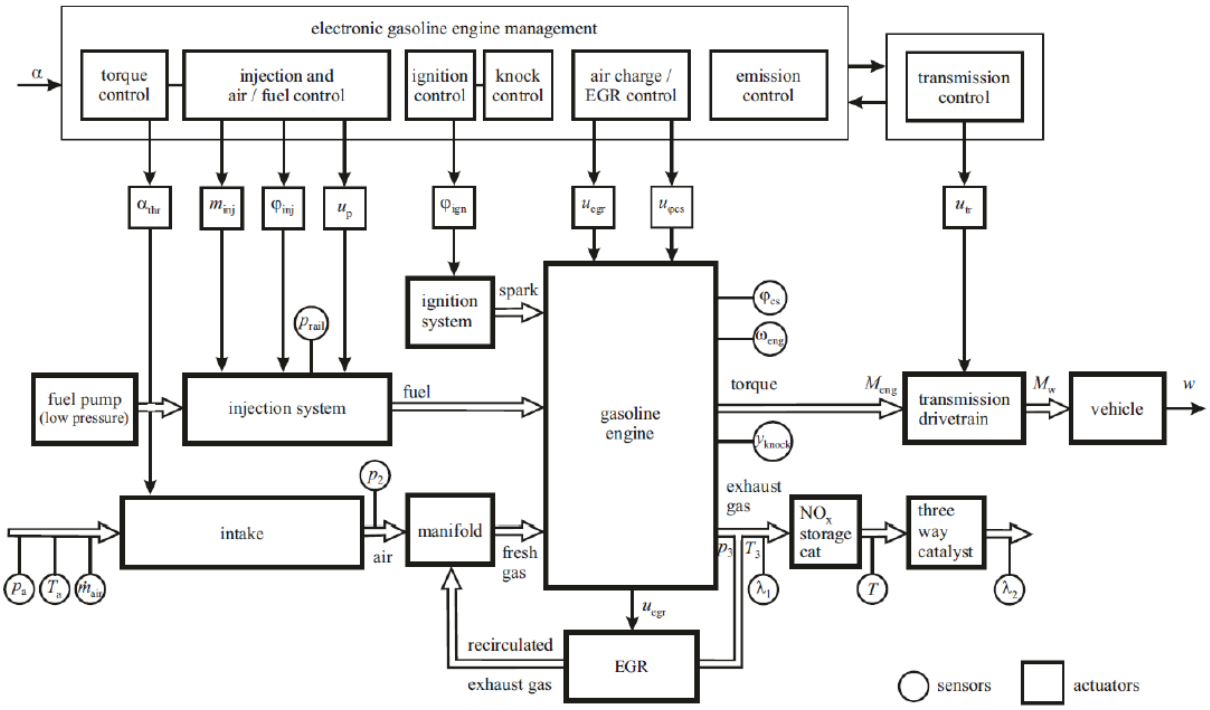


Figure 2.2 Gasoline engine control system architecture [8]

In coordination with the torque control module, some core functions found in the EMS's architecture are air-fuel ratio control, electronic throttle control, ignition timing control, knock control, idle speed control, diagnosis control, aftertreatment control, turbocharger and camshaft controls. These modules work together, in parallel to the torque control module to provide the required engine output for the torque demand. The multiple control functions are managed by a software control algorithm which is implemented by a mathematical model-based design, while non-linear feedforward control is implemented using engine maps (i.e. matrix-based lookup tables derived through extensive engine test bench operations during calibration). The

automotive market is very competitive and engine manufacturers save time and cost by employing model-based calibration, where some of the work carried out by a calibration engineer on engine dynamometers is replaced by optimization algorithms applied to an engine model.

2.1.1 Knock Sensor

Knock is the term given to the noise that is caused by the spontaneous autoignition of part of the air-fuel mixture ahead of the propagating flame front. During knock, the isolated and uncontrolled burning generates a rapid heat release in the end gas zone which induces pressure waves that propagate and interact with the flame front, causing pressure oscillations in the combustion chamber [9]. The subsequent oscillations intensify engine vibrations and increase the risks of damage which could manifest as cylinder head gasket leakage, cylinder bore scuffing, piston ring land cracking, and piston crown melting. Engine performance is also affected by knock as it prevents SI engines from reaching optimized combustion phasing and high compression ratios. On top of high-octane fuel utilization, knock can be greatly reduced through the adjustment of fuel injection and ignition timing in addition to proper calibration inside the EMS. Such corrective measures are made possible thanks to knock detection, which can be based on ion current, combustion noise, in-cylinder pressure or body vibration. Although knock detection based on in-cylinder pressure provides the highest accuracy, vibration-based detection is mostly used

nowadays. This is because pressure sensors are expensive and the high temperatures and pressures present in the cylinders reduce their lifespan. This solution is mostly suited for research purposes. On the other hand, vibration analysis provides a low-cost solution with relatively high accuracy [10]. Therefore, the vibration assessment of an engine can be captured by a knock sensor, which is a low-cost accelerometer (usually piezoelectric) as shown in Figure 2.3. However, the measured signal does not only contain the knock signature but also other vibration signatures and noises. Hence signal processing algorithms are used to extract relevant features for knock recognition and intensity evaluation. Nonetheless, the capacity of knock sensors to record the vibration signatures of an engine is highly valuable for research. With advanced signal processing algorithms, relevant features can be extracted for the detection, diagnosis and prognosis of other engine faults.

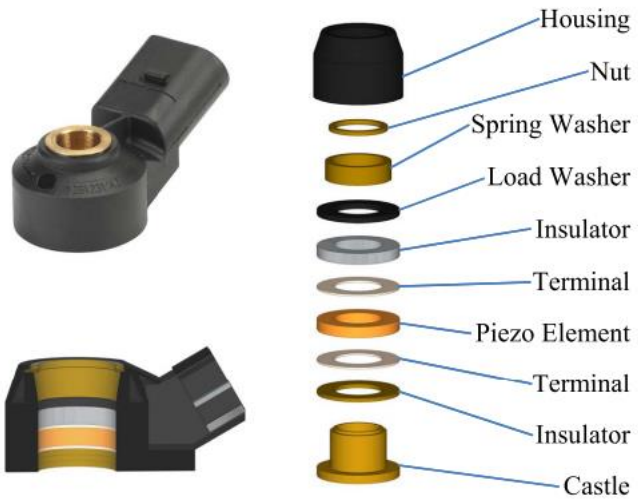


Figure 2.3 A knock sensor and its main components [11]

2.2 Data Processing Techniques

This chapter discusses common signal processing techniques used in the field of fault diagnosis and prognosis, with particular attention to the techniques applied for the analysis of internal combustion engine data. Sensor measurements are made in the time domain; and therefore, an intuitive approach is to use time-series analysis to process data. Some of these techniques are the mean, variance, standard deviation, root mean square, kurtosis, and skewness. They usually employ statistical indices to extract information from data. However, time-series analysis has a major drawback when it comes to the analysis of complex machinery with rotating and reciprocating parts. This is because, in time-series analysis, it is very difficult to decipher events in the data. The cyclic nature of engine data is more geared for their analysis in the frequency domain, where engine measurements can be decomposed into their constituent frequencies using the Fourier Transform (FT). With this technique, events can be deciphered by the occurrence of peaks that indicate their presence. The FT uses Equation 2.1 as follows:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-2\pi i \omega t} dt \qquad 2.1$$

Xia et al [12] used FT on a diesel engine to detect piston ring faults by comparing engine vibration data under normal and faulty conditions as shown in Figure 2.4. The baseline vibration band of the engine was between 2500 and 3500Hz, and the piston

ring fault induced some other frequency band from 1000 to 2500Hz. The change in the vibration spectrum was an indicator of a fault, and the shape of the spectrum was an indicator of the type of fault.

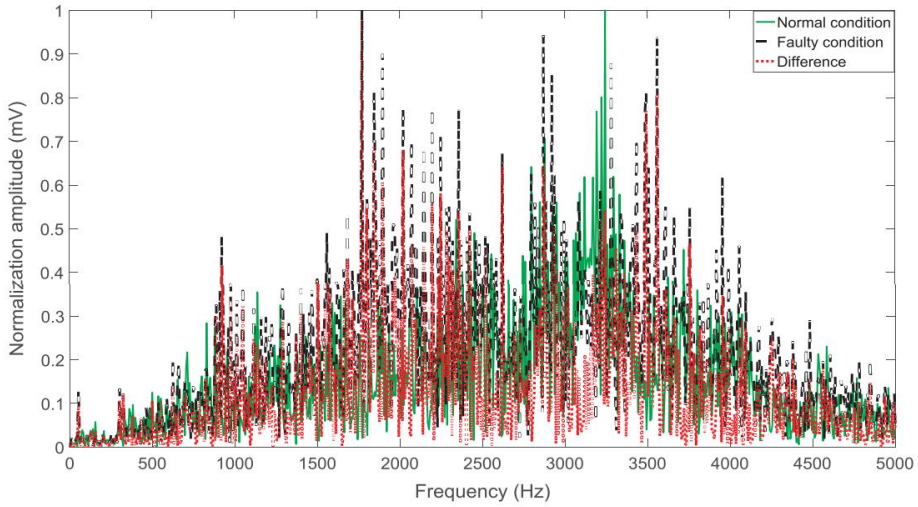


Figure 2.4 Engine frequency spectra difference under normal and faulty conditions

[12]

The FT works very well for periodic signals. In internal combustion engines, signals in the low-frequency band are mostly periodic. Such low-frequency signals are speed, torque, or pressure. Although data processing in the frequency domain provides information on the spectral dynamics of a signal, the temporal dynamics of the signal are lost. This loss of information is the major limitation of frequency domain analysis. In other words, it is not possible to determine the time at which the intrinsic frequencies of a signal occurred. The frequency domain assumes that signals are stationary (meaning signal properties do not change over time), and this assumption

does not make it a good fit for processing engine data, whose properties are non-stationary. To overcome these shortcomings time-frequency techniques are used.

2.3 Time-Frequency Techniques

Time-frequency techniques help decompose a signal into its spectral and temporal components simultaneously. This is a good fit for processing non-stationary data. Common time-frequency techniques used are the Wigner-Ville Distribution (WVD), Wavelets, Choi-Williams Distribution (CWD), and Short Time Fourier Transform (STFT). STFT is a natural extension of FT that addresses the latter's limitations. In STFT, the signal measurement is divided into time segments called windows onto which FT is applied. This enables the generation of time-localized frequency information of the signal. There are several types of window functions used for segmentation such as Gaussian or Hamming windows. The Gaussian window is suited for transient signals, while the Hamming window is suited for narrowband and random signals. Hence, a window function can perform better than another one depending on the desired output. The continuous STFT is expressed as shown in Equation 2.2.

$$X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)W(t-\tau)e^{-j\omega t}dt \qquad 2.2$$

where $W(t-\tau)$ is the window function (Hamming Type). By adjusting the value of τ, the window function can shift in time.

The discrete STFT is expressed as shown in Equation 2.3.

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x(n)\omega\,(n-m)e^{-j\omega n} \qquad 2.3$$

Modern data acquisition systems are discrete and hence, Equation 2.3 is more suited for data analysis exercises. The power of a signal can be represented in a spectrogram, which portrays the temporal and spectral energy density over the signal's lifespan as formulated in Equation 2.4.

$$spectrogram(m,\omega) = |X(m,\omega)|^2 \qquad 2.4$$

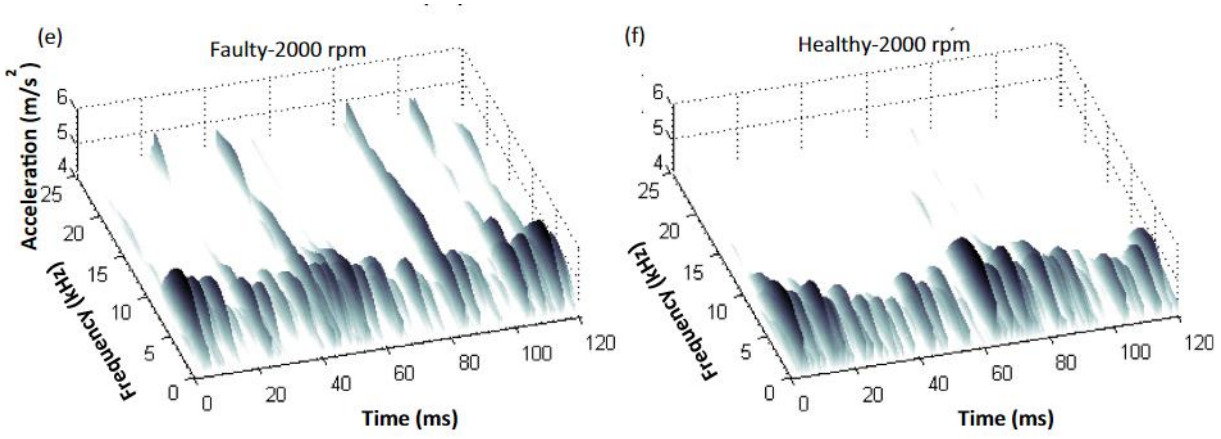


Figure 2.5 STFT of engine body vibration signal with faulty and healthy injectors

[13]

Figure 2.5 shows how STFT was used to make a comparison between a healthy (right) and faulty (left) fuel injector of a diesel engine under load conditions using vibration signals captured by three accelerometers mounted near the flywheel [13]. The STFT revealed that the dominant amplitude vibrations were between 0-10KHz for the healthy injector and above 10KHz for the faulty one. Although STFT can be a good signal processing tool, it has some limitations due to its window function, which has a fixed spectral and temporal resolution. A wide time window provides a precise frequency resolution but hinders the possibility of detecting frequency changes in transient events due to poor time localization. On the other hand, a narrow window function provides a precise temporal resolution but hinders the possibility of detecting the exact frequencies occurring at that time point due to poor frequency localization.

The effect of the window size as shown in Figure 2.7 was demonstrated by Gao et al. [14] where three different time windows (1.6, 6.4 and 25.6ms) were used to apply STFT on a non-stationary signal in Figure 2.6. The signal in Figure 2.6 is composed of impulsive signal trains organized in four clusters, each containing two transient elements of different center frequencies at 1,500 and 650 Hz, respectively. The four clusters have a 12ms time separating interval. In each cluster, the two transient elements are time-overlapped.

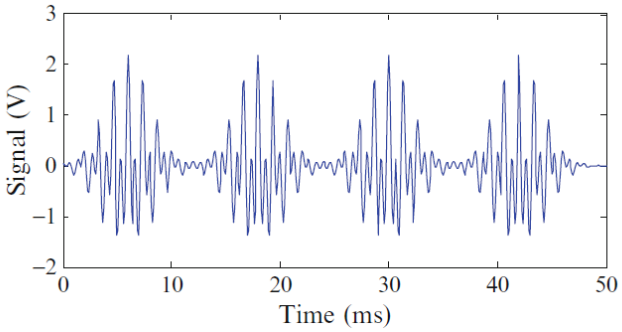


Figure 2.6 A Nonstationary Signal [14]

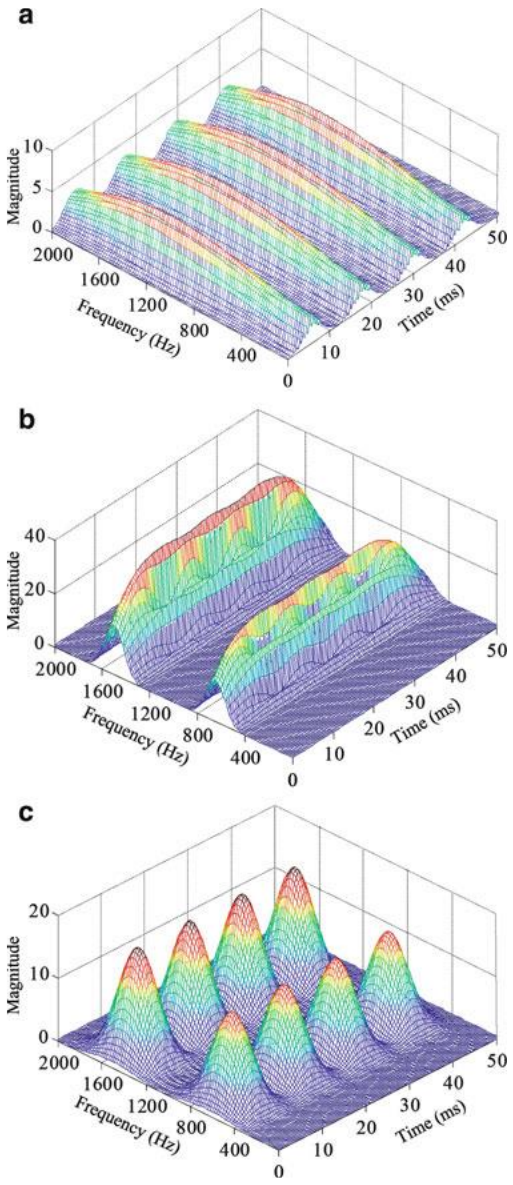


Figure 2.7 Results of the STFT of the nonstationary signal with window sizes of a) 1.6ms, b) 12ms and c) 6.4ms [14]

In Figure 2.7 a, the smallest window size of 1.6ms provided a high temporal resolution where the four signal trains were identified, while the spectral resolution

was too low to characterize the two time-overlapped transient elements found in each cluster of Figure 2.6. This resulted in the frequency elements of 1500 and 650Hz being portrayed as one lump group in the time-frequency plane. In Figure 2.7 b, the largest window of size of 25.6ms provided a high spectral resolution, where the two frequency elements were portrayed. However, the temporal resolution was too low and could not characterize the four impulse trains separated by a time interval of 12ms. Figure 2.7 c with a window size of 6.4ms provided the transient elements to be characterized in time and frequency. In reality, the given frequency contents of an experimentally measured signal are not known a priori and hence, the selection of an appropriate window size for the STFT is not guaranteed.

In fact, it is fundamentally impossible to have perfect time and frequency resolution of a signal simultaneously. This is known as the Heisenberg uncertainty principle, represented mathematically as:

$$\eta_c = \Delta_t^2 \Delta_{\omega_c}^2 \geq \frac{1}{4} \qquad 2.5$$

where η_c is the time-frequency spread of the signal, Δ_t^2 is the spread of the signal in time domain, and $\Delta_{\omega_c}^2$ is the spread of the signal in the frequency domain [15]. Having a variable window size with a high-frequency resolution at low frequencies and a high time resolution at high frequencies could circumvent SFTF limitations. This introduces multiresolution signal processing, thanks to Wavelets.

2.4 Wavelets

The concept of wavelet was introduced in 1984 by Jean Morlet, a geophysical engineer working at Elf Aquitaine who developed and implemented a technique consisting of shifting and scaling window functions in analyzing acoustic echoes used in oil prospecting. Later on, it was identified that Morlet rediscovered orthogonal transformations based on the same concept. The oldest known contribution to the concept of wavelet was by Alfred Haar in his PhD dissertation entitled "On the theory of the orthogonal function systems" in 1909 at the University of Gottingen, Germany.

Unlike STFT where the window size is fixed, wavelet transform employs a variable-size window in investigating the various frequency components within a signal. The investigation is achieved by the comparison of the signal with a set of template functions obtained from scaling (that is contraction or dilation) and shifting (that is temporal translation) of a mother wavelet $\psi(t)$, and looking for their similarities. The wavelet transform of a signal can be expressed as shown in Equation 2.6.

$$X_w(s,\tau) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} x(t) \bar{\psi} \left(\frac{t-\tau}{s} \right) dt \qquad 2.6$$

where $s > 0$ is the scaling parameter determining the spectral and temporal resolutions of the mother wavelet $\psi\left(\frac{t-\tau}{s}\right)$. τ is the shift parameter translating the scaled wavelet along the time axis, $\bar{\psi}$ is the complex conjugate of the mother wavelet

$\psi(t)$. For a mother wavelet given by $\psi(t) = e^{i2\pi f_0 t}\, e^{-(\alpha t^2/\beta^2)}$, its scaled version called daughter wavelet will be given by:

$$\psi\left(\frac{t-\tau}{s}\right) = e^{i2\pi f_0\left(\frac{t-\tau}{s}\right)}\, e^{-\alpha\frac{(t-\tau)^2}{s^2\beta^2}} \qquad 2.7$$

with f_0, α and β being constants. The spectral resolution of the mother wavelet will be expressed as $\Delta f = \sqrt{\alpha}/(s \cdot 2\pi\beta)$ while the temporal resolution will be expressed as $\Delta t = s\beta/2\sqrt{\alpha}$. From these expressions, it can be deduced that the spectral and temporal resolutions have respectively an inverse and direct relationship with the scaling parameter s.

Figure 2.8 illustrates the variation of time and frequency resolutions of the mother wavelet at two locations in the time-frequency plane: $(\tau_1, \eta/s_1)$ and $(\tau_2, \eta/s_2)$. The change in scale from s_1 at the point $(\tau_1, \eta/s_1)$ to $s_2 = 2s_1$ at the point $(\tau_2, \eta/s_2)$ divided the temporal resolution by 2 (since the width of the time window was doubled) while multiplying the spectral resolution by 2 (since the width of the window was halved). The variation of the scale s and time shift τ of a mother wavelet enables her transform to capture the intrinsic constituents of a time series through its entire spectrum by using small scales (corresponding to high frequencies) to decompose high-frequency constituents and large scales (corresponding to low frequencies) to decompose low-frequency constituents. Such a transformation is a

good match for ICE signal processing, where low-frequency signals last longer in time while high frequency signals last shorter.

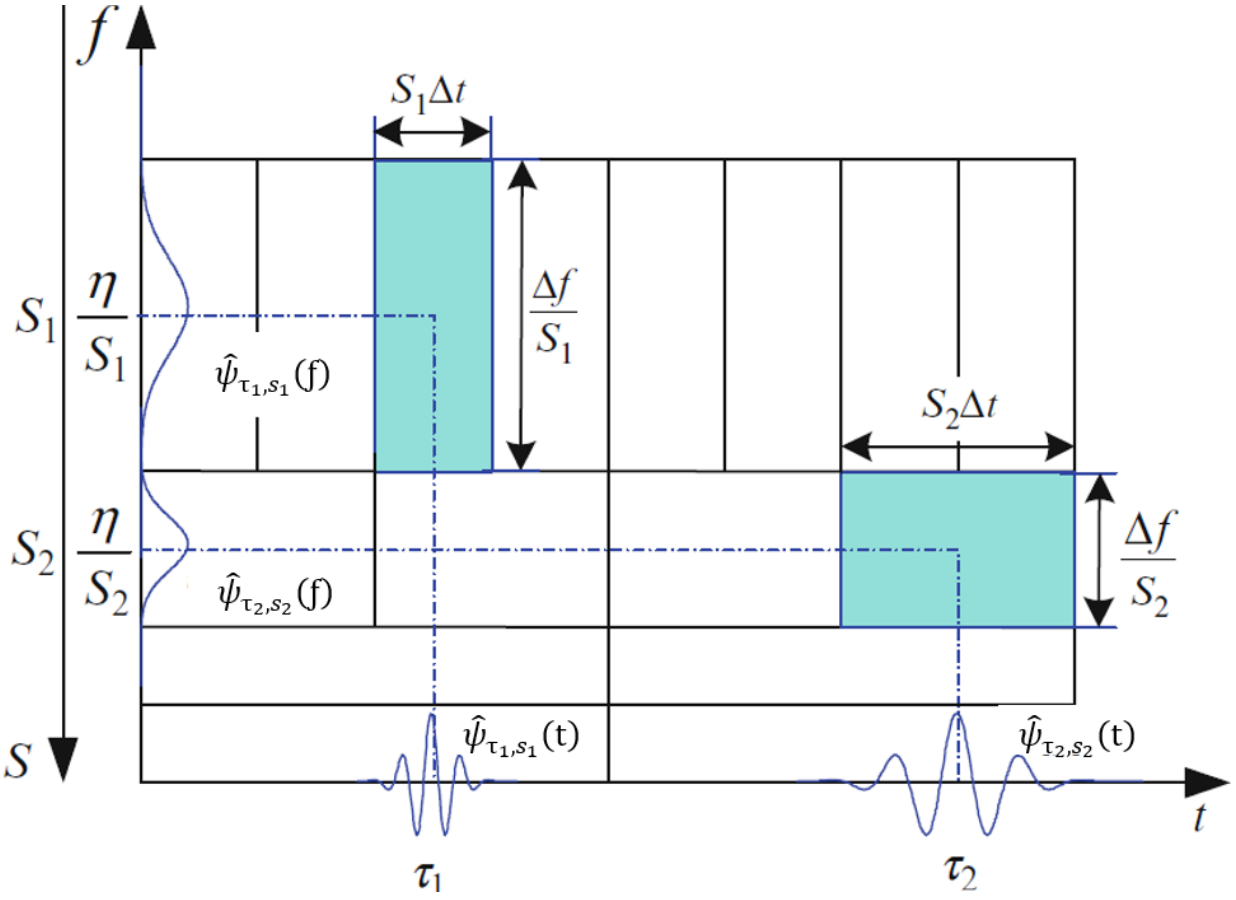


Figure 2.8 Time and Frequency resolutions of the wavelet transform [14]

The mathematical tool of wavelet transform can decompose a signal and extract "features" that characterize it. A wavelet is different from a sine or cosine wave due to certain properties which enable wavelets to have a zero average, finite energy and centred in the neighbourhood of t = 0, [16]. What qualifies as a wavelet is the admissibility condition, formulated in Equation 2.8.

$$\int_{-\infty}^{\infty} \frac{\left|\hat{\psi}(\mathrm{f})\right|^2}{(\mathrm{f})} d\mathrm{f} < \infty \qquad 2.8$$

where $\hat{\psi}(\mathrm{f})$ is the Fourier transform of the wavelet $\psi(t)$. Under the admissibility condition, the Fourier transform of the wavelet function vanishes at zero frequency which is expressed as:

$$\left|\hat{\psi}(f)\right|^2|_{f=0} = 0 \qquad 2.9$$

The zero at zero frequency property indicated in Equation 2.9 also means that the wavelet function has a zero average in time such that:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \qquad 2.10$$

Via the process of dilation/contraction and translation of the wavelet function in Equation 2.6, a series of scaled and translated daughter wavelets can be obtained in the form

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right), \qquad s > 0, \quad \tau \in R \qquad 2.11$$

The $1/\sqrt{s}$ term in Equation 2.11 is to ensure that the energy of the wavelet function is the same under various scales. If the energy of a wavelet function $\psi(t)$ is given by:

$$\varepsilon = \int_{-\infty}^{\infty} |\psi(t)|^2 dt \qquad 2.12$$

then the scaled and translated wavelet $\psi_{s,\tau}(t)$ has her energy given by:

$$\varepsilon' = \int_{-\infty}^{\infty} \left| \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) \right|^2 dt = \frac{1}{s} \int_{-\infty}^{\infty} \left| \psi\left(\frac{t}{s}\right) \right|^2 dt = \varepsilon \qquad 2.13$$

In general, a wavelet can be expressed as a Continuous Wavelet Transform (CWT) as well as in discrete form (Discrete Wavelet Transform (DWT)). The CWT of a signal x(t) is expressed as shown in Equation 2.6.

CWT can also be expressed as a convolution product such that Equation 2.6 is reformulated as:

$$X_w(s,\tau) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} x(t) \bar{\psi}\left(\frac{t-\tau}{s}\right) dt = x(t) * \psi^{\ominus}(\tau) \qquad 2.14$$

$$\psi^{\ominus}(\tau) = \frac{1}{\sqrt{s}} \bar{\psi}\left(\frac{t}{s}\right) \qquad 2.15$$

where $*$ is the convolution product. Equation 2.14 introduces the notion of filtering linked to wavelet transform. This point is illustrated by the FT of the wavelet function $\psi^{\ni}(\tau)$ such that:

$$\hat{\psi}_s^{\ni}(\omega) = \sqrt{s}\hat{\bar{\psi}}(s\omega) \qquad 2.16$$

CWT acting via a convolution process introduces the notion of the complement of a wavelet function. Consider $X_w(s,\tau)$ for $s < s_0$ is given, the reconstruction of X from its wavelet transform will require the complementary information $X_w(s,\tau)$ for $s > s_0$. The required information is obtained by a *scaling function* which is an accumulation of wavelets with scales larger than 1. The scaling function ϕ is expressed by Equation 2.17 such that:

$$\left|\hat{\phi}(\omega)\right|^2 = \int_1^{\infty} \left|\hat{\psi}(s\omega)\right|^2 \frac{ds}{s} \qquad 2.17$$

The properties of the scaling function are defined by:

$$\|\phi\| = 1 \qquad 2.18$$

$$\phi_s = \frac{1}{\sqrt{s}}\phi\left(\frac{t}{s}\right) \qquad 2.19$$

$$\hat{\phi}_s(t) = \bar{\phi}_s(-t) \qquad 2.20$$

Figure 2.9 shows a Haar's wavelet function with its scaling function and their Fourier transform.

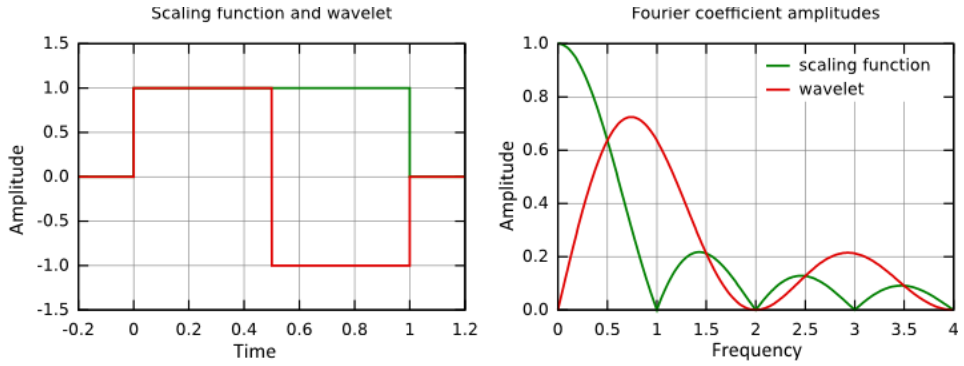


Figure 2.9 Haar wavelet functions with corresponding Fourier Transform [17]

The scale width (also called frequency band width) is conditioned by the value of s, where s can be any real value in CWT. Small incremental variations of s result in redundant information and this was the main reason for the discretization of wavelets. Discretization is fulfilled by applying octave intervals to the scale such that $s = 2^j$, j being an integer. The discretized wavelet and scaling functions are respectively expressed in Equations 2.21 and 2.22.

$$\psi_j(n) = \frac{1}{\sqrt{2}}\psi\left(\frac{-n}{2^j}\right) \qquad 2.21$$

$$\phi_j(n) = \frac{1}{\sqrt{2}}\phi\left(\frac{-n}{2^j}\right) \qquad 2.22$$

As CWT operates by convolution and filtering, the computation of discrete wavelength coefficients can be made using a family of digital filters as stated by Mallat [16]. Consider h_j and g_j respectively as the high pass and low pass filter impulse responses representing the wavelet and scaling functions at scale j. The coefficients obtained for high pass filtering are called detailed coefficients while those obtained for low pass filtering are called approximation coefficients. Let c and d be respectively the approximation and detailed coefficients, such that the transform is in the form:

$$c_{j+1}[n] = c_j * g_j[n] \qquad 2.23$$

$$d_{j+1}[n] = c_j * h_j[n] \qquad 2.24$$

From equations 2.23 and 2.24, the coefficients at scale j are used for the coefficients at scale $j + 1$. It can also be deduced from both equations that approximation coefficients are used to determine the next level coefficients; hence the original signal

can be assumed to be at an approximation coefficient of level zero such that $x[n] = c_0$. Figure 2.10 portrays the filtering process.

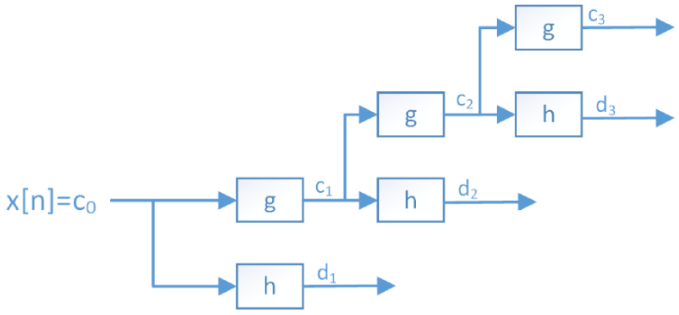


Figure 2.10 Wavelet's consecutive filtering process [17]

The inverse of the filtering operation is called the synthesis coefficient and reconnects the approximation and details coefficients. The synthesis operation is formulated in Equation 2.25.

$$c_j[n] = \frac{1}{2}\left(c_{j+1} * \tilde{g}_j[n] + d_{j+1} * \tilde{h}_j[n]\right) \qquad 2.25$$

$\tilde{h}_j[n]$ and $\tilde{g}_j[n]$ are respectively the high and low pass impulse responses of the synthesis filters. Figure 2.11 portrays the synthesis operation.

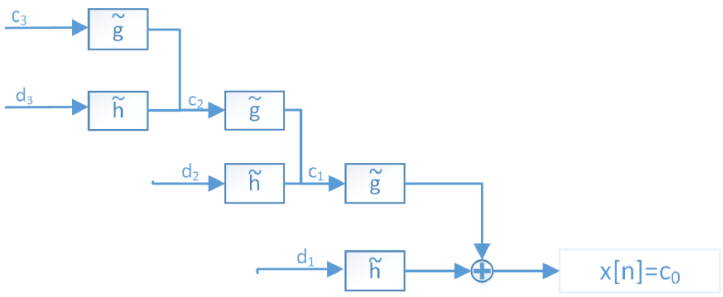


Figure 2.11 Signal synthesis of wavelet coefficient [17]

From the illustrations in Figure 2.10 and Figure 2.11, it can be easily deduced that each approximation coefficient (which is the result of the low pass filter output) is broken down at each level into two frequency bands via the low and high pass filters h and g. This multilayered filtering operation reduces the filtered output's frequency content by half. Hence the signal's down-sampling does not lose any information as a synthesis operation can be driven from the down-sampled coefficients. DWT is completed with down-sampling, which is of great use to reduce computer memory storage during calculations. DWT down-sampling is expressed in Equation 2.26 and Equation 2.27.

$$c_{j+1}[n] = c_j * g_j[2n] \qquad 2.26$$

$$d_{j+1}[n] = c_j * h_j[2n] \qquad 2.27$$

The DWT filtering of a signal is concretely illustrated in Figure 2.12 (**Note that** A is the approximate information, D is the detailed information, H is the low-pass filter and G is the high-pass filter).

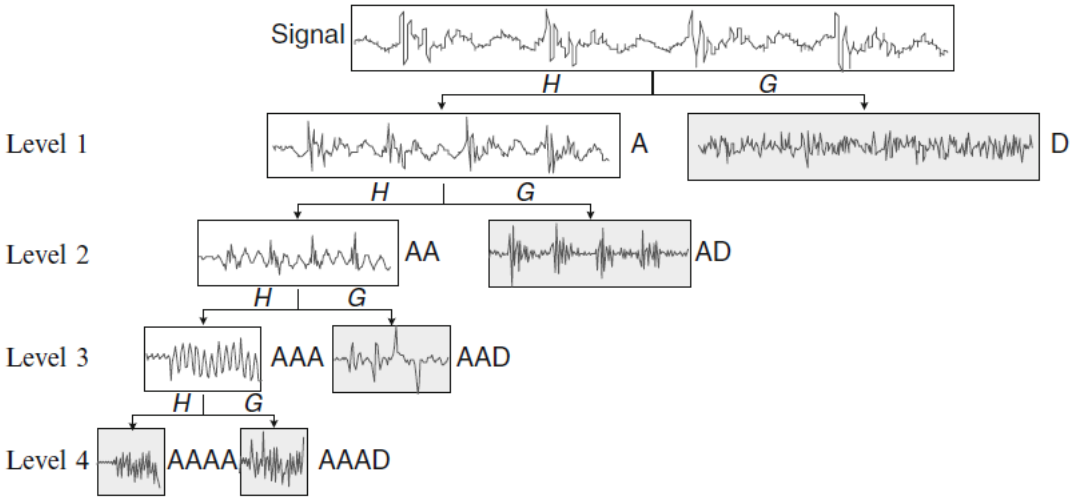


Figure 2.12 Four-level signal decomposition procedure using Discrete Wavelet Transform [14]

The filtering of a signal in DWT can be done in parallel, instead of using the down-sampling expressions in Equations 2.26 and 2.27, thereby reducing the computation time of multi-level coefficients. This is achieved by multiplying the signal $x[n]$ by the DWT matrix W_j containing the H and G filter matrices as shown in Equation 2.28.

$$W_J = \begin{pmatrix} G \\ GH \\ . \\ . \\ GH...H_{J-1} \\ HHH...H_J \end{pmatrix} \qquad 2.28$$

Given the filter (in this case high pass) length L, the matrix G is expressed in Equation 2.29 such that:

$$\mathbf{G} = \begin{pmatrix} & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\ \cdots & g(L-1) & g(L-2) & \cdots & g(0) & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & g(L-1) & g(L-2) & \cdots & g(0) & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & g(L-1) & g(L-2) & \cdots & g(0) & \cdots \\ & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \end{pmatrix} \qquad 2.29$$

The H matrix is similar to G but will have an H filter instead. Therefore, the DWT of a signal X is given by $W_x = W_J X$ and the synthesis matrix is given by $W_s = W_j^t$ where W_j^t is the transpose of W_J.

DWT are very good at detecting transients at low frequencies but do not perform well for signals in which relevant information is found at high frequencies. This is due to the signal decomposition algorithm, where only the output of the low pass filter is

broken down at each level and thus creates a high-time localization but a low-frequency resolution. The solution to this problem led to Wavelet Packet Transform (WPT) where both low-pass and high-pass filters' output are broken down at each level. The WPT down-sampling is expressed in Equations 2.30 and 2.31.

$$W_{j+1,2k}[n] = W_{j,k} * g_j[2n] \qquad 2.30$$

$$W_{j+1,2k+1}[n] = W_{j,k} * h_j[2n] \qquad 2.31$$

In Equations 2.30 and 2.31, $W_{j,k}$ is the coefficient at level j for the daughter wavelet k. $W_{j+1,2k}[n]$ represents the approximation coefficient of the signal's decomposition, while $W_{j+1,2k+1}[n]$ represents its detailed coefficient. Even values of k indicate that the coefficients result from low pass filtration, while odd values of k indicate that the coefficients result from high pass filtration. The operation of WPT is illustrated in Figure 2.13 (**Note that** A is the approximate information, D is the detailed information, H is the low-pass filter and G is the high-pass filter).

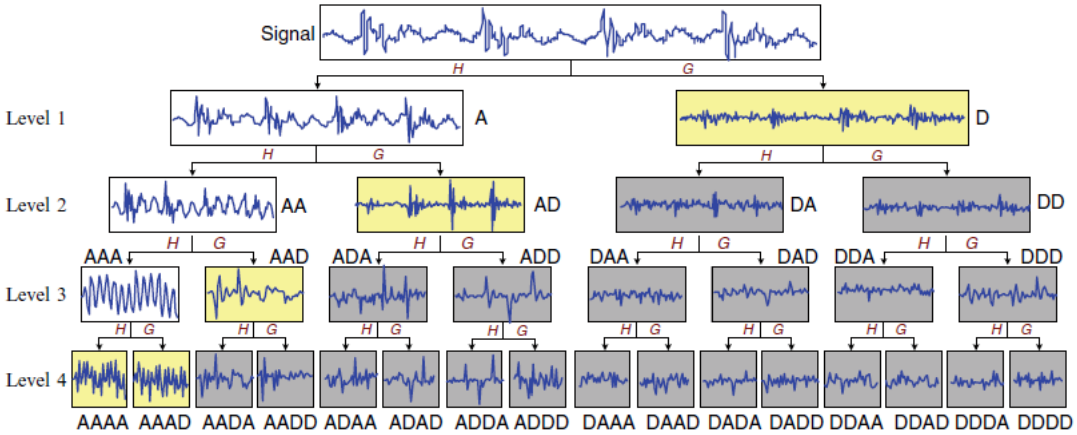


Figure 2.13 Four-level signal decomposition procedure using Wavelet Packet Transform [14]

Various types of wavelet functions with their corresponding scaling functions have been created with some examples from Figure 2.14 to Figure 2.17.

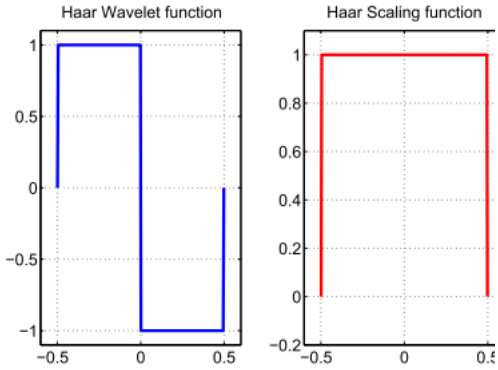


Figure 2.14 Harr [18]

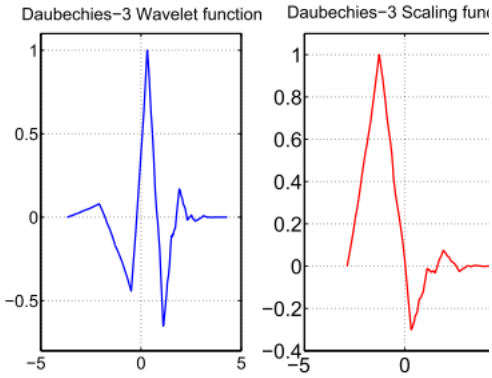


Figure 2.15 Daubechies -3 [18]

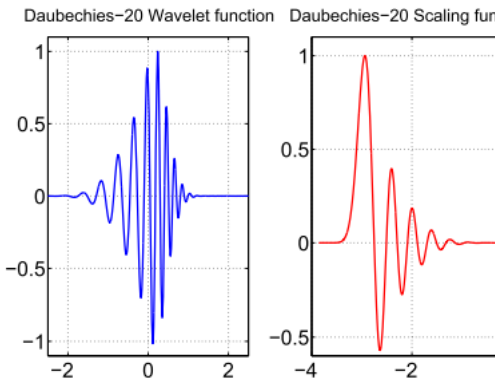


Figure 2.16 Daubechies – 20 [18]

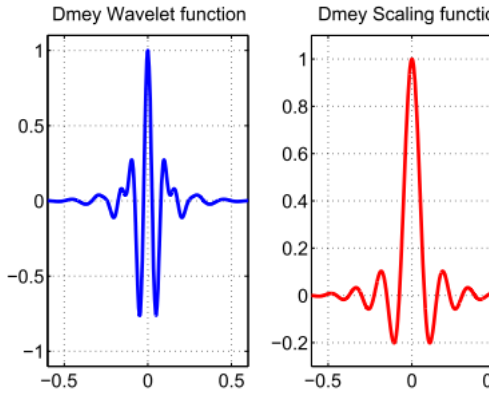


Figure 2.17 Dimey [18]

Figure 2.14 to Figure 2.17 portray the real components of the various wavelet functions; nonetheless, wavelet functions have both real and imaginary components. Figure 2.18 and Figure 2.19 portray wavelet functions with their two projections (real and imaginary) in 3 dimensions.

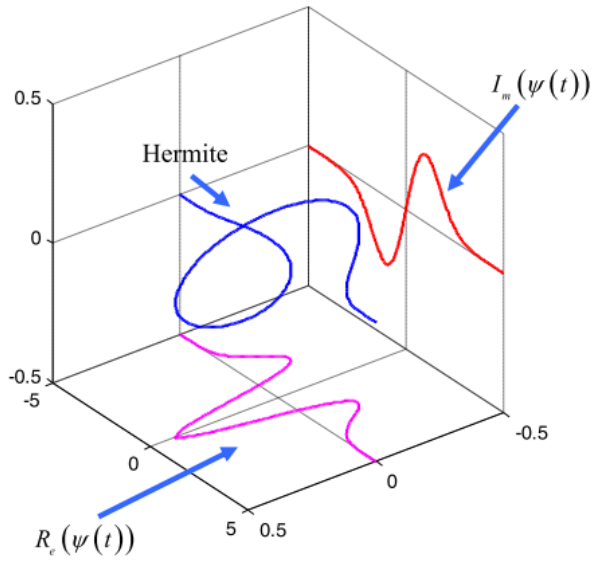


Figure 2.18 Hermine Wavelet Basis [19]

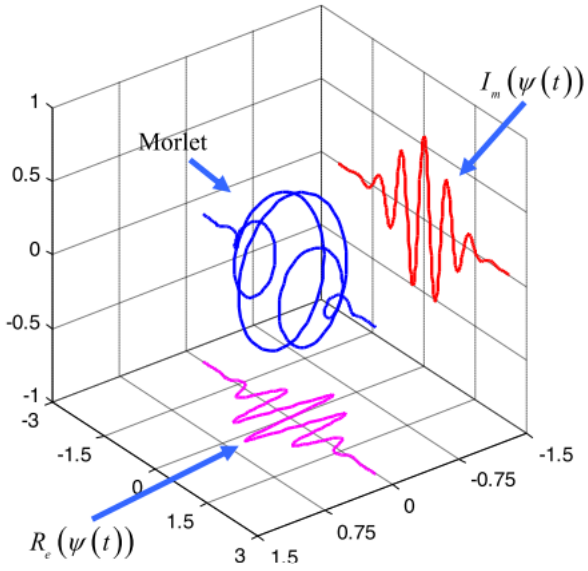


Figure 2.19 Morlet Wavelet Basis [19]

Each wavelet function and its scales form a family, with its advantages and disadvantages. The selection of a wavelet function for analysis does not follow any preestablished standards and most attempts to address this problem have been confined to specific cases. Upadhya et al. [20] developed a quantitative method for selecting mother wavelets for the detection of voltage sags based on cross-correlation sequences between the voltage signal and the mother wavelet. Megahed et al. [21] presented an algorithm for the selection of a mother wavelet for power systems fault transient grounded on the ideal reconstruction of the power signal. The metric used was the root mean square error between the original signal and the reconstructed signal. For signals buried in high levels of noise, a method tracking high kurtosis to Shannon entropy ratios for various wavelets was presented by Hemmati et al. [22] to extract features in rolling element bearing fault diagnosis. Energy and Shannon entropy criteria were used by Juhani et al. [23] for control valve leakage detection, where the winning mother wavelet would have to generate coefficients with the lowest Shannon entropy and the highest energy. When it comes to the selection of an appropriate wavelet, a general rule of thumb is to usually pick a wavelet basis which resembles the signal.

Wu et Liu [24] used WPT for fault diagnosis of an internal combustion engine. Daubechies mother wavelet was used on engine sound emission signals to extract features using Shannon entropy. Neural networks were then used for operating conditions recognition and classification of synthetic faults such as cylinder misfire,

air leakage in the intake manifold, engine coolant temperature and camshaft sensor faults. Moosavian et al. [25] used various mother wavelets for the denoising of engine vibration signals obtained from the cylinder heads for spark plug diagnosis. The denoised signals were later processed using 12 different statistical features before classification through Support Vector Machines (SVM). Ravikumar et al. [26] also used vibration signals from a tri-axial accelerometer mounted in the gearbox casing for diagnosis of an internal combustion engine gearbox. DWT based on the Haar wavelet function was used to extract features, tracking entropy change which was used as input to the K star algorithm for diagnosis.

2.5 Principal Component Analysis

Principal Component Analysis (PCA) is a distance-based ordination technique mainly used to unveil patterns in multivariate data. Its purpose is to show the relative positions of data points in fewer dimensions while keeping as much information and investigate the relationship between dependent variables [27]. PCA employs orthogonal transformation to decompose the dependent set of variables into a new set of uncorrelated independent variables onto which a hierarchical ranking is applied based on their variance. The variables with the highest variance are retained and called Principal Components (PCs) while the remaining ones are eliminated.

Considering a set of dependent measurements defined in a matrix form as $X_{(k)} = (x_1 \dots x_p)_{(k)}$ containing p variables with each variable having k measurements. In the

matrix, each column represents the number of measurements k, for a single sensor x_i. The following transformation is performed:

$$T = XP \qquad 2.32$$

where T is the principal component score matrix, and P is the principal component matrix. The principal component scores in T are the uncorrelated signals corresponding to the cross-correlated signals in X. Like in X, each column in T represents the number of measurements, k for a single variable t_i. The columns in T are arranged hierarchically starting from the highest variant component t_1 to the least variant component t_p. The PCA strategy in the field of fault detection conserves the most variant sensor measurements that contain the fault signature (useful information) and eliminates the sensor measurements that do not contain useful information.

The concept of PCA can be illustrated as shown in Figure 2.20, where the inputs are x_1 and x_2 while the t_1 and t_2 are the PCs. From Figure 2.20, it can be seen that t_1 and t_2 are orthogonal, and this condition is necessary to satisfy uncorrelation. t_1 is more variant and hence contains more information in comparison to t_2. Each component represents a different event, so a fault detection strategy employing PCA can detect faults more effectively if any of the main events change.

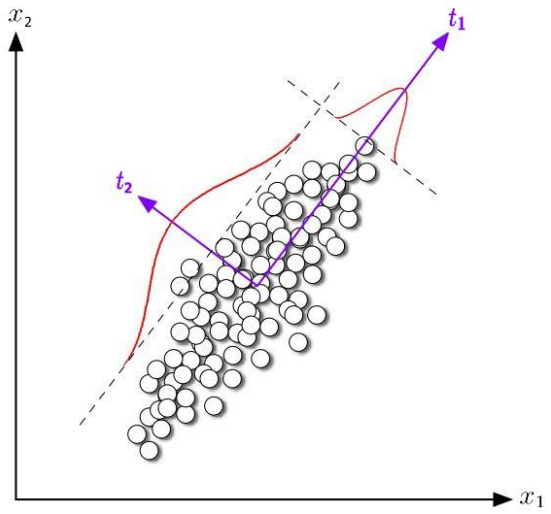


Figure 2.20 Principal Component Analysis [17]

It should be noted that matrix T is obtained from the transformation of matrix P. P should satisfy two conditions which are: the orthogonality of basis vectors (Principal Components) and the association of the principal direction with the largest variance. In a nutshell, for any matrix of measurements X, the goal is to get an orthogonal transformation of matrix P which would map X to a new matrix T which possesses a diagonal matrix. Therefore, the transformation eliminates the cross-correlation in the measurement matrix. The solution to this problem is first obtained by calculating the covariance of T as shown in [28] such that:

$$\Sigma_T = \frac{1}{n-1} T^t T$$

$$= \frac{1}{n-1} (XP)^t XP$$

$$= \frac{1}{n-1} P^t (X^t X) P \qquad 2.33$$

$$S = X^t X \qquad 2.34$$

In Equation 2.34, S is a symmetric m x m matrix. The symmetry of S makes it decomposable into its eigenvalues and eigenvectors such that:

$$S = B \Lambda B^{-1} \qquad 2.35$$

where the eigenvectors are located in B, while the eigenvalues are located in Λ, which is a diagonal matrix. B is an orthogonal matrix, meaning its inverse is equal to its transpose ($B^{-1} = B^t$). Hence,

$$\Sigma_T = \frac{1}{n-1} P^t (B \Lambda B^t) P \qquad 2.36$$

The solution to the problem is to make sure that Σ_T is diagonal and since Λ is diagonal, selecting $P = B$ solves the problem such that:

$$\Sigma_T = \frac{1}{n-1} \Lambda \qquad 2.37$$

To obtain the eigenvectors while preventing numerical errors, Singular Value Decomposition (SVD) is used. SVD is used to factor out matrices into three

components: a left singular matrix U, a right singular matrix V, and a middle singular values matrix Σ. For any matrix X, the factorization is such that:

$$X = U\Sigma V^t \qquad 2.38$$

The PCA transformation matrix P can be calculated using Equation 2.38. A better comprehension can be made if SVD is applied to $S = X^tX$ such that:

$$X^tX = V\Sigma^t\mathrm{U}^tU\Sigma V^t \qquad 2.39$$

Matrix U is unitary, meaning that its inverse is equal to its transpose. Equation 2.39 is then reformulated as:

$$X^tX = V\Sigma^t\Sigma\mathrm{V}^t \qquad 2.40$$

In SVD, the square root of eigenvectors of X^tX is given by the singular values matrix Σ in Equation 2.38. Hence Equation 2.40 is given as:

$$X^tX = V\Lambda\mathrm{V}^t \qquad 2.41$$

From the comparison of Equation 2.41 with Equations 2.34 and 2.35, it can be observed that $B = V$. Keeping in mind that the diagonality of Σ_T is thanks to the selection of $P = B$, the PCA transformation matrix P equates to the SVD right singular matrix V. SVD ensures the hierarchical ranking of eigenvalues and their

corresponding eigenvectors to enable the PCs to follow the same order from the most variant (t_1) to the least variant (t_p).

PCA is very useful when it comes to putting the spotlight on patterns in multivariate data in addition to the relationship between its dependent variables. The inverse of Equation 2.32 is given by:

$$\mathrm{X} = \mathrm{TP}^t \qquad 2.42$$

Re-expressing X in a more extensive form is given by:

$$\begin{aligned} \mathrm{X} &= \hat{X} + \tilde{X} \\ &= \hat{T}\hat{P} + \tilde{T}\tilde{P} \\ &= [\hat{T} \quad \tilde{T}][\hat{P} \quad \tilde{P}]^t \\ &= \mathrm{TP}^t \end{aligned} \qquad 2.43$$

T and P matrices are each split into two subspaces illustrated by a hat ($\widehat{\ }$) and tilde ($\widetilde{\ }$) sign. The hierarchical ranking of PCs in PCA indicates that the first subspace which is illustrated by the hat has the high varying PCs while the second subspace illustrated by the tilde has the residuals. The split of data into two subspaces enables PCA to exhibit fault detection capacities. The dominant features of a monitored system are represented by the hat, while the tilde represents the residuals such as noise and errors. For any dataset with m PCs, n components can be extrapolated to

the feature subspace and $m - n$ PCs to the residual subspace. There are different rules used to select the appropriate number of PCs in the literature. However, three rules stand out from the crowd namely: the Heuristic rule, the Kaiser rule and the manual determination rule. In the Heuristic rule, the PCs representing 95% of all PCs' variance are extrapolated to the feature subspace while the rest are deferred to the residual. In the Kaiser rule any PC with a variance greater than the average variance of all PCs is extrapolated to the feature subspace. In the last method, the extrapolation of the PCs to the subspaces is left to the end user.

The completing part of PCA fault detection is the translation of the information found in the PCs to a detection algorithm. Joe Qin and Haqshenas [29], [30] pinpointed three indices namely: Hotteling's $\boldsymbol{T}^2$, Squared Prediction Error (SPE) $\mathcal{Q}$, and Combined index φ. These indices operate by a comparison of the system's measurements under test with that of the system under healthy operation (baseline measurements). Hotteling's $\boldsymbol{T}^2$ measures PC's variation between new and baseline measurements due to its close relation to F distribution. Hotteling's $\boldsymbol{T}^2$ displays a weighted extrapolation of the new measurements on the baseline's feature subspace. Anomalies in new measurements that preserve the covariance relationship between variables in the baseline measurements are detectable using this index. In SPE, the extrapolation of new measurements is made on the baseline's residual subspace instead. Anomalies in new measurements that violate the covariance relationship between variables in the baseline measurements are detectable using this index. The

combined index φ merges $\boldsymbol{\mathcal{T}}^2$ and $\mathcal{Q}$ within it, and enables any anomaly in new measurements preserving the covariance relationship between variables or not in the baseline measurements to be detectable. Hotteling's $\boldsymbol{\mathcal{T}}^2$ is mathematically expressed as shown in Equation 2.44.

$$\boldsymbol{\mathcal{T}}^2 = x_{new}\hat{P}\hat{\Lambda}^{-1}\hat{P}^t x_{new}^t \qquad 2.44$$

$$= x_{new}\hat{D}x_{new}^t \qquad 2.45$$

where $x_{new}$is the new set of measurements, and $\hat{D}$ is the $\mathcal{T}^2$extrapolating matrix, which is formulated as shown in Equation 2.46.

$$\hat{D} = \hat{P}\hat{\Lambda}^{-1}\hat{P}^t \qquad 2.46$$

$$\hat{\Lambda} = \frac{1}{n-1}\hat{T}\hat{T}^t \qquad 2.47$$

where n is the measurement size. SPE $\mathcal{Q}$ is mathematically formulated as shown in Equation 2.48:

$$\mathcal{Q} = \|\tilde{x}_{new}\|^2 = \left\|x_{new}\tilde{C}\right\|^2 \qquad 2.48$$

$$\tilde{C} = \tilde{P}\tilde{P}^t \qquad 2.49$$

C is the $\mathcal{Q}$ extrapolating matrix. Hence, $\tilde{x}$ is the extrapolation of the measurement vector x to the residual subspace using $\tilde{C}$ such that $\tilde{x} = x\tilde{P}\tilde{P}^t$.

The Combined index φ is mathematically expressed as shown in Equation 2.50.

$$\varphi = \frac{\mathcal{Q}}{\delta^2} + \frac{\boldsymbol{\mathcal{T}}^2}{\mathcal{T}^2} \qquad 2.50$$

$$= x_{new} \Phi x_{new}^t \qquad 2.51$$

where $\delta^2$and $\mathcal{T}^2$ are respectively the $\mathcal{Q}$ and $\boldsymbol{\mathcal{T}}^2$ upper limits [31] and $\boldsymbol{\Phi}$ is the φ extrapolation matrix defined as:

$$\boldsymbol{\Phi} = \frac{\tilde{P}\tilde{P}^t}{\delta^2} + \frac{\hat{P}\hat{\Lambda}^{-1}\hat{P}^t}{\mathcal{T}^2} \qquad 2.52$$

The indices in $\boldsymbol{\Phi}$ are calculated and compared against their respective thresholds or upper limits such that any violation of the threshold will imply the presence of a fault in the system. Weighted chi-square distribution was shown by Box et al. [32] to be a good approximation of quadratic forms like $\mathcal{Q}$ and hence expressed in Equation 2.53 as:

$$\delta_{\mathcal{Q}} = \left(\frac{v}{2m}\right)\chi^2\left(\frac{2m^2}{v}\right) \qquad 2.53$$

where the sample mean and variance are represented respectively by m and v. The term $\frac{v}{2m}$ in Equation 2.53 represents the weight of the chi-squared distribution χ^2 while $\frac{2m^2}{v}$ represents the chi-squared degrees of freedom.

Jackson [33] was able to formulate the upper limit of Hotteling $\boldsymbol{\mathcal{T}}^2$ thanks to the fact that $\boldsymbol{\mathcal{T}}^2$ follows F distribution. Hence the upper limit is given by:

$$\mathcal{T}^2 = \frac{l(n-l)}{n-l} F(l.n-l,\alpha) \qquad 2.54$$

where the sample size and the number of variables are represented by n and l respectively. The required confidence level is represented by α. Equation 2.54 is appropriate when there is a high sample size n. Tracy et al. [34] formulated the upper limit for smaller sample sizes, given by:

$$\mathcal{T}^2 = \frac{l(n^2-l)}{n(n-l)} F(l.n-l,\alpha) \qquad 2.55$$

On the other hand, Yue and Qin [31] were able to formulate the upper limit of the combined index φ thanks to the fact that φ can be approximated by a chi-squared distribution. Hence the upper limit is defined as:

$$\zeta_\alpha^2 = g_\varphi \chi_\alpha^2 h_\varphi \qquad 2.56$$

From Equation 2.56, g_φ represents the weight of the chi-squared distribution χ^2 while h_φ represents the chi-squared degrees of freedom. The required confidence level is represented by α. g_φ and h_φ are formulated as:

$$g_\varphi = \frac{\frac{l}{\mathcal{T}_\alpha^4} + \frac{\theta_2}{\delta_\alpha^4}}{\frac{l}{\mathcal{T}_\alpha^2} + \frac{\theta_1}{\delta_\alpha^2}} \qquad 2.57$$

$$h_{\varphi} = \frac{\left(\frac{l}{\mathcal{T}_{\alpha}^{2}} + \frac{\theta_1}{\delta_{\alpha}^{2}}\right)^{2}}{\frac{l}{\mathcal{T}_{\alpha}^{4}} + \frac{\theta_2}{\delta_{\alpha}^{4}}} \qquad 2.58$$

where $\theta_1 = trace(\tilde{\Lambda})$ and $\theta_2 = trace(\widetilde{\Lambda^2})$ with $\tilde{\Lambda}$ defined as:

$$\tilde{\Lambda} = \frac{1}{n-1}\tilde{T}\tilde{T}^{t} \qquad 2.59$$

Another statistical index was proposed by Haqshenas [29] based also on the PCs covariance, T. Taking the assumption that a measurement signal contains the baseline measurement in addition to faulty components, then the measurement signal can be formulated as:

$$\mathrm{X} = X^{*} + F \qquad 2.60$$

where X^{*} is the baseline healthy component and F the faulty components. The PCs covariance is:

$$COV(T) = E(TT^{t}) - E(T)E(T)^{t}$$

$$= E(\mathrm{X}PP^{t}X^{t}) - E(XP)E(XF)^{t} \qquad 2.61$$

Considering Equation 2.60, Equation 2.61 becomes:

$$COV(T) = E(X^{*}PP^{t}X^{*}T) + E(FPP^{t}F^{t}) + E(X^{*}PP^{t}F^{t}) \qquad 2.62$$

$$+ E(FPP^{t}X^{*t})$$

Taking the assumption that X^* and F are independent, Equation 2.62 is reformulated as:

$$COV(T) = COV(X^*P) + COV(FP) \qquad 2.63$$

From Equation 2.63, $\Sigma_{T,healthy} = COV(X^*P)$ was defined in Equation 2.33 as the PC covariance of the baseline measurement. If $\Sigma_{T,faulty} = COV(T)$, Equation 2.63 becomes:

$$\Sigma_{T,faulty} = \Sigma_{T,healthy} + COV(FP) \qquad 2.64$$

Hence,

$$COV(F) = P(\Sigma_{T,faulty} - \Sigma_{T,healthy})P^t \qquad 2.65$$

Equation 2.65 represents the fault covariance extrapolation into the feature subspace. The PCA model is constructed based on the healthy (or baseline) measurements, where the PC covariance matrix is diagonalized. Hence, $\Sigma_{T,healthy}$ is a diagonal matrix in Equation 2.65. On the other hand, $\Sigma_{T,faulty}$ is not diagonal due to the fault effects as its PCs were found using the baseline PCA model. In addition, the faulty condition dictates the diagonal elements found in the $\Sigma_{T,faulty}$ matrix. Therefore, the presence of a fault in a monitored system is represented and characterized by the $(\Sigma_{T,faulty} - \Sigma_{T,healthy})$ term. Consequently, Haqshenas [29] defined the following index:

$$S_c = \sum_{i=1}^{m} \frac{\sum_{i=1}^{m} (\Sigma_{T,faulty} - \Sigma_{T,healthy}) \circ (\Sigma_{T,faulty} - \Sigma_{T,healthy})}{\lambda_i} \qquad 2.66$$

where λ_i is the baseline PC variance. Due to the symmetric nature of the covariance matrix in Equation 2.66, the summation can be performed in any order (that is row-wise then column-wise or vice versa). The multiplication symbol ∘ in Equation 2.66 represents the Hadamard (entry-wise) multiplication. S_c is compared against a threshold, such that if it is higher than the latter, the monitored system is stated as faulty. The advantage of Haqshenas' index is that it produces a single output used for comparison with the threshold in contrast to other indices. In a nutshell, statistical indices help indicate the presence of a fault in a system but are not able to do fault diagnosis.

Fault diagnosis is the process of sourcing a fault and its severity through its symptoms, applying knowledge and analyzing results. Accurate fault diagnosis starts after detection, where the required features are extracted for efficient fault classification or identification. Identification and subsequent curative actions can increase productivity and reduce the cost of maintenance in many industrial applications.

One of the most used methods in fault diagnosis is contribution plots, which are graphical representations used to pinpoint which variable, sensor, or feature has the most impact on a fault. Alcala and Qin [35] defined three different plots for $\boldsymbol{T}^2$, Q,

and φ indices. An example of a contribution plot is given in Figure 2.21 where the contribution of each variable (in this case four) is shown in each sample. From the example, it can be deduced that the variable in blue has the highest contribution to the fault.

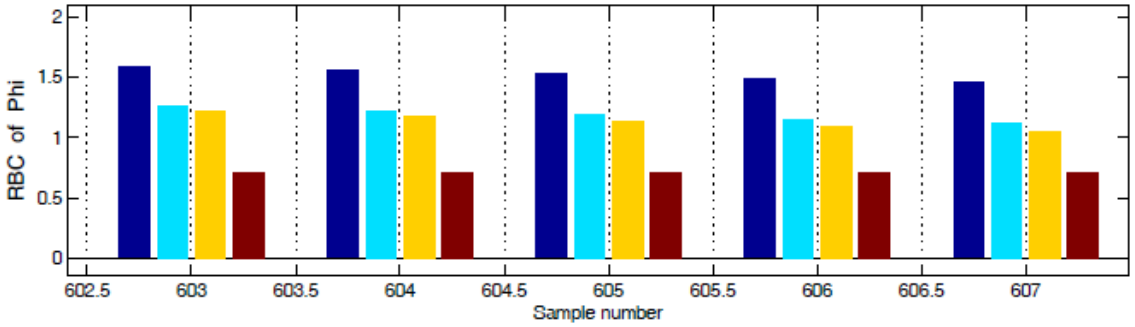


Figure 2.21 RBC chart example [17]

Each of the three indices has its corresponding plot defined such that:

$$c_{i,T^2} = (x_{new}\hat{D}^{0.5}\xi_i)^2 \qquad 2.67$$

$$c_{i,Q} = (x_{new}\tilde{C}\xi_i)^2 \qquad 2.68$$

$$c_{i,\varphi} = (x_{new}\Phi^{0.5}\xi_i)^2 \qquad 2.69$$

where x_{new} is the new set of measurements and $c_{i,(\,)}$ is the contribution of the i^{th} variable. ξ_i is a standard basis, whose size equates to that of the number of variables. It is a zero vector where only the i^{th} element equates to 1 such that:

$$\xi_i = [0\ 0\ 0\ \ldots\ 0\ 1\ 0\ \ldots\ 0\ 0]^t \qquad 2.70$$

Overall, the contribution equations extrapolate each variable to a scoring subspace. However, discrepancies in the contribution are observed when these equations are applied to healthy measurements in such a way that faults existing in low-contribution variables are not adequately represented as they would be in a scenario where the faults would exist in high-contribution variables. To make the faults existing in different variables even, normalization of the contributions is used. In order to reach this goal, Dunia and Joe Qin [36] used the following mathematical expressions:

$$RBC_{i,\,T^2} = \frac{(x_{new}\hat{D}^{0.5}\xi_i)^2}{\hat{d}_{ii}} \qquad 2.71$$

$$RBC_{i,Q} = \frac{(x_{new}\tilde{C}\xi_i)^2}{\tilde{c}_{ii}} \qquad 2.72$$

$$RBC_{i,\varphi} = \frac{(x_{new}\Phi^{0.5}\xi_i)^2}{\phi_{ii}} \qquad 2.73$$

where,

$$\hat{d}_{ii} = \xi_i^t \hat{D} \xi_i \qquad 2.74$$

$$\tilde{c}_{ii} = \xi_i^t \tilde{C} \xi_i \qquad 2.75$$

$$\phi_{ii} = \xi_i^t \Phi \xi_i \qquad 2.76$$

The index S_c in Equation 2.66 was extended by Haqshenas [29] for fault diagnosis, simplifying Equation 2.65 to:

$$COV(F) = P(\Sigma_{T,faulty} - \Sigma_{T,healthy})\, P^t$$

The fault covariance is used for isolation via a newly defined index. This is performed by finding every variable contributing to the fault in the matrix $COV(F)$ and normalizing it by its variance. This index is given by:

$$F_c = \left(\sum_{i=1}^{m} COV(F) \circ COV(F) \right) \circ \Sigma_\lambda \qquad 2.77$$

where $\Sigma_\lambda = [\frac{1}{\lambda_1} \dots \frac{1}{\lambda_n}]$ is a vector containing the inverse of PC variances λ_i. The output of F_c is a vector with a size equal to the number of variables where the value of the i^{th} element constitutes the fault contribution of the i^{th} variable. Therefore, for a detected fault, the fault variables can be isolated.

2.6 Multiscale Fault Diagnosis

The concepts of PCA and wavelet transforms were discussed in the previous sections. PCA is a strong tool used in multivariate analysis and wavelet transform is a strong time-frequency signal processing tool used for spectral and temporal decomposition of signals. The combination of wavelets and PCA creates a strong multivariate multiscale tool called Multiscale Principal Component Analysis (MSPCA) which stretches the field of fault diagnosis and is suitable for complex dynamic systems.

MSPCA applies DWT on signals followed by PCA on the resulting coefficients originating from DWT. This way, PCA resolves the correlation between sensor measurements at different spectral resolutions, scrutinizing for valuable information between signals and their frequency scales. Figure 2.22 schematizes the concept of MSPCA where g and h are the low pass and high pass filters respectively.

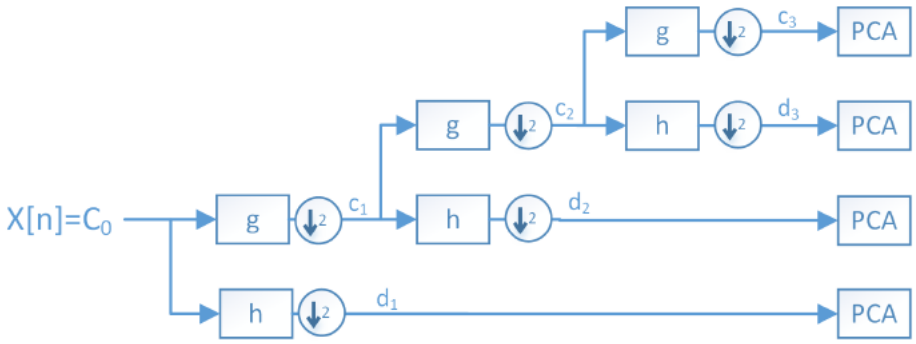


Figure 2.22 MSPCA Technique

MSCPCA was later modified by Yoon and MacGregor [37] to incorporate RBC (Reconstruction Based Charts) in the diagnosis. The DWT in the MSCPA resulted in down-sampling of the signals at each consecutive scale, creating shorter length coefficients. The Mod-MSPCA proposed up-scaling and reconstruction. Instead of feeding the wavelet coefficients directly in PCA, they are reconstructed via reconstruction filters. This conceptualizes the translation of measurement decomposition to their fragmentation into a combination of scaled-based signals. The decomposition of a data matric X at J different scales is formulated as follows:

$$W_J X = H_1 X + H_2 X + \cdots + H_J X + G_J X \qquad 2.78$$

where G and H are respectively the low and high pass filters. By applying reconstruction filters to the Equation 2.78, the reconstruction can be written as:

$$\begin{aligned} W_s W_J X &= H_1^t H_1 X + H_2^t H_2 X + \cdots + H_J^t H_J X + G_J^t G_J X \\ &= X_1 + X_2 + \cdots + X_J + X_{J+1} \end{aligned} \qquad 2.79$$

where W_s is the wavelet reconstruction (or synthesis) matrix. The newly reconstructed components ($X_1 + \cdots + X_{J+1}$) are fed into PCA in the Mod-MSPCA.

At this level of analysis, it should be noted that both Mod-MSPCA and MSCPCA use DWT where only the low pass filter is decomposed to the next level. However valuable information is lost in the high pass filter components of the signal and if the system dynamics exhibit changing patterns in high frequencies, its complete description might be lost and the subsequent FDD strategy might be less efficient. To

solve this problem, Ismail [17] replaced the DWT with WPT. Hence reformulating the reconstruction of the data matrix X such that:

$$W_s W_J X = H_1^t H_1 X + G_1^t G_1 X + H_2^t H_2 X + \cdots + H_{2^{J-1}}^t H_{2^{J-1}} X + G_{2^{J-1}}^t G_{2^{J-1}} X$$

$$= X_1 + X_2 + \cdots + X_{2^{J-1}} + X_{2^J} \qquad 2.80$$

The newly reconstructed components ($X_1 + X_2 + \cdots + X_{2^{J-1}} + X_{2^J}$) are fed into PCA allowing the FDD strategy to be performed in tighter high-frequency bands of measurements spectrum. This upgrade to the Mod-MSPCA done by Ismail [17] is called Extended Multiscale Principal Component Analysis (EMSPCA). The FDD of MSPCA, Mod-MSPCA and EMSPCA used the same statistical system monitoring concept where a comparison between the baseline (healthy) and the monitored system is performed. As such, a modification of statistical indices defined in Equations 2.44, 2.48, 2.50, 2.66 and 2.77 was needed for decomposition at any Jscale such that:

$$\boldsymbol{\mathcal{T}}_j^2 = \sum_{i=1}^{A} \frac{\mathcal{T}_i}{\lambda_{i,j}} \qquad 2.81$$

$$Q_j = \sum_{i=1}^{m} \tilde{x}_{i,j}^2 \qquad 2.82$$

$$\varphi_j = \frac{\mathcal{Q}_j}{\delta_j^2} + \frac{\boldsymbol{\mathcal{T}}_j^2}{\mathcal{T}_j^2} \qquad 2.83$$

where j = 1.2….J+1 and A is the number of PC.

The statistical indices at each scale are totalized such that:

$$\boldsymbol{\mathcal{T}}^2 = \sum_{j=1}^{J} \boldsymbol{\mathcal{T}}_j^2 \qquad 2.84$$

$$\mathcal{Q} = \sum_{j=1}^{J} \mathcal{Q}_j \qquad 2.85$$

$$\varphi = \frac{\mathcal{Q}}{\delta^2} + \frac{\boldsymbol{\mathcal{T}}^2}{\mathcal{T}^2} \qquad 2.86$$

The multiresolution version of Hasquenas index was formulated as:

$$S_{c,j} = \sum_{i=1}^{m} \frac{\sum_{i=1}^{m} (\Sigma_{T,faulty} - \Sigma_{T,healthy})_j \circ (\Sigma_{T,faulty} - \Sigma_{T,healthy})_j}{\lambda_{i,j}} \qquad 2.87$$

The fault diagnosis RBC defined in Equations 2.71, 2.72 and 2.73 then become:

$$RBC_{i,\mathcal{T}_j^2} = \frac{(x_j \hat{D}_j^{0.5} \xi_i)^2}{\hat{d}_{j,ii}} \quad 2.88$$

$$RBC_{i,\mathcal{Q}_j} = \frac{(x_j \tilde{C}_j \xi_i)^2}{\tilde{c}_{j,ii}} \quad 2.89$$

$$RBC_{i,\varphi_j} = \frac{(x_j \Phi_j^{0.5} \xi_i)^2}{\phi_{j,ii}} \quad 2.90$$

where:

$$\hat{d}_{j,ii} = \xi_i^t \hat{D}_j \xi_i \quad 2.91$$

$$\tilde{c}_{j,ii} = \xi_i^t \tilde{C}_j \xi_i \quad 2.92$$

$$\phi_{j,ii} = \xi_i^t \Phi_j \xi_i \quad 2.93$$

The revised fault diagnosis index is given by:

$$F_{c,j} = \left(\sum_{i=1}^{m} COV(F) \circ COV(F) \right)_j \circ \Sigma_{\lambda_j} \qquad 2.94$$

In Equation 2.87, $S_{c,j}$ provides a single value at each spectral decomposition level and therefore is capable of detecting a fault and providing information on the fault frequency in its corresponding level. The fault diagnosis, via $F_{c,j}$ portrays a detailed contribution map to the fault where for each spectral level, the contribution of each variable is presented. The fault type is hence characterized by the topography of the contribution map. It should be noted that both $S_{c,j}$ and $F_{c,j}$ monitor the variation between the "Healthy" and "Faulty" PCs covariance matrices. M. Ismail [17] added another step in his EMSPCA due to the vibroacoustic nature of his analyses. The vibration and sound measurements have different scales and variances and hence could result in misleading information if fed to the EMSPCA. Sound and vibration have different sensitivities to faults and hence, he added a normalization step in the signal pre-processing such that:

$$Normalized\ signal = \frac{raw\ signal - mean\ value(baseline\ centre)}{Variance(baseline\ centre)} \qquad 2.95$$

The raw signal is the monitored signal which is computed against the baseline (healthy) signal as for any signal-based algorithm, the monitored signal is compared to the baseline.

Cao et al. [38] used PCA for the fault diagnosis of cracks on a centrifugal pump's blades. The multi-source signals obtained from the pump via a variety of sensors were processed by PCA to reduce the dimensionality and increase the feature resolution of the signals. The extracted features were used as input to a Gaussian Mixed Model (GMM) for classification. In [39], PCA was used to reduce the dimensionally of the data samples obtained from a marine diesel engine fuel oil supply system, using SVM for diagnosis. Xu et al. [40] used advanced PCA techniques for the fault diagnosis of turbomachinery damage subjected to uncertainty. The proposed methodology was a combination of discrete wavelet packet transform, Bayesian hypothesis testing, and probabilistic principal component analysis. The set of sensor data recording time series signals was decomposed in various wavelet coefficients. Possible imperfections assessment in the decomposed coefficients and subsequent avoidance of over-denoising was provided by the ratio of posterior odds Bayesian approach. The Bayesian wavelet clean-up method's effectiveness was assessed using power spectral density estimated by the Welch method. Moreover, multivariate correlation and uncertainty were addressed by probabilistic principal component analysis of the multiple time series for crack damage diagnosis. Sheriff et al [41] used MSPCA for fault detection of a chemical processing plant thanks to its capacity for deterministic feature extraction and decorrelation of autocorrelated sensor measurements. The proposed algorithm is operated first by wavelet decomposition of the input data matrix followed by PCA on each of the wavelet coefficients. Then $\boldsymbol{\mathcal{T}}^2$ or $\mathcal{Q}$ threshold

limits were applied to retain the wavelet coefficients of concern at each scale. Then the data matrix was reconstructed with the retained coefficients and scales and finally, PCA was applied to the reconstructed data matrix to obtain an approximate data matrix and residuals.

2.7 Artificial Intelligence

A complete fault diagnosis and prognosis strategy consists of three main tasks mainly: determining whether the monitored system is normal or not; locating the incipient fault and its reason; and predicting the trend of fault development. Essentially, fault diagnosis and prognosis can be considered as a pattern recognition problem, in regards to condition monitoring of ICEs. Artificial intelligence (AI), considered a powerful pattern recognition tool has retained the attention of researchers and exhibits promise in rotating machinery fault recognition applications. Direct fault pattern recognition is impossible due to the variability and abundance of monitored signals' content. Most common intelligent fault diagnosis strategies have a workflow consisting of pre-processing of signals by feature extraction algorithms to modify the input pattern in such a way that they can be transformed to low dimensional feature vectors for smoother match and comparison [42]. The feature vectors are then fed as inputs to the AI techniques to achieve fault recognition. The fault recognition step sums up to mapping of the information gained in the feature space, to machinery fault in the fault space. Several AI techniques have been used in

fault diagnosis of rotating machinery, more specifically classifiers and statistical learning techniques such as K-Nearest neighbour (KNN), Bayesian classifier, Support Vector Machines (SVM) and Artificial Neural Networks (ANNs). One of the most recent techniques employed in the field of fault diagnosis is Deep Learning (DL). AI can also be categorized under the prism of the learning approach (that is supervised or unsupervised learning). In supervised learning, the training data used is already labelled. Here, the training data is used to infer a mapping function or learning algorithm from the input variable X to the output variable Y. The labels (also called desired outputs) are already known for any given pair of input-output, $M = \{\sum(X_i, Y_i)\}_i^N$ where N is the number of training examples. In unsupervised learning, the AI model uses unlabelled, unclassified and categorized learning data. The main goal of unsupervised learning is to decipher the patterns hidden in unlabelled data. Unlike supervised learning which is used for classification and/or regression, unsupervised learning is mostly used for clustering and/or self-organizing maps. The mosaic of AI techniques is shown in Figure 2.23 while the relationship between AI disciplines is shown in Figure 2.24.

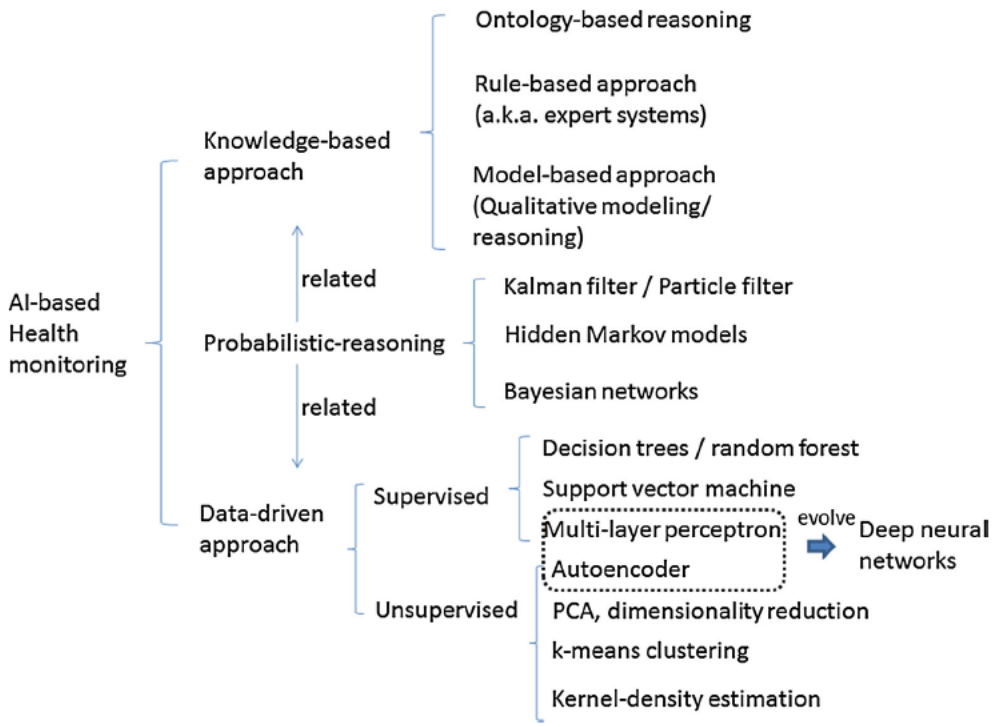


Figure 2.23 Categories of AI techniques used in Machinery Health Monitoring [43]

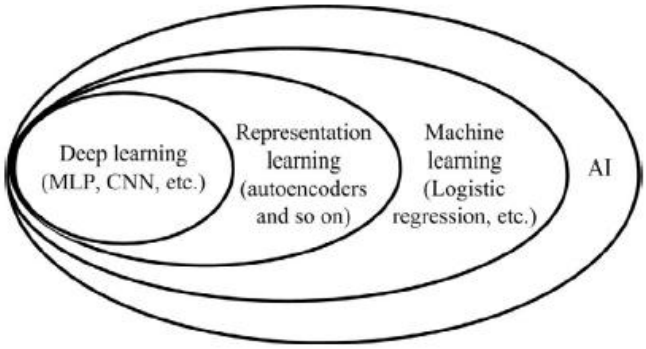


Figure 2.24 Relationship between AI disciplines [44]

2.7.1 K-Nearest Neighbour

K-Nearest Neighbour (KNN) is an instanced based supervised learning algorithm based on the postulate that instances in a dataset will mostly exist in close vicinity to other instances with the same characteristics [44]. Given any set of classified instances, $T = [(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)]$ where $x_i$is the feature vector of the unlabelled instance and y_i the labelled instance. $y_i = c_1, c_2, \dots, c_k$, i= 1, 2…N. For a given training sample (x, y), the KNN algorithm looks for the nearest K instances to x based on a certain distance metric. $N_k(x)$ is the vicinity (neighbourhood) containing these k instances. The label y of the test sample x can be calculated on the basis of decision rules such that:

$$y = arg\,\max_{c_j} \sum_{x_i \epsilon N_k(x)} I(y_i = c_J). \qquad I = 1,2, \dots, N; \qquad J = 1,2, \dots, K \tag{2.96}$$

where I is the indicator function.

When a classification label is attributed to the instances, then an unclassified instance's label can be deduced by observing the attribute of it nearest neighbours as shown in Figure 2.25

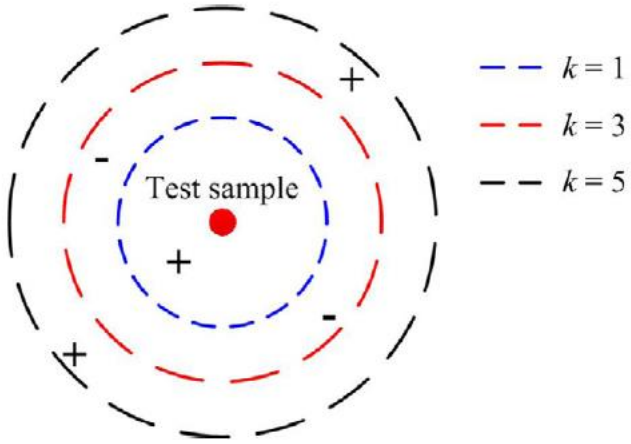


Figure 2.25 KNN Diagram [44]

Figure 2.25 shows that when K = 1 or 5, the test sample is classified as positive while when K =3, it is classified as negative. In a nutshell, the KNN algorithm possesses the following three basic elements: the number of measured instances k, the distance metric and the classification decision rule. KNN has the advantage of having a simple implementation in contrast to other AI algorithms.

Wang [45] was able to identify up to 5 levels of severity in gears cracks via KNN and the redundant statistical features built using Debauchees 44 binary WPT. Pandya et al. [46] used the Hilbert–Huang transform (HHT) for feature extraction as input to KNN in an article on bearing fault diagnosis. A comparison with other AI techniques such as Naïve Bayes, ANN and weighted KNN was performed in the paper. It should be noted that KNN faces some difficulties such as the optimal selection of the parameter k and the indifferentiable neighbourhood boundary [47]. A modified

version of KNN was used by some researchers to overcome these problems. Lei et al. [48] used a weighted KNN integrated into their fault diagnosis strategy of bearings, where the extracted features were weighted per their sensitivity to the machine's health states during the training process of the KNN model. Dong et al. [49] used particle swarm optimization to improve KNN and obtained better diagnosis results for bearings. Furthermore, Li et al. [50] used KNN in combination with PCA where the latter helped the compression of high dimensional time-frequency features into low dimensional uncorrelated eigenvectors as input to KNN for life-grade recognition of rotating machinery.

2.7.2 Naïve Bayesian Classifier

Mitchell [51] defined the Naïve Bayes methods as a supervised classification method, leveraging probabilistic computations based on Bayes' Theorem and the assumption of conditional independence. Given a training set $T = [(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)]$ labelled as y, $y_i = c_1, c_2, \dots, c_k$, i= 1, 2…N and assuming there exist S_i possible values for x^l, l= 1, 2, …, n; with K possible values for Y. Naïve Bayes starts by learning the combined probability distribution $P(X, Y)$ of the input and output by conditional probability distribution based on the assumption of conditional independence, such that:

$$P\big(X = x \big| Y = c_j\big) = P(X^{(1)} = x^{(1)}, \dots, X^{(n)} = x^{(n)} | Y = c_j)$$

$$= \prod_{l=1}^{n} P(X^{(l)} = x^{(l)}) \qquad j = 1, 2, \dots, K \tag{2.97}$$

Then, on the basis of the learnt model, the output label y showing the biggest posterior probability for the input x can be computed through Bayes' Theorem:

$$P(Y = c_j | X = x) = \frac{P(X = x | Y = c_j) P(Y = c_j)}{\sum_j P(X = x | Y = c_j) P(Y = c_j)} \tag{2.98}$$

and

$$y - \arg \max_{c_j} P(Y = c_j) \prod_l P(X^{(l)} = x^{(l)}) | \, Y = c_j) \tag{2.99}$$

Its high efficiency and ease of implementation make naïve Bayes classification a common classification method.

Seshadrinath et al. [52] used duel tree wavelet transform for feature extraction before implementing the naïve Bayes fault diagnosis model for induction machines. Cunha Palácios et al. [53] used the time domain as the analysis ground of their fault diagnosis strategy via naïve Bayes, where they used the amplitude of current signals from motor faults. Flett and Bone [54] proposed a diagnosis strategy for a diesel engine valve train and compared the results of different classifiers such as ANN, KNN and naïve Bayes classifiers with the latter having the best overall performance.

2.7.3 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning algorithm used for classification, clustering and regression. Its algorithm builds an optimal separating hyperplane $f(x) = 0$ between data sets. Given a training set $T = [(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)$ labelled as y, $y_i \epsilon \{-1, 1\}$, it solves a constrained quadratic problem based on the minimization of structural risk [55] , such that:

$$f(x) = W^T x + b = \sum_{i=1}^{N} W_i x_i + b \qquad 2.100$$

where W is a vector of dimension N and b is a scalar. The separation of the samples between the positive and negative classes necessitates the hyperplane to be subjected to:

$$y_i f(x_1) = y_i (W^T x + b) \geq 1, \quad i = 1.2. \dots, N \qquad 2.101$$

As shown in Figure 2.26, both hyperplanes H_1 and H_2 meet the constraint in Equation 2.101. However, the optimal separating hyperplane H^*, is the one which discriminates the datasets by creating the maximum distance between the plane and the nearest data; that is the maximum margin, given by $\gamma = {}^{2}/_{\|W\|}$. The optimization objective of the linear SVM is given in Equation 2.102.

$$\min_{W,b} \quad \frac{1}{2}\|W\|^2 \qquad 2.102$$

Using the Kuln-Tucher condition to convert the optimization problem into the equivalent Lagrangian dual quadratic optimization problem, the classifier based on the support vector can be obtained [44].

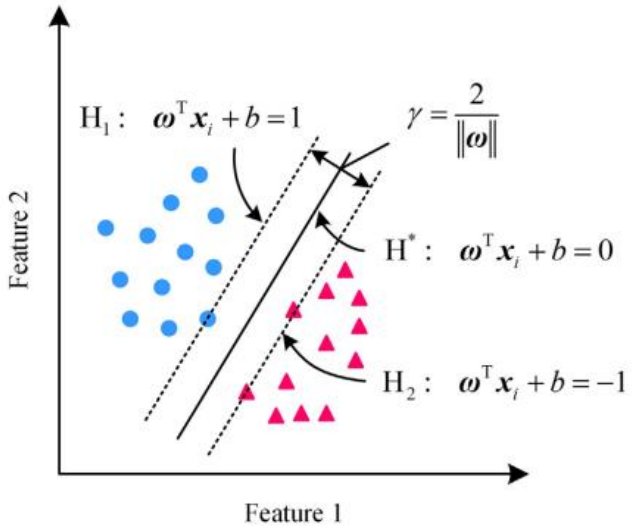


Figure 2.26 Linear SVM classification [47]

Classification problems often exhibit a linear nature as shown in Figure 2.26 due to the fact that the hyperplane used is straight. However, in more challenging classification problems the optimal solution may employ a curved hyperplane as shown in Figure 2.27.

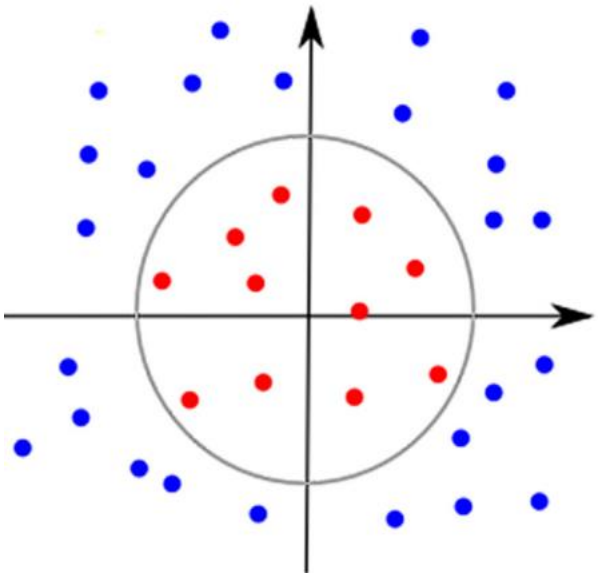


Figure 2.27 Non-linear SVM classification with curved hyperplane [56]

It should also be noted that some datatypes are nonlinearly separable, hence the classification problem is denoted as non-linear and is solved using some type of kernel method. This method typically requires the support vectors to be transformed to a higher-dimensional input space. To rephrase it, this extra step converts a nonlinearly separable set of features to a set of linearly separable ones [56]. A critical step before the training and evaluation of SVM is feature selection. The feature selection methods can be categorized into three main types namely: embedded methods, filter methods, and wrapper methods [57].

In embedded methods, the feature selection tool is integrated into the classifier and the selection is automatically performed during the SVM training phase. This is achieved by what is called the “Kernel trick”; a kernel method capable of improving

the computational efficiency of SVM in training and conveniently aiding in preventing overfitting in ill-conditioned problems where the dimensionality of monitored signals is by far exceeding the number of available examples for training. A kernel function illustrates pairwise analogous measures between all example patterns summarized in a kernel matrix of dimension N x N, where N is the number of observations. Rather than depending on the raw feature vector as direct input to the SVM classifier, the kernel trick permits the SVM to be trained using the kernel matrix, hence mapping the raw measurements to a higher dimensional feature space for both linear and non-linear problems as shown in Figure 2.28.

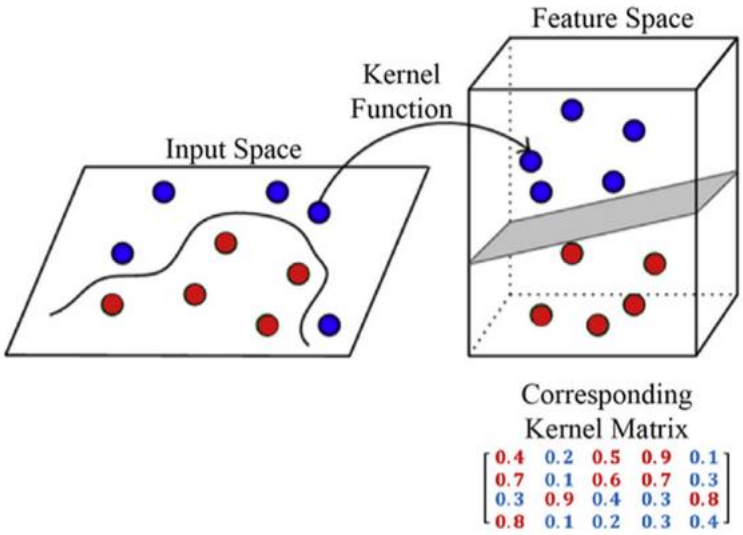


Figure 2.28 Conceptual portray of the 'Kernel Trick' [56]

Filter methods on the other hand perform dimensionality reduction of features prior to the classification and perform relevance measures on the training set, hence

removing the least important features before inserting a hyperplane. The feature reduction helps in:

- Redundancy reduction in raw measurement for greater sample training data proportion relative to the dimensionality of the features
- Interpretation facilitation of the final classifier by pinpointing the data having the most predictive information relevant to differentiating classes
- Training process acceleration and computational load reduction

An example of feature reduction involves the removal of near-zero variance and significantly correlated features.

Wrapper methods train the classifier through successive iterations where the feedback from the previous iteration is used to select a subset of features for the next one. Certainly, wrapper methods exploit more computation power in contrast to embedded methods, but nonetheless, they are able to remove data points that when taken independently, do not encroach on the differentiation between class labels. The most commonly used wrapper method is the Recursive Feature Elimination (RFE) which selects features by their recursive ranking among smaller and smaller subsets of features via cross-validation.

There are instances where real measurements can be associated with more than one class or group. In such situations, the SVM is extended using a multi-class classification algorithm. The one-versus-all approach is one of the simplest

techniques used in multi-class classification, where the measurements belonging to a certain group j have to be differentiated from all other groups $j = 1, 2, \dots, k$. In this scenario, all the groups other than j are combined into one group and the selection of the hyperplane discriminating the samples in j from the combined group is done using the standard SVM algorithm. Multiple class problems can be generalized by employing trained multiple one-versus all classifiers and use the obtained answers to combine them with a decision function. With $\left(w^{(j)}, b^{(j)}\right)$ as notation, the optimal hyperplane for class j, and $t^{(j)} = sgn(\langle w^{(j)}, x + b^{(j)}\rangle)$ the decision function for j after all the k optimal separating hyperplanes have been found, the final classifier h is expressed in Equation 2.103.

$$h(x) = \arg\max_{j} \left(t^{(j)}(x)\right) \qquad 2.103$$

Li et al. [58] used SVM on acoustic signals analysis based on pseudo-Wigner-Ville Distribution (PWVD) for the diagnosis and prediction of several rotor crack depths. Li et al. [59] proposed a fault diagnosis method for mechanical systems using second-generation WPT, Neighborhood Rough Set (NRS) and SVM, capable of performing fault detection, dimensionality reduction and pattern classification. Wang et al. [60] performed engine fault diagnosis using engine noise signals, onto which HHT was applied for feature extraction followed by the use of SVM for training, taking as input the noise signals' fault feature vectors. Zhang et al. [61] used instantaneous crankshaft speed (ICS) as their basis for the combustion fault diagnosis of a large V-

type marine engine. Feature extraction was done by using a cyclic polar map representation based on ICS. The newly extracted features were used as input to an SVM model. Li et al. [62] implemented a multi-kernel SVM for the fault diagnosis of a gas turbine engine, which achieved improved results in contrast to traditional SVM.

2.7.4 Artificial Neural Networks

ANN is thought to be one of the most widely used algorithms [44]. Their concept originates from biological neural networks which they emulate in a reduced manner. The smallest unit of ANN is a neuron or perceptron, which is a processing element interconnected with other perceptrons. Perceptrons are arranged in a layer or vector, with the output of one layer being the input to the subsequent layer and possibly other layers. ANNs are formed of three components: input layer, hidden layer and output layer. A perceptron may be connected with all or a subset of other perceptrons in a subsequent layer as shown in Figure 2.30, hence simulating the brain's synaptic connections as shown in Figure 2.29.

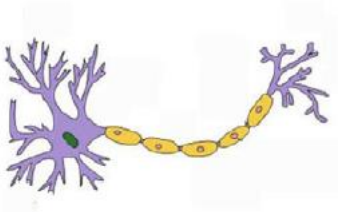


Figure 2.29 A human neuron [44]

Weighted data signals entering a perceptron are analogous to electric signals into a nerve cell, creating a transfer of information in the network. A back propagation neural network (BPNN) is a multilayered perceptron consisting of forward and backward propagation by supervised learning. Multi-class layers process the input samples as shown in Figure 2.30 in forward propagation and they are finally mapped to the target class via the output layer. Given a training set $T = [(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)]$ with N samples, where $x_i \, \epsilon \, \mathcal{R}^d$ contains d features and $y_i \, \epsilon \, \mathcal{R}^l$ include l health states, the output of the hth hidden layer is given as shown in Equation 2.104.

$$(x_i^h)_j = \sigma^h \left(\sum_{i=1}^{n_{h-1}} W_j^h \cdot x_i^{h-1} + b_j^h \right), \qquad 2.104$$

$$j = 1, 2, \dots, n_h, \quad h = 1, 2, \dots, H,$$

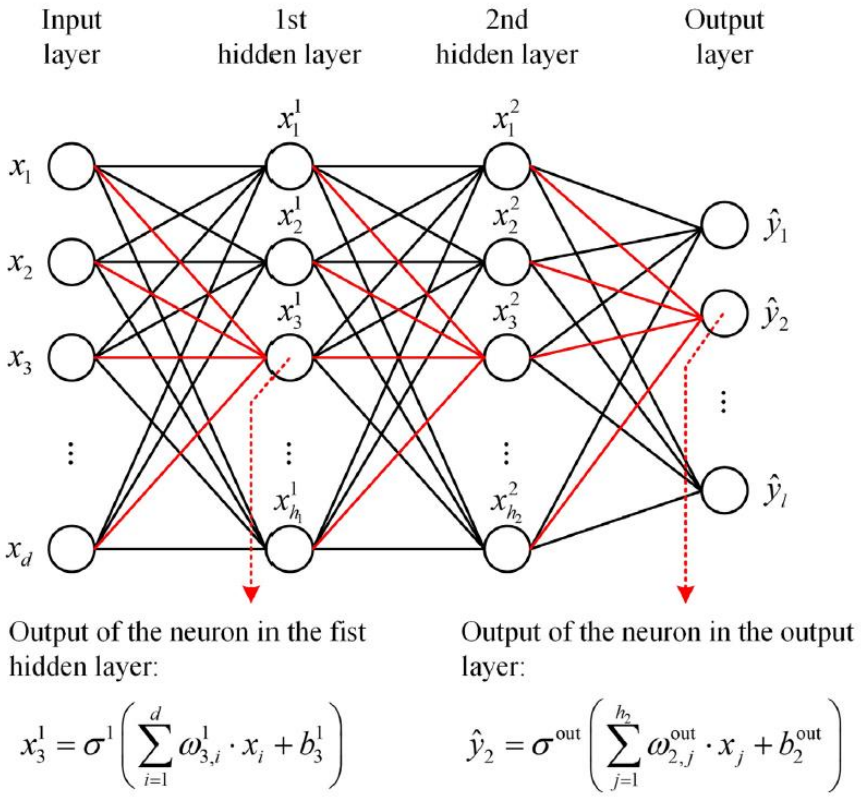


Figure 2.30 Back Propagation Neural Network architecture with 2 hidden layers

[47]

In Equation 2.104, $(x_i^h)_j$ is the output of the jth neuron in the hth hidden layer, and $x_i^0 = x_i$. The number of neurons in the hth hidden layer is given by n_h. σ^h is the activation function of the hth hidden layer and n_{h-1} is the number of neurons in the $(h-1)$th hidden layer. W_j^h is the weights between the neurons in the previous layer and jth neuron in the hth hidden layer and b_j^h is the bias of the hth hidden layer. The output of a BPNN is predicted as:

$$(\hat{y}_i)_k = \sigma^{out}\left(\sum_{i=1}^{n_H} W_j^{out} \cdot x_i^H + b_j^{out}\right), \qquad k = 1, 2, \dots, l, \tag{2.105}$$

where $(\hat{y}_i)_k$ is the output predicted for the kth neuron in the output layer. σ^{out} is the output layer's activation function. W_j^{out} and b_j^{out} are the output layer's weights and bias respectively. For a given training set $\{x_i, y_i\}$, the objective of the BPNN optimization aims at the reduction of the error between the target and the predicted output by:

$$\min_{W,b} \quad E_i = \frac{1}{2}\sum_{k=1}^{l}[(y_i)_k - (\hat{y}_i)_k]^2 \tag{2.106}$$

For the resolution of this problem, the training parameters W and b are updated by gradient descent such that:

$$W \leftarrow W - \eta \cdot \frac{\partial E_i}{\partial W}, \qquad b \leftarrow b - \eta \cdot \frac{\partial E_i}{\partial b}, \tag{2.107}$$

where η represents the learning rate. The error gradient spreads backward from the output layer to the input layer and updates the training parameters layer by layer [63]. ANNs effectuate algorithms that simulate neurological-related capabilities like learning from experience, performing generalizations from analogous events and judging states where poor results were achieved in the past.

Zabihi-Herasi et al [64] used ANN for combustion fault detection and diagnosis on a 12-cylinder 588 kW trainset diesel engine based on vibration data obtained from the intake manifold and cylinder heads. Feature extraction was performed using discrete

wavelet transform and they were used as input into a Multilayer Perceptron (MLP) for diagnosis of faulty cylinders subjected to the abnormal fuel injection. In [65], combustion torque was used as the basis for the misfire fault diagnosis of a gasoline engine. Due to the impossibility of directly measuring combustion torque, an Optimized Luenberger Sliding Mode Observer (OLSMO) was used to estimate dynamic engine combustion torque in transient conditions. Feature extraction was done by summation of the estimated combustion torque for each combustion cylinder in one cycle. The extracted features were used as input for various ANN architectures such as Back-Propagation Neural Network (BPNN), and Elman Neural Network (ENN) for comparison with the latter showing the best results. Wang et al. [66] used engine acoustics signals for fault diagnosis of a gasoline engine. An incomplete wavelet packet analysis (WPA) model consisting of a five-level discrete wavelet transform (DWT) and a four-level WPA was applied to engine noise signals for feature extraction. This was followed by classification using BPNN which was capable of diagnosing nine different fault conditions.

2.7.5 Deep learning

It might be difficult to determine which features should be extracted as input to AI algorithms. Deep Learning (DL) has the potential to alleviate this problem. As such DL has shown top-notch performance in fields such as computer vision, audio recognition, natural language processing, as well as fault diagnosis. DL-based diagnosis models are capable of automatically learning features from the monitored input data and simultaneously recognizing health states in accordance with the learned features. Via non-linear functions models, they are capable of adaptively capturing representation information from natural input signals and approximate complex non-linear functions with little error. They are mostly composed of feature extraction layers and classification layers. First, the models use hierarchical networks like autoencoders (AEs), deep belief networks (DBNs), convolutional neural networks (CNNs), and Recurrent Neural Networks (RNNs) for abstracted features learning layer by layer. Second, the output layer is placed after the last extraction layer for the recognition of the health state, usually with ANN-based classifiers due to their high capacity in multiclass classification. The error between the target and the actual output in the training phase is minimized by backpropagation to update the training parameters of the models.

2.7.5.1 Convolutional Neural Network

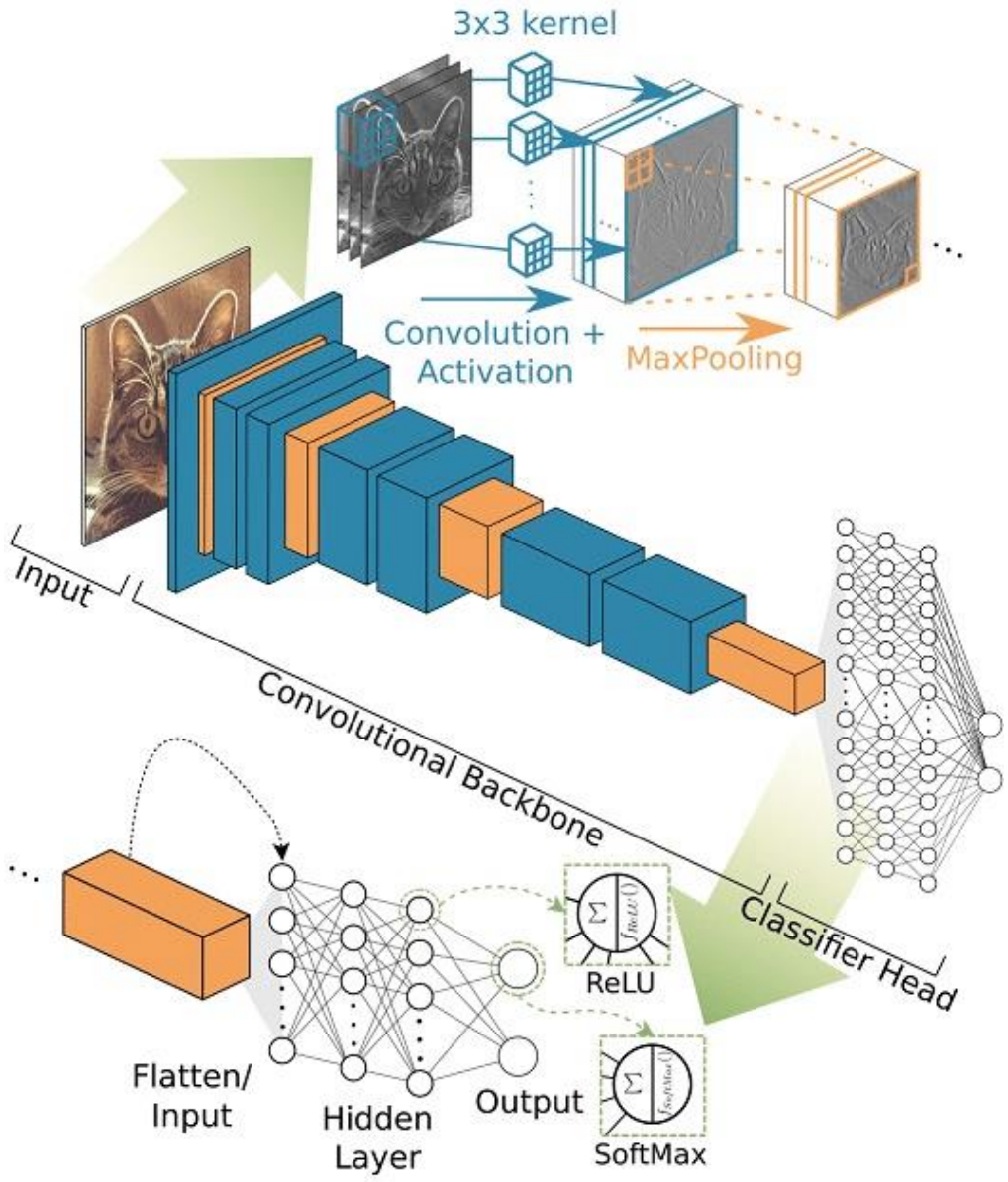


Figure 2.31 Image recognition Convolutional Neural Network architecture: Overview and details [67]

Convolutional Neural Network (CNN) as shown in Figure 2.31 is a supervised DL method which has shown great achievements in image identification, target tracking and speech recognition [68]. CNN is generally composed of convolution layers, pooling layers and full-connected layers. Figure 2.32 details the basic principle of convolution and pooling.

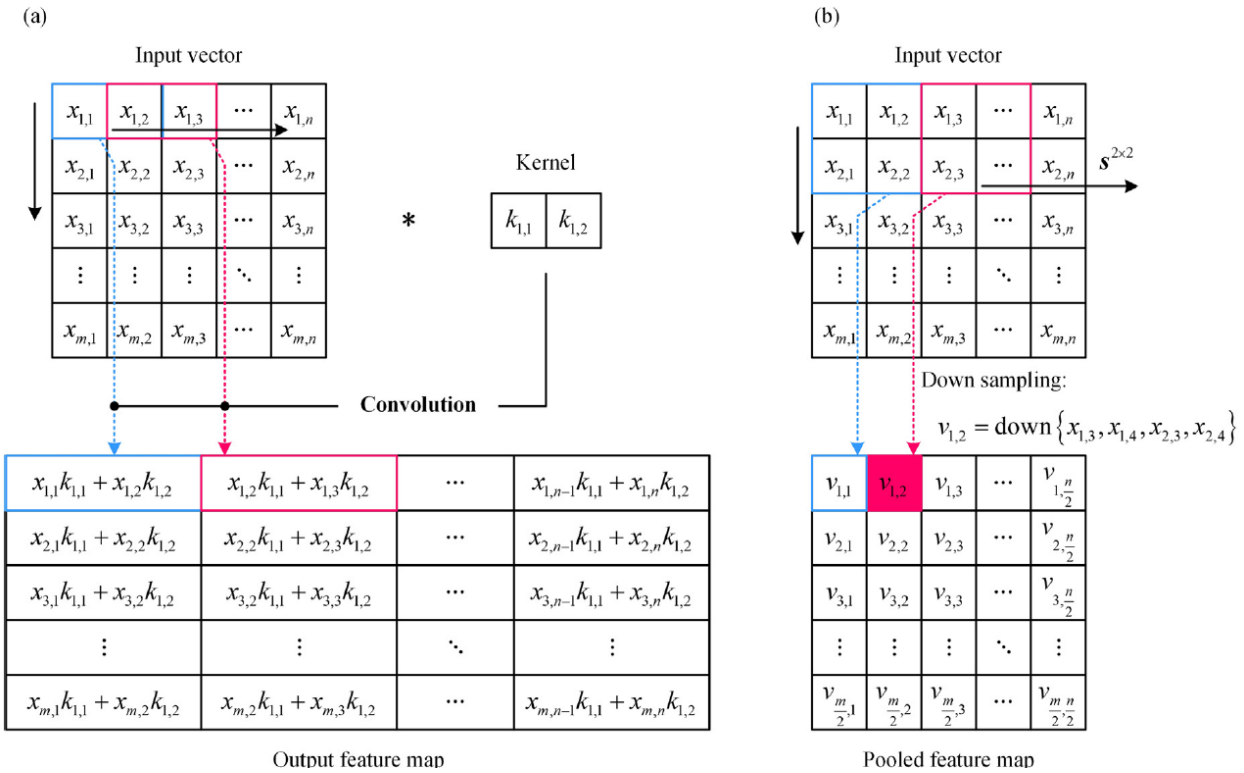


Figure 2.32 a) Convolution process and b) Pooling process [47]

The filter kernels $k^c \in \mathcal{R}^{H \times L \times D}$ used in convolution to convolute the input vector $x^{c-1} \in \mathcal{R}^{M \times N}$ from the precedent $(c-1)$th layer where H is the kernels' height; L and D are respectively the kernels' length and depth. The output mapping of the cth layer is obtained such that:

$$x_i^c = \sigma_r(x_i^{c-1} * k^c + b^c) \in \mathcal{R}^{(M-H+1)\times(N-L+1)\times D} \qquad c = 2, 3, \ldots \qquad 2.108$$

$$(x_i^{c-1} * k^c)_{j,k,d} = \sum_{m=1}^{M} \sum_{h=1}^{H} \sum_{l=1}^{L} x^c_{(i),j+h-1,k+l-1,m} \cdot k^c_{h,l,d}$$

In Equation 2.108 σ_r is the activation function rectified linear unit (ReLU). Down-sampling processing in the pooling layers is effectively used to reduce overfitting and diminish the number of training parameters as shown in Figure 2.32b. Max pooling and mean pooling are the commonly used down-sampling forms and hence the pooling feature map is expressed as shown in Equation 2.109.

$$(v_i^p)_{m,n,d} = down\left\{x^{p-1}_{(i),j,k,d} | \forall x^{p-1}_{(i),j,k,d} \in x_i^{p-1}, j, k \in N^+ \quad , s^{r\times t}\right\} \qquad 2.109$$

$$s.t. \quad s^r \le j \le s^r m, \qquad s^r(n-1) \le j \le s^t n$$

$down\,(\cdot)$ is the down-sampling functions including max $(\cdot)$ and mean $(\cdot)$ respectively, and $s^{r\times t}$ are the filters in the pooling layer. The compilation of convolution and pooling layers, permits CNN to learn the deep layer features from the input data.

These features are subsequently flattened into a 1D vector as the input to the fully connected layers. Furthermore, they are mapped into the target class via the multilayer neural network. The output of the fully connected layers is given as:

$$x_i^f = \sigma_r\left(W^f \cdot x_i^{f-1} + b^f\right), \qquad f = 2, 3, \ldots \qquad 2.110$$

In Equation 2.110, $x_i^1 =$ flatten (v_i^p) is the fully connected layers' output and $\theta^f = \{W^f,\ b^f\}$ are the training parameters of the fully connected layers.

Qin et al. [69] used a multi-domain twin CNN for the misfire fault diagnosis of a diesel engine. In their proposed model, vibration signals from the engine's cylinder heads are fed into the input layer of the model. Three input layers were constructed to automatically extract and combine the time domain, time-frequency domain and handcraft time domain statistical features. The extraction of this multi-domain information of vibration signals was achieved thanks to twin convolutional networks with a large first layer kernel. This configuration enabled the model to be robust to environmental noise and changes in operating conditions and achieve 97.019% accuracy with signal signal-to-noise ratio of 4 dB. In the study presented by Huangfu et al. [70], knock sensors' and accelerometers' signals were decomposed in 2D Mel-frequency Spectral Coefficients feature maps. These 2D spectrograms were stacked into a 3D matrix, hence creating an analogy with an image. This image containing patterns was used as input to a CNN for classification. The proposed method helped in the diagnosis of gasoline engine combustion faults related to spark plugs. In the

paper presented by Shahid et al. [71], the MPU sensor signal measuring the angular speed of a marine diesel engine was converted to a 1D vector Crank Angle Degree (CAD) signal. The 1D signal was the input to a CNN engineered for multi-class classification which combined feature extraction and pattern recognition in a single learner. The reduced time complexity and robustness of the 1D-based CNN architecture were put forward in contrast to 2D-based CNNs. Hence the proposed architecture was tested for the diagnosis of misfires and load changes together in engine operations with more than 99% accuracy.

2.7.5.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a category of ANNs that are tailored to process sequential data and time-series data, by defining the dependencies between various time stamps [72]. However, long-term dependencies are handled by two types of RNNs: The Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU). Under the prism of DL, LSTM has become the main focus due to its production of interesting results for RNN-based models [73]. This success is due to the fact that the information flow in and out of the cell is regulated by the introduction of a memory cell. This memory cell regulates and protects the cell state and is composed of three non-linear gated units. Information flow along an entire network is facilitated thanks to these gated units which eliminate the problem of gradient vanishing by selective pattern remembering, for a long period of time [74]. For this

reason, LSTM networks leverage a huge potential in prediction and prognostics. The LSTM cell architecture is shown in Figure 2.33.

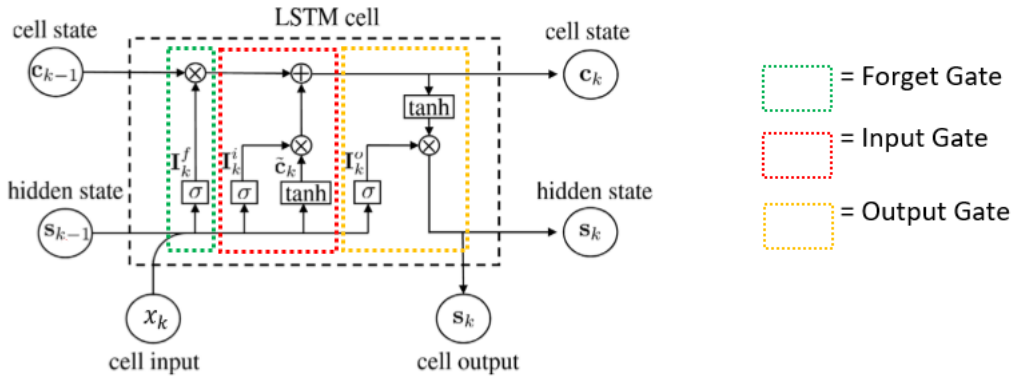


Figure 2.33 An LSTM cell with a detailed overview of its internal operations

An LSTM cell memory is defined by a cell state c_k and a hidden state s_k, both updated during each time step. For each time instance, k the external input $x_k$is fed to the model, and the cell generates the corresponding output s_k The LSTM combines the hidden state s_{k-1} of the precedent time step with the new input x_k, to update the cell state c_k. As a result, the hidden state s_k is updated using a filtered version of c_k, and this updated hidden state also functions as the output of the cell. In each of the gated units, the σ operator is exploited by a neural layer to create a vector $\boldsymbol{I}_k(0 \leq \boldsymbol{I}_k \leq 1)$. Each element in $\boldsymbol{I}_k$ commands the (partial) maintain or erase of its corresponding state cell element. The first gate also known as the *forget gate* rules during each time instance k, what portion of information found in the prior cell state c_{k-1} becomes

irrelevant based on x_k which contains the new information, and the precedent hidden state s_{k-1}. This operation is mathematically expressed as:

$$\boldsymbol{I}_k^f = \sigma(W_1^s s_{k-1} + W_1^x x_k + b_1) \qquad 2.111$$

The *input gate* rules the relevancy of the information inlaid in the input x_k and prior hidden cell state c_{k-1} combined. Two mathematical operations govern the insertion of this new information. The elements of c_{k-1} which are going to be uploaded by the new information are determined by a calculated vector, $\boldsymbol{I}_{\mathrm{k}}^{\mathrm{i}}$. $\tilde{c}_k$typically houses the actual information that should be added to the prior cell state c_{k-1}. The information found in the precedent hidden state s_{k-1} and the input x_k is compressed by the hyperbolic tangent operator between -1 and 1, thereby helping in the network regulation such that:

$$\boldsymbol{I}_k^i = \sigma(W_2^s s_{k-1} + W_2^x x_k + b_2) \qquad 2.112$$

and

$$\tilde{c}_k = tanh(W_3^s s_{k-1} + W_3^x x_k + b_3) \qquad 2.113$$

The updated cell state c_k is obtained by the combination of the information obtained from the forget and input gates. The relevant information in the prior cell state c_{k-1} and in the new information in $\tilde{c}_k$ which will be inserted in the updated cell state c_k, is determined by the sigmoid outputs $\boldsymbol{I}_{\mathrm{k}}^{\mathrm{f}}$ and $\boldsymbol{I}_{\mathrm{k}}^{\mathrm{i}}$, onto which entry wise multiplication is performed such that:

$$c_k = \boldsymbol{I}_k^f \circ c_{k-1} + \boldsymbol{I}_k^i \circ \tilde{c}_k \qquad 2.114$$

The LSTM cell's output c_k is defined by the *output gate* which is also a filtered version of the updated cell state c_k. The components that should be erased from the cell state c_k are determined by the vector $\mathbf{I}_k^o$, which in its essence is influenced by the information contained in x_k and s_{k-1}. Hence:

$$\boldsymbol{I}_k^o = \sigma(W_4^s s_{k-1} + W_4^x x_k + b_4) \qquad 2.115$$

and

$$s_k = \boldsymbol{I}_k^o \circ tanh(c_k) \qquad 2.116$$

In summary, the LSTM cell's memory is characterized by a cell state $c_k \in \mathbb{R}^{n_s}$ and hidden state $s_k \in \mathbb{R}^{n_s}$, both subject to updates at each time step. The initial cell state c_0 and initial hidden state s_0 are initialized to zero. Given an input $x_k \in \mathbb{R}^{n_i}$, the trainable weight matrices are denoted as $W_j^s \in \mathbb{R}^{n_s \times n_s}$ and $W_j^x \in \mathbb{R}^{n_s \times n_i}$ for $j \in \{1,\ldots,4\}$. The corresponding bias vectors are defined by $b_j \in \mathbb{R}^{n_s}$ for $j \in \{1,\ldots,4\}$.

Calvo-Bascones et al [75] presented a methodology to perform the prognosis of a diesel engine generator's cooling system. The focus of the methodology was to analyze the degradation of its cylinders via inputs such as the gross power and the temperatures of the intake and exhaust manifold gas with that of the coolant at the engine block outlet. LSTM was used to learn the temporal features and evolution of two computed indicators (Deviation and Similarity) which were based on the

aforementioned inputs and characterize the behaviours of the engine. Run-to-failure data of a marine diesel engine in [76] was used to investigate the prognostics of two faults namely: an air filter fault and a turbocharger fault. The data obtained was used to construct remaining useful life targets, mapped to sensor measurements. The LSTM network was used to learn this temporal mapping.

Zhang et al. 2023 [77] used a multivariate LSTM trained using the Adam optimization algorithm to realize power demand predictions with input data delivered by the advanced driving assistance system combined with road condition data given by map service providers. This NN architecture inserts itself in a data-driven predictive energy consumption minimization strategy to explore the fuel-saving potential of a connected plug-in hybrid vehicle. The proposed strategy achieved a fuel consumption reduction of 3.1% when compared to the adaptive energy consumption minimization strategy and 13.2% when compared to rule-based control.

Chapter 3: Experimental Set-up

The experimental phase of the research was conducted at the Centre for Mechatronics and Hybrid Technologies (CMHT) at McMaster University. The CMHT houses a dynamometer laboratory (henceforth referred to as “dyno lab”) where ICEs can be tested as shown in Figure 3.1.

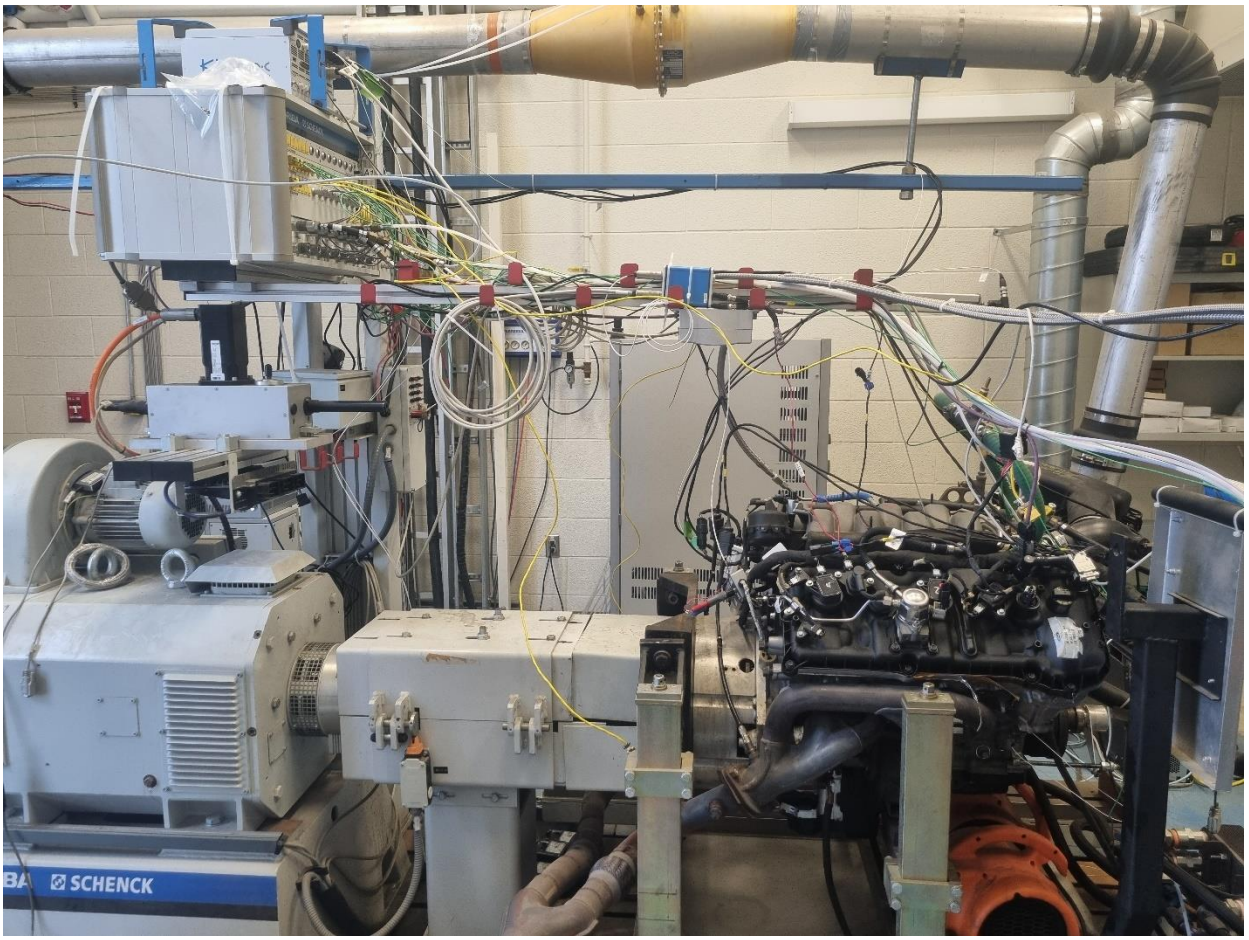


Figure 3.1 Engine connected to a dynamometer

As shown in Figure 3.1, the dynamometer shaft is coupled to the engine shaft and hence engine power output can be measured with accuracy. The test cell houses various measurement equipment to collect data related to pressure, temperature, vibration, sound, and light. The test cell is part of a larger framework which is the dyno lab as shown in Figure 3.2. Further description of the dyno lab's main facilities is found downstream of this chapter.

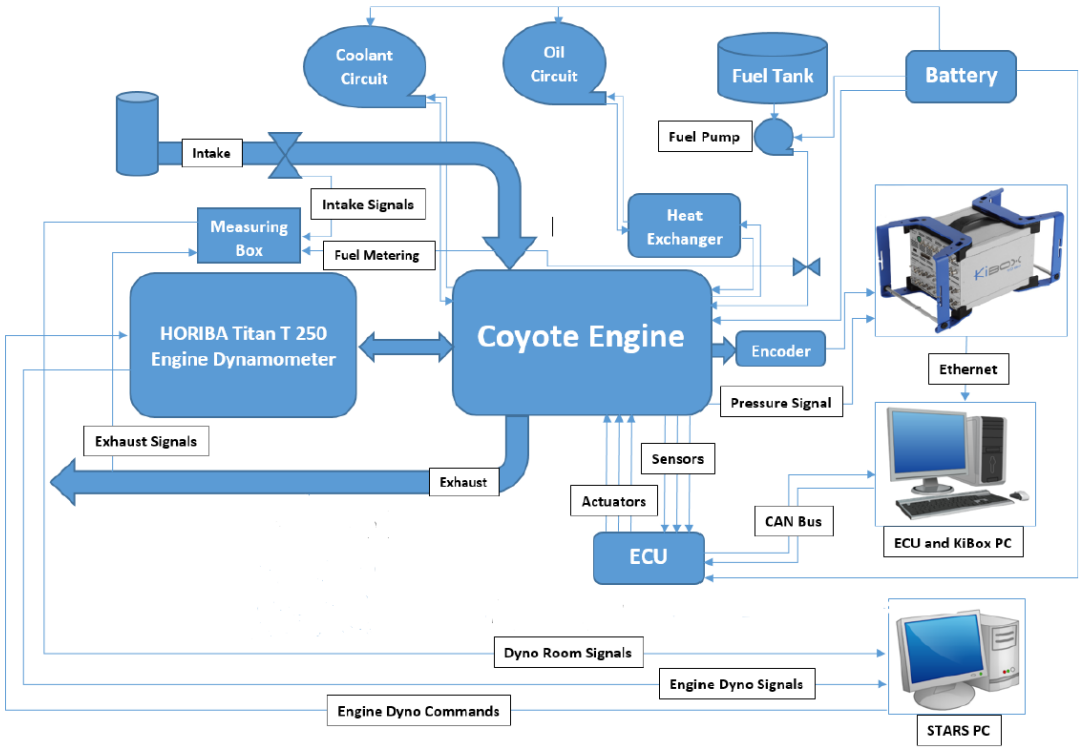


Figure 3.2 Engine Dyno Lab

The dyno lab integrates the control loops, connecting pipes, accessories, communication and measuring systems in addition to an operator place for total control of experimental research activities. The dyno lab provides open access to all

engine components and thus enables engine fine-tuning and development under optimum operating conditions. In addition to performance testing (which includes fuel consumption, emissions, speed, torque and in-cylinder pressure), other engine development and test activities can be conducted on aspects such as combustion, engine controller calibration, engine controller unit (ECU) development, and emerging technologies in FDD. The dyno lab at the CMHT was instrumental in [78] where it was used to provide data for the elaboration of an FDD solution for faults such as exhaust gas recirculation valve and cylinder leakage. The CMHT dyno lab was used in [79] where engine data was collected by Mehdi Sadeghkazemi to assess the performance of a novel NanoSpark spark plug. The data collected by Mehdi Sadeghkazemi is the one used for the implementation of the FDD solutions in this research.

Further to the process followed by [79] and prior to conducting the actual engine testing, some preliminary safety guards were followed to address the issue of results repeatability. Engine tests were performed more than once to check the testing process's precision. Parameters that could affect engine outputs (torque, exhaust temperature, fuel consumption and emissions) were monitored to satisfy the repeatability condition. These could be any of the engine fluids' conditions, control parameters, accessory parameters and ambient air conditions in the test cell. This monitoring process is essential when comparing the alternative engine hardware changes as well as FDD implementation.

At constant air temperature and speed, the intake manifold pressure is proportional to the air and fuel mass flow rates into the cylinders. Hence the engine load and power are dictated by the cylinder charge. The ECU controls the air mass flow and fuel injection timing by taking information from the torque and power demand, and other vehicle subsystems to determine the opening of the throttle and control the latter thanks to a PID controller [80]. In order to avoid unfair comparison, the engine load was set by overriding the throttle position directly from the ECU. Other affecting parameters to the air mass flow such as air intake temperature and intake and exhaust valve timing were monitored. In addition to engine load, the second parameter determining engine operating points is engine speed and it was controlled by operating the engine in Throttle-Speed mode via the dynamometer.

3.1 Testing Set-up

Table 3.1 lists the engine test cell facilities at the Centre for Mechatronics and Hybrid Technologies (CMHT) which were used for data collection under and as described in [79].

Table 3.1 Testing set-up facilities

Facility	**Description**
Engine (Gasoline)	2018 Ford Coyote Engine, Gen 3
Engine Dynamometer	Horiba Schenk Titan T 250
Engine Control Unit (ECU)	EFI Technologies Open Access ECU
Crank Angle Position Sensor	Kistler Optical Encoder type 2614CK
Combustion Analyzer	Kistler KiBox to Go type 2893A
Dynamometer Accessories	Fuel Supply System, Engine Coolant Circuit, Lube Cooling Circuit, Electrical Power Supply System

3.1.1 Ford Coyote Engine

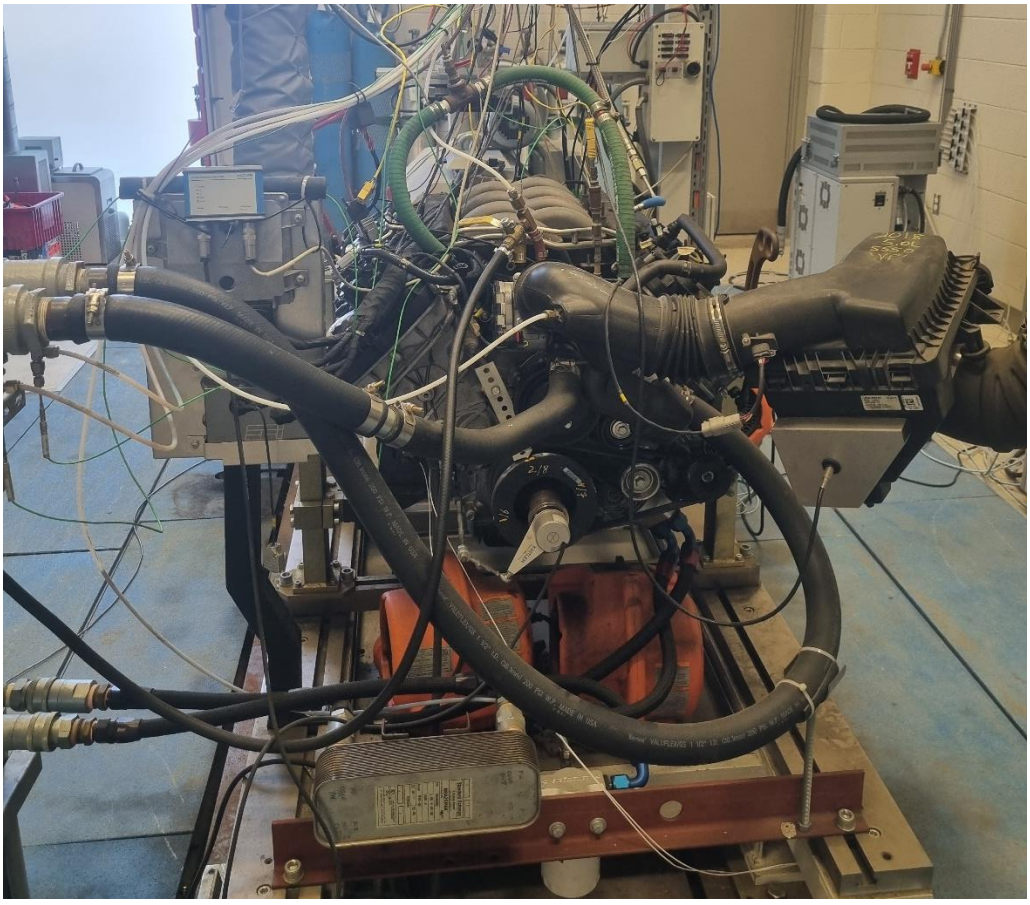


Figure 3.3 Ford Coyote Engine

The engine used for the experiment is manufactured by Ford Motors Company and is a Generation 3 2018 Ford Coyote Engine as shown in Figure 3.3. Table 3.2 summarizes the engine's specifications.

Table 3.2 Ford Coyote Engine Specifications

Metric	**Specification**
Maximum Horsepower	343 kW @ 6750 rpm
Maximum Torque	569 Nm @ 4500 rpm
Number of cylinders	8
Displacement	5.038 litres
Compression ratio	12.1
Bore	93.0 mm
Stroke	92.7 mm
Firing order	1-3-7-2-6-5-4-8
Fuel system	Direct Fuel Injection
Spark plug	12405
Spark plug gap	1.25-1.35 mm
Valvetrain	Dual Overhead Camshaft (DOHC), 4 valves/cylinder
Weight (without accessory drive components)	205.5 kg

3.1.2 Engine Dynamometer

The CMHT has at its disposal a Titan T 250 dynamometer manufactured by HORIBA Automotive Test Systems as shown in Figure 3.1 with specifications summarized in Table 3.3.

Table 3.3 HORIBA Titan Engine Dynamometer Specifications

Metric	Specifications
Power	220 KW
Torque	600 Nm
Speed	8000 rpm
Idle Speed	>700 rpm
Mass moment of inertia	> 0.15 kgm^2

It is composed of 5 modules (shown in Figure 3.4), namely:

- Loading Unit, which consists of a Dynas3 asynchronous electric motor/generator with a power unit, a shaft connection to the engine and a SPARC controller
- Test stand automation system, which controls and monitors engine tests using STARS software

- Data logging, which consists of measuring sub-modules coupled to other optional measuring instruments all housed in a measuring box for the measurement of variables during engine testing
- Media conditioning, which are optional units that can be applicable to engine fuel, lube and coolant conditioning
- Operator place, which consists of an instrumentation and control cabinet coupled to a PC with a built-in cable lead and holder for the STARS-PC automation software

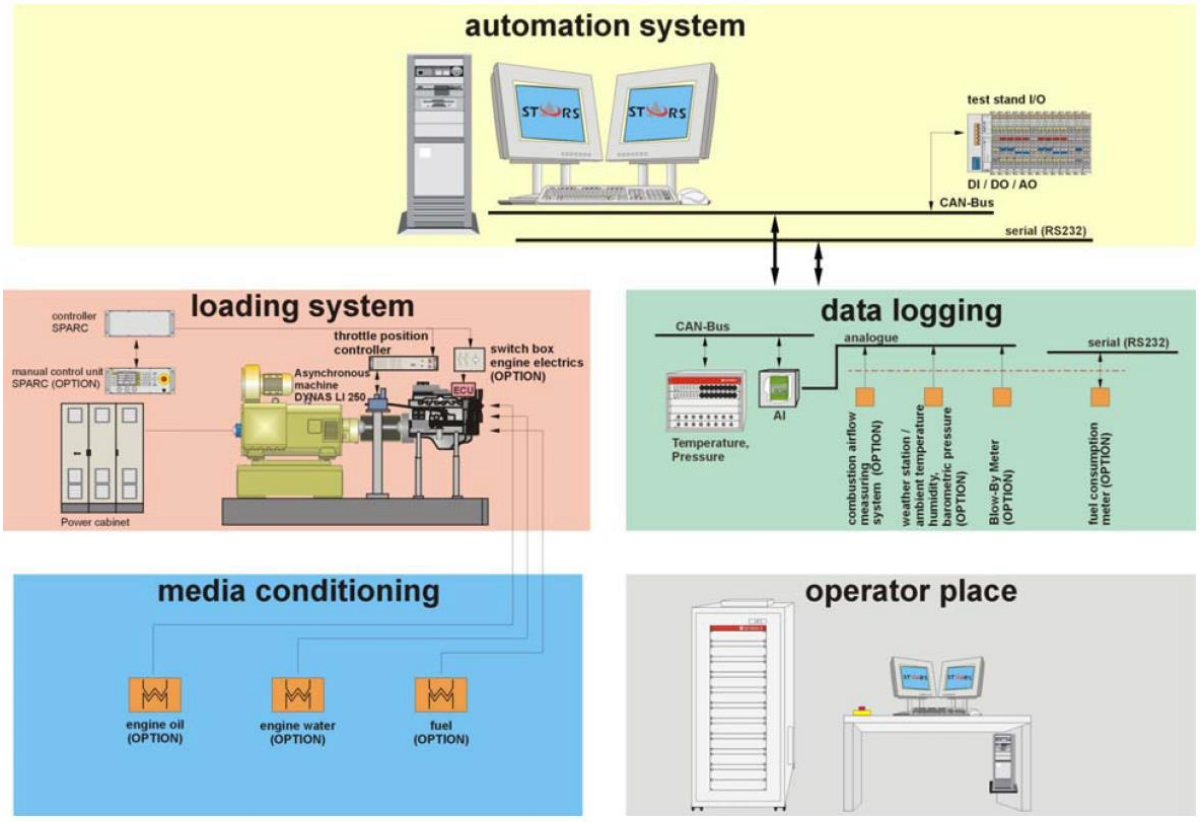


Figure 3.4 Titan T 250 test stand overview [79]

3.1.3 Engine Control Unit

The Ford Coyote engine used for the experiment has an ECU manufactured by EFI Technologies Incorporation, which is a supplier to Ford Motor Company for its racing activities. Engine data logging and control parameters are openly accessible to users via the ECU. When investigating the effects of a certain hardware configuration on engine performance, it is possible to maintain other parameters constant or locked in order to exclude their effect on engine outputs thanks to the ECU. To maintain consistency in results, all control parameters in addition to factors affecting the engine fluids' conditions and ambient air should be monitored and if necessary, kept the same. A portion of this recommendation is executed by the ECU control functions while the other portion related to the engine dynamometer is executed by the HORIBA Titan T 250 media conditioning unit module. The ECU Configuration Tool (ECT) is the software that operates the EFI and enables data logging and analysis. It also enables engine management system calibration for calibration adjustments, duplication or loading.

3.1.4 Combustion Analyzer

To tap into the knock sensor signals coming from the engine, a measurement and analysis system called KiBox To Go manufactured by Kistler Instruments AG was used. This equipment was developed to measure in-cylinder pressure and engine signals. It is operated by its associated software called KiBox Cockpit and can

synchronously present and analyze combustion results in real-time and simultaneously with other control and process variables of the ECU. It has a resolution of 10 samples per crank angle (equivalent to 7200 samples per engine cycle) via its 8 analog input channels. The measurement set-up can be connected to an engine's ECU and provide the latter combustion and other engine measurements for control and diagnosis. On the other hand, the KiBox can also be set up in a standalone configuration, where it is not connected to the engine ECU. The standalone configuration as shown in Figure 3.5 was what was used in this research.

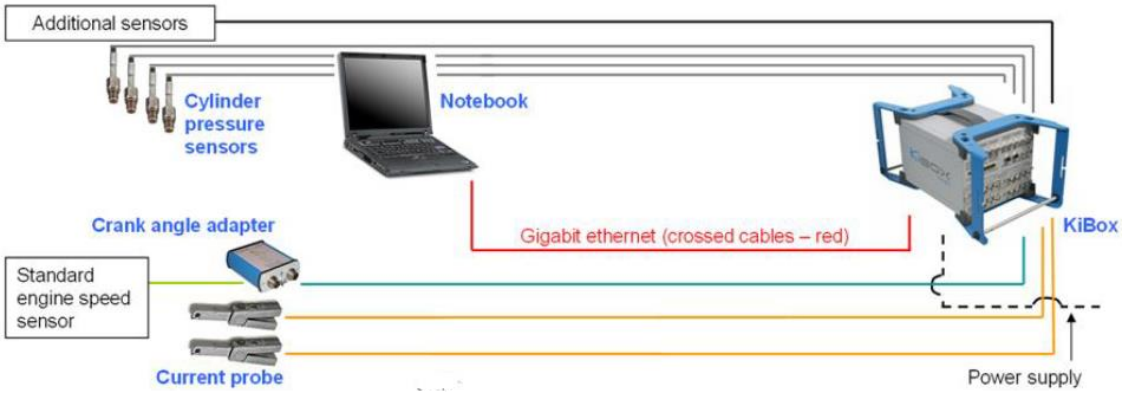


Figure 3.5 KiBox set-up in a standalone configuration

The high resolution provided by the KiBox was one of the parameters that dictated the use of this equipment to perform knock sensor data collection. The equipment is also capable of collecting the signal of the crank wheel sensor and determining the current crank angle position thanks to a crank angle adapter which can use the engine crank wheel position sensor or Kistler optical encoders if a higher precision is

required. Hence the measured variables can be resolved in the crank angle domain with reference to the Top Dead Centre (TDC) of cylinder number 1 (indicated in Figure 3.8) at the power stroke. This resampling capacity in the crank angle domain is the second parameter that dictated the use of the KiBox. The determination of the TDC is a critical step for data analysis in the crank angle domain. For this purpose, a pressure sensor was inserted at a precise location in the cylinder head as shown in Figure 3.6, with Table 3.4 resuming the technical characteristics of the Kistler 6125C pressure transducer.

Figure 3.6 Kistler 6125C pressure transducer (left) and combustion chamber location (right)

Table 3.4 Kistler 6125C pressure transducer technical characteristics

Metric	**Range**
Measuring range	0 to 300 bar
Operating temperature range	-20 … 350 °C
Natural frequency	> 70 kHz
Overload	300 bar
Torque wrench setting	10 Nm
Insulation resistance at 30^0 C	≥ 1013 Ω
Thermal shock error (at 1500 rpm and IMEP = 9 bar)	$\Delta p \leq 0.3\ bar$

The engine was operated by the dynamometer in motoring mode, to simulate the coasting of the reference cylinder. Coasting is the unfuelled running of the engine, where the crankshaft is solely driven by mechanical forces in the absence of combustion. In motoring mode, only air is present in the cylinders and it can be considered as an ideal gas such that the compression expansion process becomes frictionless. Hence the in-cylinder pressure signal, deprived of any combustion effects, enables the TDC to be determined by locating the point of peak pressure while considering thermodynamic losses caused by heat transfer and air leakage. The position of the TDC is very important as an error of 1^0 can cause up to 10% error in

indicated mean effective pressure, which directly correlates with engine torque and power [81]. The resulting TDC was stored as a reference in KiBox Cockpit for subsequent data collection.

3.1.5 Crank Angle Encoder

The TDC reference coupled to the crank angle signal is needed by the KiBox setup for the synchronization of engine data collected and engine revolution. The relatively low resolution of the crank angle signal, oriented the experimental phase of the research to opt for an optical encoder, namely Kistler optical encoder type 2614CK as shown in Figure 3.7 which offers a much higher resolution of 0.5^0 of crank angle and can operate to speeds up to 1200rpm.

Figure 3.7 Type 2614CK Kistler optical encoder

3.1.6 Dynamometer Accessories

The engine dyno lab as shown in Figure 3.2 houses a number of circuits, loops, and systems which contribute to the operation, monitoring and control of the engine. Fuel is delivered with the required pressure to the engine thanks to a fuel system, in which fuel flow is measured for fuel consumption calculations. The lube and coolant circulation pumps with the fuel supply pump, in addition to the engine ignition system and ECU are electrified by the power system. The supply voltage also directly feeds some engine sensors and actuators, which imposes the supply to be free of any oscillations in order to have no impact on engine control phenomena on aspects such as variable camshaft timing and lambda control.

The dyno lab's lube and coolant circuits are connected to the engine for the control of the latter's oil and coolant temperatures. The engine and dyno lab's cooling circuits are directly connected to each other, allowing the engine's coolant carrying excess heat to circulate through the dyno's cooling circuit. On the other hand, the engine's and dyno's lube circuits are not in direct contact and the heat transfer from the engine to the dyno circuit is performed via a cross-flow heat exchanger. The adjustment of the set points coupled to the control of the lube and coolant flow via their corresponding control flow valves enables the regulation of the engine's lube and cooling temperatures thanks to the STARS automation system onto which both lube and cooling systems are linked.

It should be noted that additional sensors are installed in the dyno lab to measure intake air pressure, temperature, and flow, ambient air pressure, temperature and humidity, exhaust gas temperature, engine lube and coolant temperature and pressure, and the fuel line pressure. The different quantities measured enable the engine dyno operator to perform monitoring during engine tests and analyze the effect of each parameter on the engine output to protect the integrity of the engine and engine dyno facilities.

3.2 Data Logging and Management

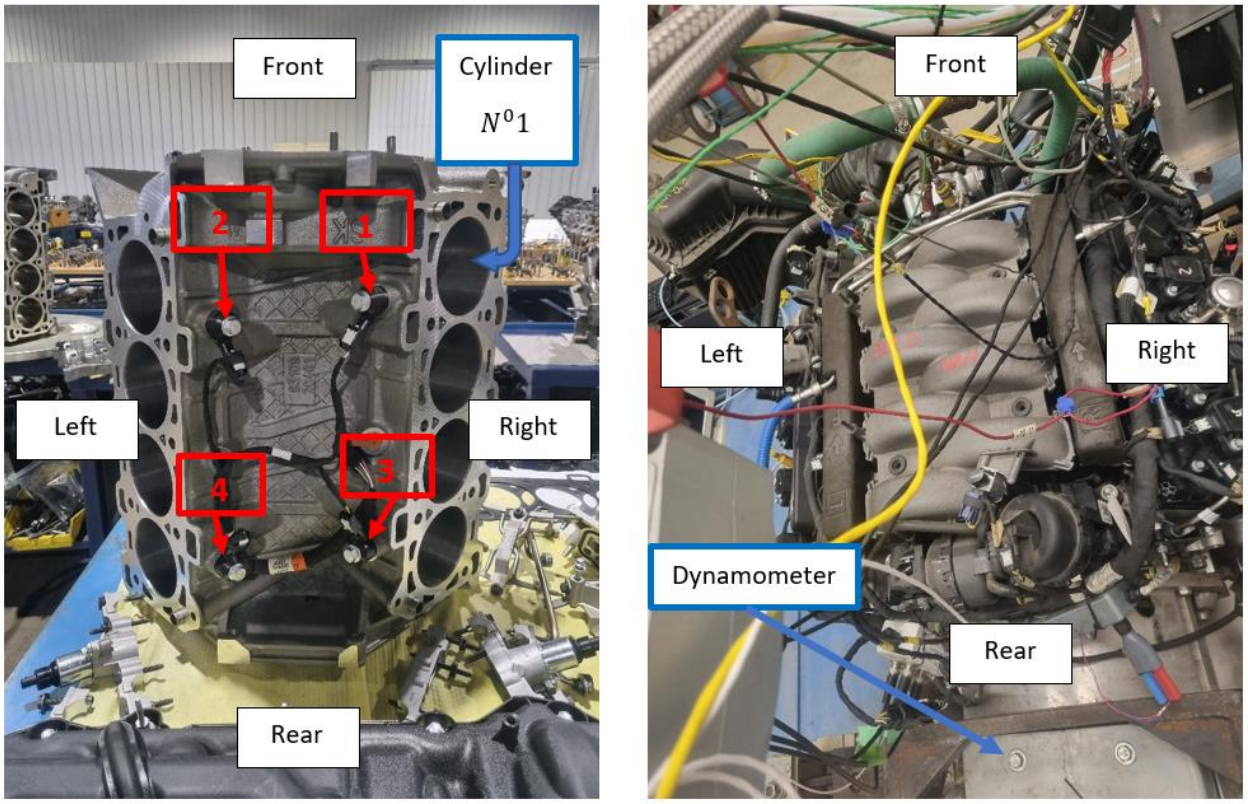
Throughout the experiment, three different data logging/control paths were present namely: The Coyote engine's ECU, the Horiba dynamometer's software STARS, and the Kistler combustion analyzer. In the optic of keeping the recorded engine parameters as close as possible for a fair comparison between engine hardware changes, close attention was kept on engine air flow as it was the parameter that affected the most engine torque, in addition to control variables such as throttle position, exhaust and intake valve timing, and finally the ambient conditions. All comparative tests were conducted within the same day for a specific engine speed to avoid day-to-day variations. A large number of parameters from the three aforementioned paths were recorded with different formats of origin. Therefore, to facilitate the post-processing step of the experiment, an extensive MATLAB m-file was developed to process the large volume of recorded data. With recorded

parameters within an acceptable range to satisfy the repeatability of tests, four main variables were of key interest throughout the experiment. The variables were the four knock sensor signals which were logged via the Kistler combustion analyzer path and used for the purpose of fault detection and diagnosis of the various engine configurations that simulate a particular health state. In the MATLAB m- file, these signals were logged as indicated in Table 3.5.

Table 3.5 Logged knock sensor signals

Signal Name	**Location on Engine**
UNI0	Right Front Knock Sensor (1)
UNI_A0	Left Front Knock Sensor (2)
UNI_B0	Right Rear Knock Sensor (3)
UNI_C0	Left Rear Knock Sensor (4)

The knock sensors are hidden by the intake manifold as shown in Figure 3.8 b. Hence the manifold had to be removed to reveal the position of the knock sensors in Figure 3.8 a as described in Table 3.5



a **b**

Figure 3.8 Knock sensor locations

3.3 Engine Faults Dataset

The data used for the development of the fault detection and diagnosis strategy in this research was collected by Mehdi Sadeghkazemi [79] on the case study of spark plugs. The spark plug design has a significant impact on flame kernel development, engine performance and emissions. One of the key aspects of a spark plug design that must be set properly before the plug is installed in the engine is the spark plug gap. This is due to three main reasons: (1) if the gap is too wide, the electric arc may not have enough energy to bridge the gap, which would result in a misfire. (2) if the gap is too narrow a lean air/fuel mixture may not be ignited, which could result in a misfire. (3) there is a direct proportional relationship between the required voltage and the gap [82].

Over the course of data collection, one of the cylinders was subjected to different health states by exchanging the Original Equipment Manufacturer (OEM) spark plug with an aftermarket one. The aftermarket spark plug was precisely profiled to have two incorrect gaps of 0.020″ and 0.080″, one less than and the other above the recommended gap range. In this way, the fault conditions could be achieved in a controlled manner. It should be noted that aftermarket spark plugs were used for changing the gap due to the fact that on the OEM spark plug, the changes could not be easily done without causing fault effects difficult to evaluate by the scrapping of the precious metal terminals. This is what dictated the choice of aftermarket spark

plugs that do not use precious metal, and hence precise quantifiable and repeatable fault conditions were achieved. The likelihood of retarded combustion or uneven firing causing the engine to run erratically due to the fault conditions was acknowledged throughout the experiment. Evaluating the fault and detection strategy over a controlled combustion irregularity was the prime motive. Three engine configurations, simulating three different health states were defined for the engine test; namely:

- Healthy, where the OEM spark plug was used with the correct gap of 0.051″
- Fault 1, where the aftermarket spark plug was used with a gap of 0.020″
- Fault 2, where the aftermarket spark plug was used with a gap of 0.080″

The data set as summarized in Table 3.6 consists of measurements from the four knock sensors in Table 3.5, recorded in the crank angle domain at 10 samples per crank angle or 7200 samples per engine cycle to fully grasp the engine vibration dynamics under the various health states. The two operating conditions for the engine were 700 rpm at 15% load and 1000 rpm at 25% load. For every engine run, the speed and load were kept constant throughout. Seven hours worth of data was collected over seven days for each operating condition, resulting in a data set containing 168,000 engine cycles in total, meaning 28,000 engine cycles for each combination of spark plug gap and engine speed. It should be noted that for each health condition, the data was gathered in two stages on the same day. That is, the first half of the data was gathered during the morning (AM) and the second half in

the afternoon (PM). The data from days 1 to 7 was what was used to build, train and test the deep learning model for classification. To simulate an operating environment with perturbations in data quality, additional engine cycles were gathered on the 8th day to perform another test on the model. Hence, the complete data set is composed of 192,000 cycles equivalent to 32,000 engine cycles for each combination of spark plug gap and engine speed.

Table 3.6 Engine Fault Dataset for both 700rpm and 1000rpm operating conditions

	Number of Cycles		
Test Day	Healthy – 0	0.020″ aftermarket - 1	0.080″ aftermarket - 2
1	8000	8000	8000
2	8000	8000	8000
3	8000	8000	8000
4	8000	8000	8000
5	8000	8000	8000
6	8000	8000	8000
7	8000	8000	8000
8	8000	8000	8000

It should be noted that the different health states were simulated in Cylinder number 1 for the demonstration of the fault detection and diagnosis strategy. Should

the strategy be extended to all cylinders, additional measurements will have to be gathered for the various conditions in each cylinder. Finally, the 192,000 engine cycles were considered as samples composed of multidimensional time series corresponding to the 4 knock sensor measurements. The samples were then labelled in accordance with their health states (fault conditions) into three classes namely: healthy gap – 0, smaller gap (faulty) – 1, and larger gap (faulty) – 2. Further post-processing of the three classes coupled with model training was performed in the methodology section of the research.

Chapter 4: Methodology

This chapter presents the process that was used to satisfy the research objectives. Figure 4.1 depicts an overview of the methodology, composed of two main steps:

- Input construction
- Convolutional neural network

Each sample, corresponding to an engine cycle serves as the input data to the algorithm. If $X_1 = [x_1^1, x_1^2, \dots, x_1^m]^t$ is a column vector with length m, corresponding to the number of measurements (7200) of a knock sensor signal for one engine cycle, then given 4 knock sensors used in this engine, the input data is represented as:

$$X = [X_1,\ X_2,\ X_3,\ X_4] \qquad 4.1$$

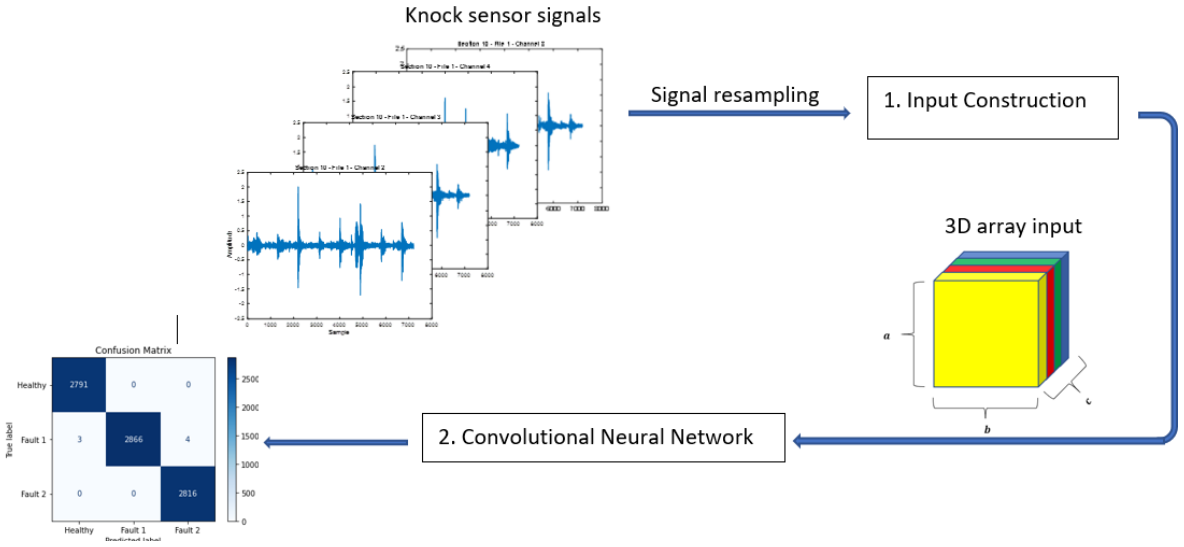


Figure 4.1 Methodology

4.1 Input construction

This step helped extract the input data's most significant characteristics to facilitate the detection and diagnosis part of the overall strategy and remove as many noisy elements in the data as possible. X is multidimensional, where every X_i or channel, captures the engine vibrations from its location as shown in Figure 3.8. Each channel, which represents a knock sensor is analogous to a camera and the engine vibration is analogous to a scene. The recording of the engine vibration by the knock sensors can be conceptualized as taking a picture of the same scene from cameras located at four different angles. In other words, the knock sensors capture the vibration signature of the engine from four different positions. This way, if a fault's vibration signature is expressed more in a certain location, the knock sensor closest to that point could record better that signature. To eliminate the differences in scale

of the various signals, which if left as such might allow skewed features and mislead the FDD strategy, a Z score normalization pre-processing step was applied to the input data as discussed in Equation 2.95. Following normalization, STFT was applied to the input data to represent the vibration signature in the crank angle frequency domain. The sampling rate was 10 samples per crank angle; hence for the operating point of 700rpm, the sampling frequency equivalent in the time domain is 42 kHz, while for the 1000rpm operating point, the sampling frequency is 60 kHz. The sampling frequency is calculated using Equation 4.2. Taking for example the 1000rpm case, the sampling rate can be calculated as shown in Equation 4.3.

$$Sampling\ rate = \frac{n^0\ of\ samples}{crank\ angle} \times \frac{n^0\ of\ crank\ angles\ per\ revolutions}{n^0\ of\ revolutions} \times \frac{n^0\ of\ revolutions}{minute} \qquad 4.2$$

$$Sampling\ rate = \frac{10\ samples}{1\ \cancel{CA}} \times \frac{360\ \cancel{CA}}{1\ \cancel{rev}} \times \frac{1000\ \cancel{rev}}{1\ \cancel{min}} \times \frac{1\ \cancel{min}}{60\ sec} = \frac{60000\ samples}{sec} \qquad 4.3$$
$$= 60KHz$$

The generated spectrogram was obtained by using the following parameters: Hamming window of 128 points, a 75% overlapping and DFT containing 1024 samples. The data matrix is transformed

$$from\ X = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & x_4^1 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^m & x_2^m & x_3^m & x_4^m \end{bmatrix} to\ X = \begin{bmatrix} s_1^1 & s_2^1 & s_3^1 & s_4^1 \\ s_1^2 & s_2^2 & s_3^2 & s_4^2 \\ \vdots & \vdots & \vdots & \vdots \\ s_1^m & s_2^m & s_3^m & s_4^m \end{bmatrix}$$

$s_1^i, s_2^i, s_3^i, s_4^i$ are respectively the spectrograms of their corresponding $x_1^i, x_2^i, x_3^i, x_4^i$. s is a 2D array of shape (a,b). For each sample, the corresponding spectrograms were reshaped to be in the form s^i as shown in Figure 4.2 such that s^i is a 3D array $\in \mathbb{C}^{a \times b \times c}$ where a *and* b represent the size of the spectrogram with a being the number of frequency bins and b the number of time steps, while c represents the number of channels. It should be noted that the size of the spectrogram generated with the parameters mentioned earlier is constant irrespective of engine speed. This is because the raw data is resampled according to crank angle revolution and not speed. Hence the sampling frequency is automatically adjusted during the signal resampling in the crank angle domain so that a resolution of 10 samples per crank angle is always obtained.

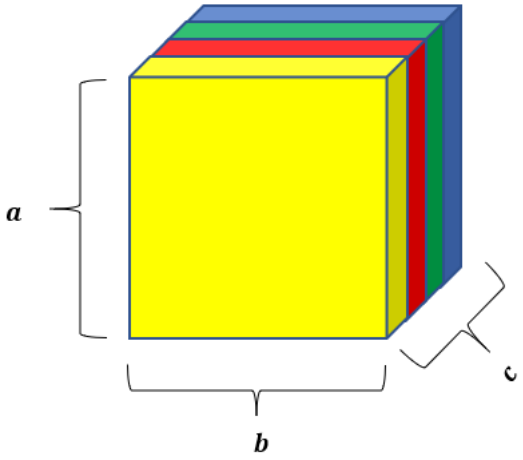


Figure 4.2 Schematic representation of s^i

The data matrix, $X \in \mathbb{C}^{m \times a \times b \times c}$ becomes a 4D tensor as shown in Equation 4.4 with shape (m,a,b,c).

$$X = \begin{pmatrix} s^1 \\ s^2 \\ \vdots \\ s^m \end{pmatrix} \qquad 4.4$$

Furthermore, the absolute values of the spectrograms were computed and expressed on a log scale for better feature representation as shown in Figure 4.3.

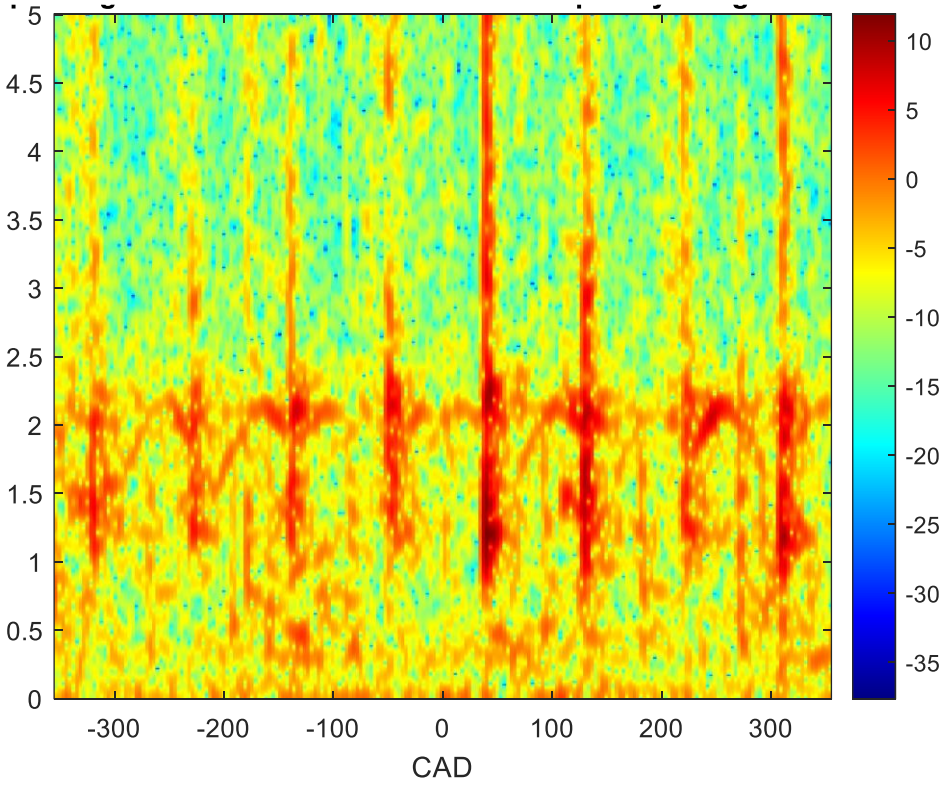


Figure 4.3 Spectrogram of a knock sensor signal

From Figure 4.3, it can be observed that most of the dynamics happen within the frequency range between 1 sample/crank angle and 2.5 samples/crank angle which correspond respectively to 6KHz and 15KHz. Therefore, the spectrogram was truncated to keep features only within that range as shown in Figure 4.4. At this point, $X \in \mathbb{R}^{m \times a \times b \times c}$ and each sample in X constitutes the input to the second section of the methodology. The result of the input construction step is a 3D array $\in \mathbb{R}^{a \times b \times c}$.

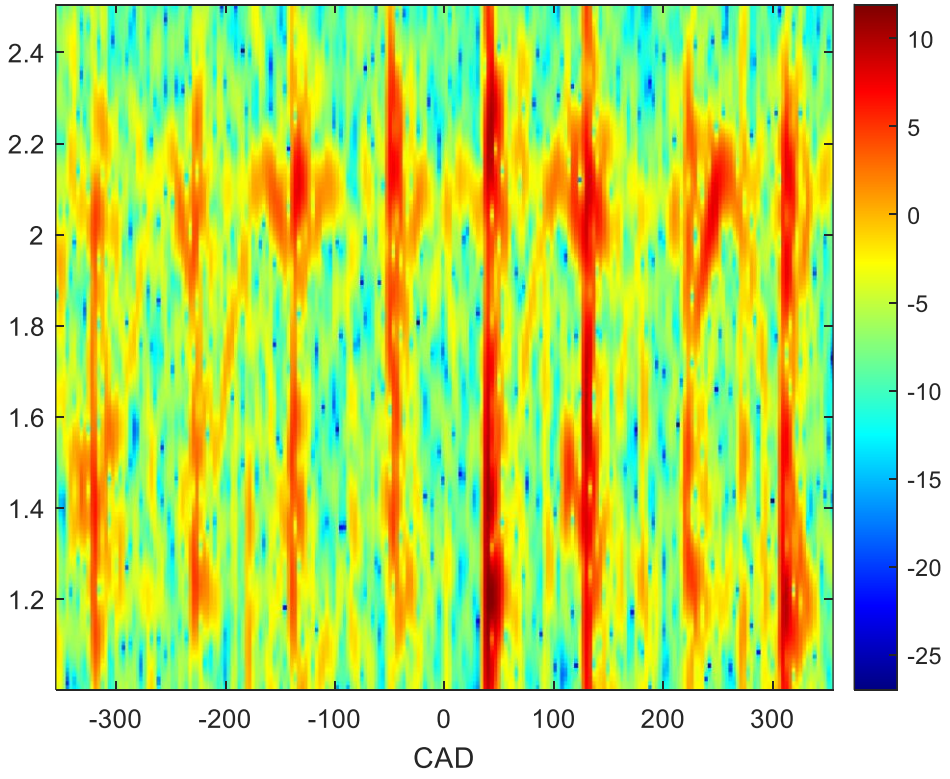


Figure 4.4 Truncated Spectrogram

4.2 Convolutional Neural Network

A combination of feature extraction and pattern recognition was performed on the input data to perform the FDD. The CNN architecture was able to learn the crank angle-frequency patterns found in the input data. Indeed, the CNN could detect effectively variations in contrast within the spectrogram, which are caused by differences in magnitudes at specific frequency bands for various fault conditions. The CNN architecture used for the FDD strategy in this research was inspired by the LeNet5 design [83] which has the following characteristics:

- The convolutional layers are composed of three parts: convolution, pooling, and nonlinear activation functions
- The convolution is used to extract special features
- Subsampling is performed using average pooling
- The activation function is tanh
- Fully connected layers are used as the last classifier
- Computation reduction thanks to the sparse connection between layers

It should be noted that these characteristics can be altered to tailor a particular problem as the architecture serves mainly as a basis for further development. The selection of LeNet5 in contrast to other designs such as AlexNet and VGGNet was dictated by the fact that it is simple, computationally efficient and has inferential performance in real-time fault diagnosis. The LetNet5 advantages were highlighted

in a comparison between the three architectures in [84] where they were used in smart behaviour biometric systems taking as inputs face data for identification and eye blink characteristics for behaviour tracking.

The input to the CNN architecture is a 3D matrix of shape (a, b, c). However, the type of CNN used was a 2D-based CNN, and the input had to be reshaped to have a shape in the form $(a, b \times c)$ as shown in Figure 4.5.

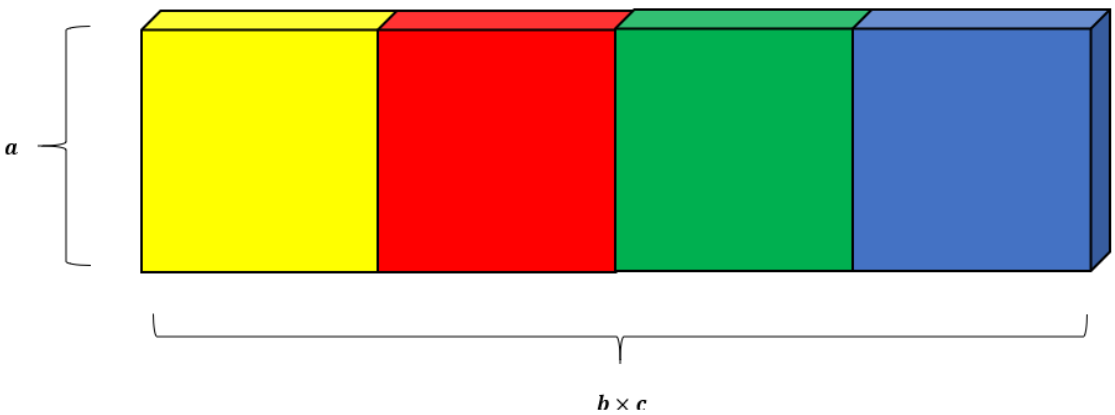


Figure 4.5 Input reshaping

The first layer consists of a 2D convolutional layer with a 5x5 kernel, 16 filters, and a ReLU activation function in addition to batch normalization. Following this, there is a max pooling layer with a size of 2, which divided the size of the feature maps by 2 as shown in Table 4.1. These two layers are then followed by the application of a dropout of 0.25 whose purpose is to regularize the weights. Batch normalization and dropout are essential as they help in preventing overfitting problems. The next convolution and pooling layers are similar to the previous two

with the difference of the convolution layer having 32 filters instead. Even though max pooling layers diminish the size of the extracted feature maps, augmenting the number of filters in operations of convolution helps to pull out complex hidden features.

After these layers, the input sizes are disregarded via the introduction of a flattening operation, then the flattened featured maps are fed consecutively to two fully connected layers of 128 and 64 units respectively and finally fed to another fully connected layer, whose units correspond to the number of classes – in this case, three. Table 4.1 describes the properties of each layer and the number of parameters. The data set from days 1 to 7 was divided with the results in the following proportions: 70% training,20% validation and 10% testing. The training set is purposed at training the model to learn the patterns and relationships within this data. This helps the model to adjust its weights and parameters to minimize the loss function. The validation set helps in the tuning of the model's hyperparameters. The model does not see this data during the training and the data is used to prevent overfitting and evaluate the model by ensuring it fits well on data that it has not seen before. Finally, the testing set is used to provide an unbiased evaluation of the performance of the model on entirely new data. It is performed in two steps: first on the 10% of the remaining data that was not used from the initial dataset (Days 1 to 7) and second on the data gathered on Day 8. A key step before the division of the data set is the shuffling of the samples. This step has the advantage of providing

a better distribution across the dataset, avoiding bias in training, improving model generalization, and reducing the impact of outliers and noise.

Table 4.1 CNN model architecture and hyperparameters

Layer	Output shape	Number of Parameters
Con2D, 16 kernels	(None, 99, 884, 16)	416
Batch normalization	(None, 99, 884, 16)	64
Max pooling 2D	(None, 49, 442, 16)	0
Dropout	(None, 49, 442, 16)	0
Con2D. 32 kernels	(None, 45, 438, 32)	12832
Batch normalization	(None, 45, 438, 32)	128
Max pooling 2D	(None, 22, 219, 32)	0
Dropout	(None, 22, 219, 32)	0
Flatten	(None, 154176)	0
Dense	(None, 128)	19734656
Batch normalization	(None, 128)	512
Dropout	(None, 128)	0
Dense	(None, 64)	8256
Batch normalization	(None, 64)	256
Dropout	(None, 54)	0
Dense	(None, 3)	195

From Table 4.1, the convolutional layers contain 13,376 parameters, keeping them lightweight for feature extraction with down-sampling through max pooling, which helps computational efficiency. However, the fully connected layers contribute to the bulk of the model's 19.8 million parameters, largely due to the flattened vector feeding into a dense layer with 128 units. This configuration allows the model to capture rich feature representations at the cost of memory and computational demand, mainly due to the fully connected layers after flattening. That is why for this research a computer with 32GB RAM and an Intel(R) Core (TM) i9-14900, 2.00 GHz was used in addition to an NVIDIA RTX 4500 Ada Generation graphic card with 24GB GPU memory. The average time for model building is 2 hours, but once the model is built, the testing time is 71 seconds.

4.3 Comparison

The methodology outlined in 4.1 and 4.2 is compared to LSTM summarized in Table 4.2 to asses its performance, strengths and weaknesses and/or suitability. The input to the LSTM network as illustrated in Figure 4.5 is similar to the input used for the CNN with the first LSTM layer consisting of 64 units. This layer is designed to return sequences, allowing the next LSTM layer to process the sequential data as well. After this, a batch normalization operation is applied to normalize the activations and improve convergence. A dropout of 0.25 is also introduced to regularize the network

by randomly deactivating 25% of the units during training, which helps to prevent overfitting. Next comes a second LSTM layer with 64 units, which does not return sequences as it feeds into fully connected layers instead. Like the previous LSTM layer, batch normalization and dropout (with a rate of 0.25) follow this layer to ensure smooth training and mitigate overfitting risks. After the LSTM layers, the network transitions into the fully connected (dense) layers. The first dense layer has 128 units with a ReLU activation function. It is followed by batch normalization and a higher dropout rate of 0.5, to provide stronger regularization as the network narrows down the features. The second dense layer is similar, consisting of 64 units with a ReLU activation function. As before, batch normalization and a dropout rate of 0.5 are applied. Finally, the model concludes with an output dense layer, where the number of units corresponds to the number of output classes.

Table 4.2 LSTM model architecture and hyperparameters

Layer	Output shape	Number of Parameters
LSTM, 64 units	(None, 103, 64)	243968
Batch normalization	(None, 103, 64)	256
Dropout	(None, 103, 64)	0
LSTM. 64 units	(None, 64)	33024
Batch normalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 128)	8320
Batch normalization	(None, 128)	512
Dropout	(None, 128)	0
Dense	(None, 64)	8256
Batch normalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 3)	195

Chapter 5: Results and Discussions

The section is decomposed into three sections. The first section discusses the results of both load cases (700 rpm at 15% load and 1000 rpm at 25% load) over the initial dataset which is composed of the data acquired from day 1 to day 7. On the other hand, the second section discusses the results of both load cases over the data acquired on day 8, which simulates an operating environment with perturbations in data quality. The third section discusses the results obtained in response to the challenges observed in the second section.

5.1 Initial Dataset

The dataset was divided into training, validation and testing sets, respectively representing 70%, 20% and 10% of the initial dataset. The training was performed over 30 epochs and the hyperparameters were fine-tuned using the validation set. The loss and accuracy curves for the 700rpm load case are shown in Figure 5.1. The training loss curve remained flat throughout the epochs, indicating that the model was fitting the training data well. The validation loss was initially low but suddenly spiked near the end of the training. The validation accuracy mirrors the validation loss as it was initially high and spiked near the end. The sudden spike is an indication of overfitting and early stopping was implemented in the model-building process to overcome that effect as demonstrated in the new loss and accuracy curves for training and validation shown in Figure 5.2.

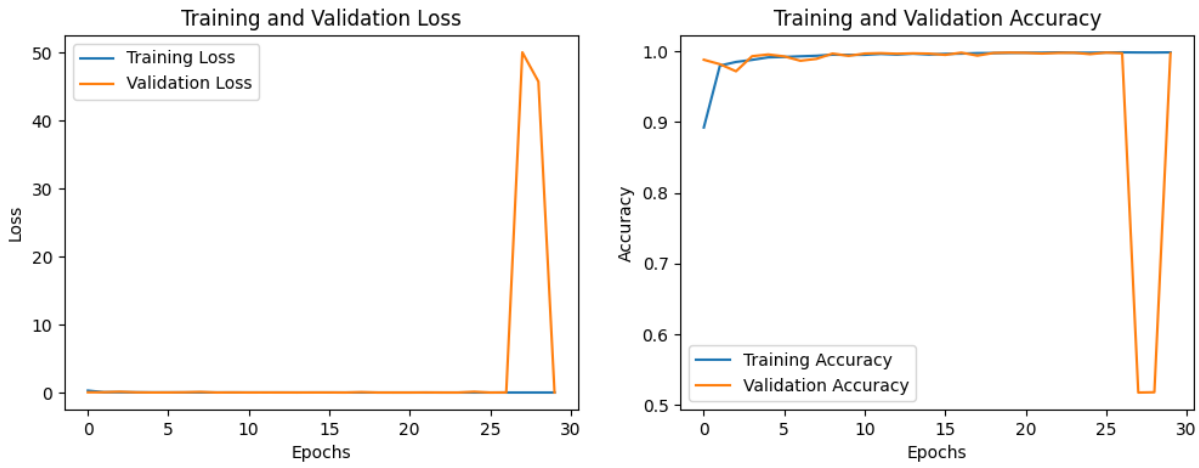


Figure 5.1 Initial loss and Accuracy curves for the 700rpm load case

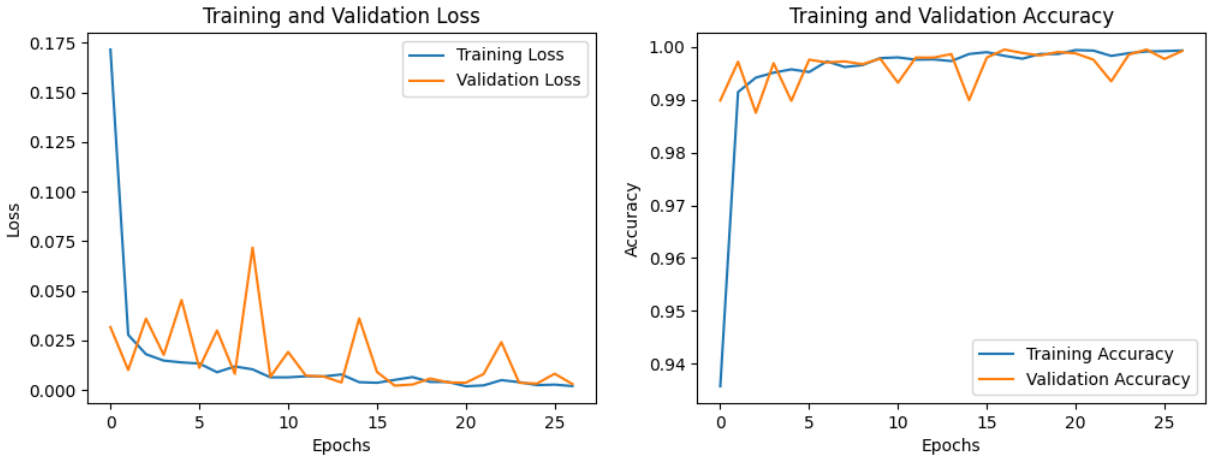


Figure 5.2 New loss and accuracy curves with early stopping for the 700rpm load case

The evaluation of a classification problem is usually done using a confusion matrix, which is a cross-tabulation used to describe the performance of a model on a set of data for which true values are known. It helps to understand which classes were

confused by the model, by showing for each class how many predictions were correct and incorrect. Furthermore, other metrics derived from the confusion matrix can be used to evaluate the performance of pattern recognition applications. These metrics are precision, and recall, which provide a score between 0 and 1 for a set of predicted items P with respect to a ground truth G. Set theory as shown in Figure 5.3 can be used to define precision and recall. Recall is the ratio between the number of correct results and the number of expected results (Equation 5.1) while precision is the ratio between the number of correct results to the number of all results (Equation 5.2). Finally, the F-1 score helps to understand the trade-off between recall and precision via a harmonic mean (Equation 5.3). The confusion matrix of the model on the testing set is shown in Figure 5.4 and the overall performance of the model is summarized in Table 5.1

$$Recall = \frac{|G \cap P|}{|G|} \tag{5.1}$$

$$Precision = \frac{|G \cap P|}{|P|} \tag{5.2}$$

$$F1 - score = 2\frac{|G \cap P|}{|G| + |P|} \tag{5.3}$$

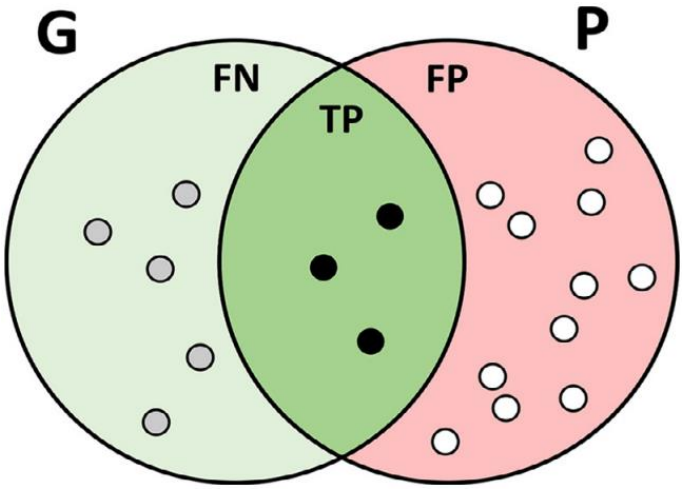


Figure 5.3 Set theory illustrating the ground truth (G), predicted results (P), True positives (TP), false positives (FP) and false negatives (FN) [85]

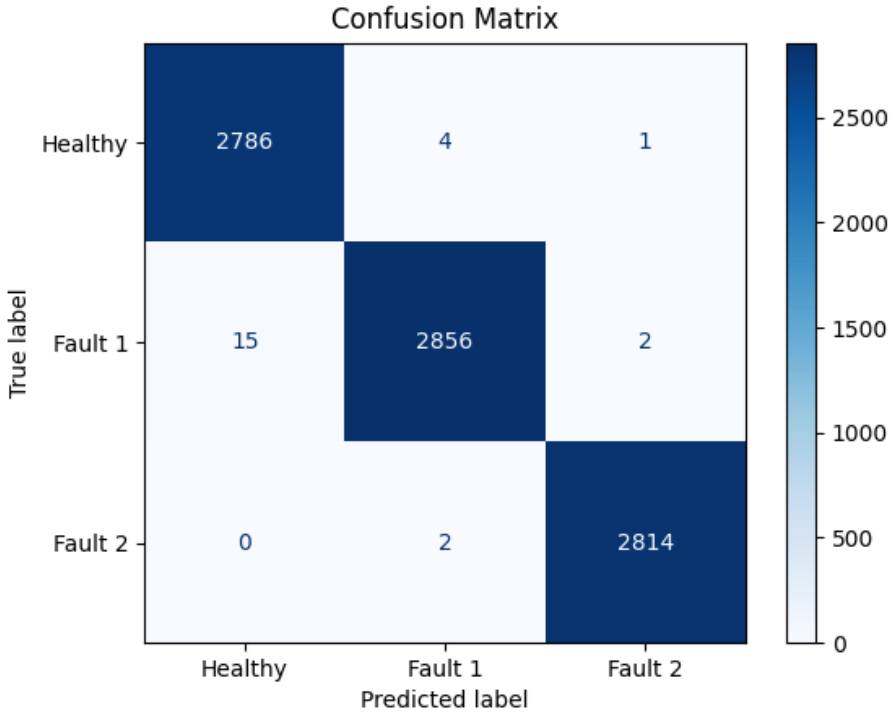


Figure 5.4 Confusion matrix for the 700rpm load case

Table 5.1 700rpm load case model performance

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.99	0.99	1.00	1.00
Fault 1	0.99	1.00	0.99	1.00
Fault 2	0.99	1.00	1.00	1.00
	Model Accuracy on Testing set (Days 1-7) = 0.99			

Despite the spike observed on the validation loss and accuracy curves, the model performed very well. From Table 5.1, the following observations were made:

- **Accuracy**: The proportion of true results (both true positives and true negatives) among the total number of cases examined. In this case, the model had an overall accuracy of 0.99, meaning it correctly classified 99% of the instances.
- **Precision**: The proportion of positive identifications that were actually correct. For Faults 1 and 2, the precision was 1.00, indicating that when the model predicted these faults, it was always right. For the Healthy category, it was high (0.99), but not perfect.
- **Recall**: The proportion of actual positives that were correctly identified by the model. For Fault 1 and Fault 2, the recall was 0.99 and 1.00, respectively, meaning the model identified 99% and 100% of these faults when they were

present. For the Healthy category, the recall was 1.00, indicating it identified all healthy cases correctly.

- **F1-score**: All F1-scores were 1.00, indicating perfect balance and performance across all categories

The loss and accuracy curves of the 1000 rpm load case are shown in Figure 5.5. The training loss started high and decreased steadily, indicating that the model was learning and improving over time. Both training and validation losses decreased over time, suggesting that the model was learning well. On the other side, the training accuracy quickly reached nearly 100% and remained stable. The validation accuracy was a mirror image of the validation loss, as the validation accuracy also increased rapidly and remained close to 100% with some fluctuations. The confusion matrix of the model on the testing set is shown in Figure 5.6and the overall performance of the model is summarized in Table 5.2.

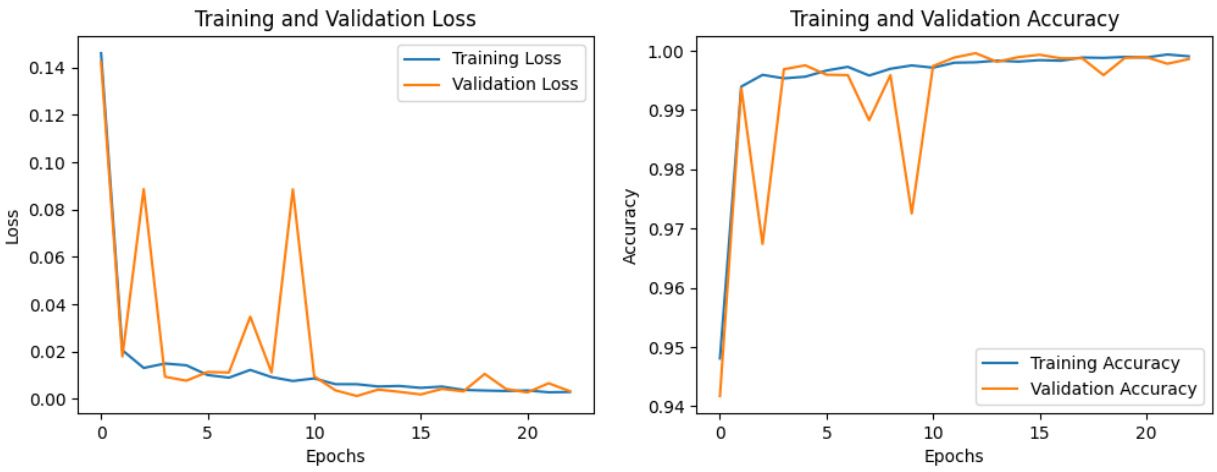


Figure 5.5 Loss and Accuracy curves for the 1000rpm load case

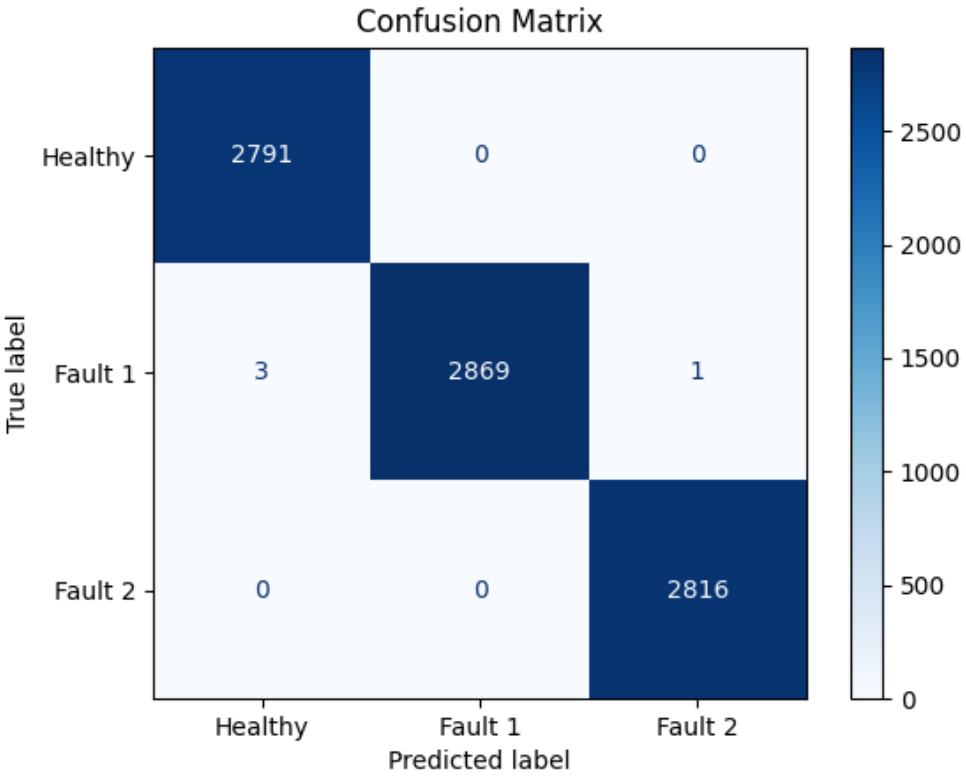


Figure 5.6 Confusion matrix for the 1000rpm load case

Table 5.2 1000rpm load case model performance

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	1.0	1.0	1.0	1.0
Fault 1	0.99	1.0	1.0	1.0
Fault 2	1.0	1.0	1.0	1.0
	Model Accuracy on Testing set (Days 1-7) = 0.99			

From Table 5.2, the following observations were made:

- **Accuracy**: The model had an overall accuracy of 0.99, meaning it correctly classified 99% of the instances in the testing set. This indicates the model is highly reliable in making predictions across all categories
- **Precision**: For all categories (Healthy, Fault 1, and Fault 2), the precision was 1.0, meaning that whenever the model predicted any of these categories, it was always correct. There were no false positives, indicating perfect precision.
- **Recall**: For Healthy, Fault 1, and Fault 2, recall is 1.0, indicating the model successfully identified 100% of the actual instances in these categories. This means there were no false negatives, and the model never missed any instances of these categories.
- **F1-score**: All F1-scores are 1.0, indicating a perfect balance and performance across all categories. The model achieved the ideal combination of precision and recall, signifying optimal performance.

5.1.1 Comparison study on the initial dataset

The performance of the proposed methodology is compared to the LSTM model whose confusion matrix is shown in Figure 5.7 and overall performance is summarized in Table 5.3 for the 700rpm load case. Figure 5.8 and Table 5.4 are respectively for the 1000rpm load case. The performance of the LSTM models was similar to that of the CNN models for the 700rpm and 1000rpm load cases with performance indicators having an overall score of 99% approximately.

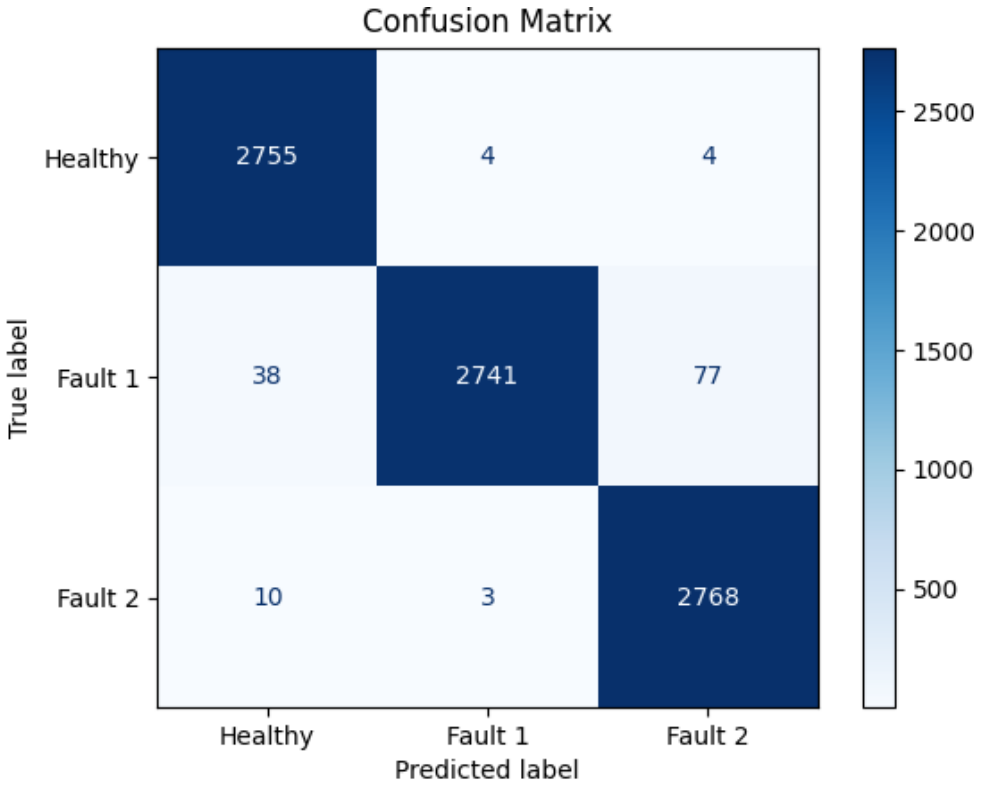


Figure 5.7 Confusion matrix for the700rpm load case with LSTM

Table 5.3 700rpm load case model performance with LSTM

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	1.00	0.98	1.00	0.99
Fault 1	0.96	1.00	0.96	0.99
Fault 2	1.00	0.97	1.00	0.98
	Model Accuracy on Testing set (Days 1-7) = 0.98			

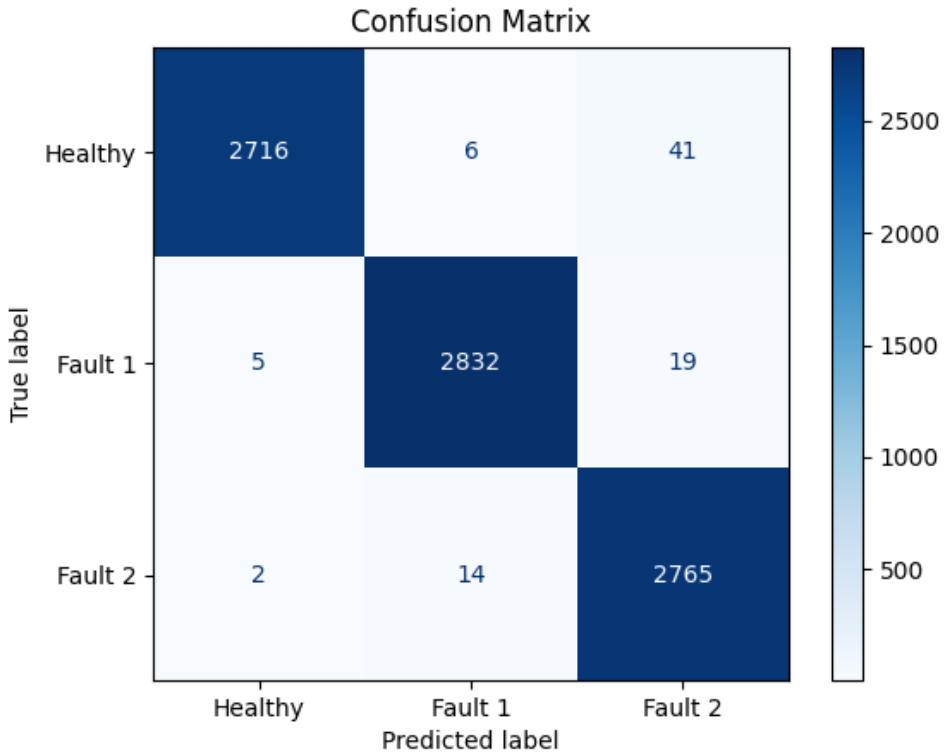


Figure 5.8 Confusion matrix for the1000rpm load case with LSTM

Table 5.4 1000rpm load case model performance with LSTM

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.98	1.00	0.98	0.99
Fault 1	0.99	0.99	0.99	0.99
Fault 2	0.99	0.98	0.99	0.99
	Model Accuracy on Testing set (Days 1-7) = 0.99			

5.2 Model Evaluation on Day 8

The high accuracies experienced in sections 5.1 and 5.1.1 were because the data split for training, validation and testing used all of the days from the initial dataset (that is days 1 to 7). Unlike the initial data set where samples from all 7 days were shuffled and split into training, validation and testing sets for generalization, the data acquired over Day 8 was not used for the model building. Hence Day 8 data simulates an operating environment with perturbations in data quality and can help in generalization. For the 700rpm load case, the confusion matrix of the model is shown in Figure 5.9, while its overall performance is summarized in Table 5.5

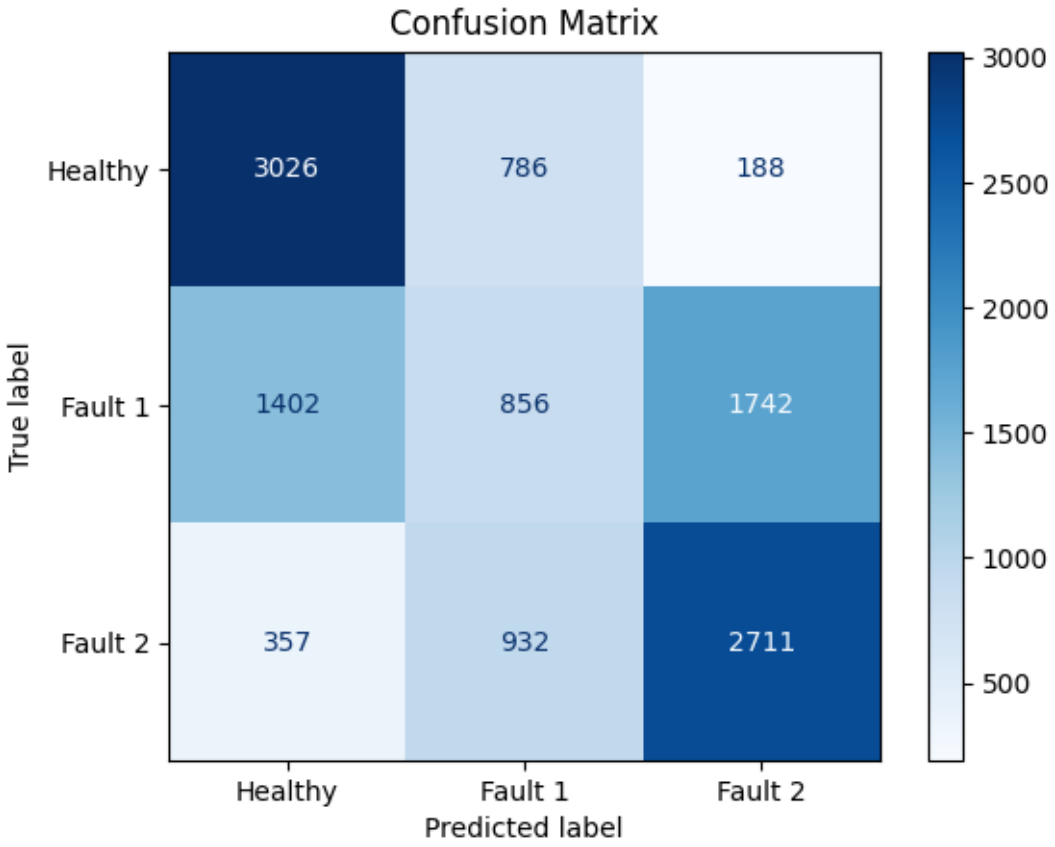


Figure 5.9 Confusion matrix for the 700rpm load case on Day 8

Table 5.5 700rpm load case model performance on Day 8

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.76	0.63	0.76	0.69
Fault 1	0.21	0.33	0.21	0.26
Fault 2	0.68	0.58	0.68	0.63
	Model Accuracy on Day 8 = 0.55			

From Table 5.5 the following observations can be made:

- **Healthy**
    - Accuracy: 76% of the Healthy instances were correctly identified. This suggests the model did a decent job of recognizing Healthy instances
    - Precision: Of all instances predicted as Healthy, only 63% were actually Healthy. This indicates that the model had a moderate level of false positives, where it incorrectly classified other categories as Healthy.
    - Recall: The model identified 76% of the actual Healthy cases, meaning it missed 24% of Healthy instances. While not poor, it suggests that some Healthy cases were incorrectly classified as faults.
    - F1-score: The F1-score of 0.69 reflects a balance between precision and recall but suggests that the model's performance in accurately and consistently identifying Healthy instances was only moderate.
- **Fault 1**
    - Accuracy: Only 21% of the instances predicted within the Fault 1 category are correctly classified, which indicates very poor performance. The model struggled significantly to identify Fault 1 accurately.
    - Precision: When the model predicted Fault 1, it was correct only 33% of the time. This is quite low, indicating a high rate of false positives where the model incorrectly classifies other categories as Fault 1.

  - Recall: The model only detected 21% of actual Fault 1 cases, meaning it missed nearly 80% of them. This very low recall shows that the model failed to identify most Fault 1 instances, leading to significant under-detection.
  - F1-score: The F1-score of 0.26 reflects the severe imbalance between poor precision and low recall, indicating that the model was not effective at detecting or correctly classifying Fault 1 cases.

- **Fault 2**
  - Accuracy: 68% of instances in the Fault 2 category were correctly classified, indicating a moderate level of performance. The model was somewhat effective at recognizing Fault 2 instances but could be better.
  - Precision: Of all instances predicted as Fault 2, only 58% were actually Fault 2. This suggests a moderate rate of false positives, where the model incorrectly classified other categories as Fault 2.
  - Recall: The model correctly identified 68% of the actual Fault 2 cases, showing that while it could detect most Fault 2 cases, it still missed about 32% of them.
  - F1-score: The F1-score of 0.63 indicates a moderate balance between precision and recall, with performance being better than for Fault 1 but still not optimal, especially in terms of precision**.**

The model had a lot of difficulties especially in detecting and classifying Faults 1 and 2, with Fault 1 being particularly challenging. This could be explained by the fact that in this load case, the engine is almost idling, meaning that the combustion dynamics and the vibrational signatures are very mild, making it very difficult for an artificial intelligence model to perform correctly.

For the 1000rpm load case, the confusion matrix of the model is shown in Figure 5.10 while its overall performance is summarized in Table 5.6.

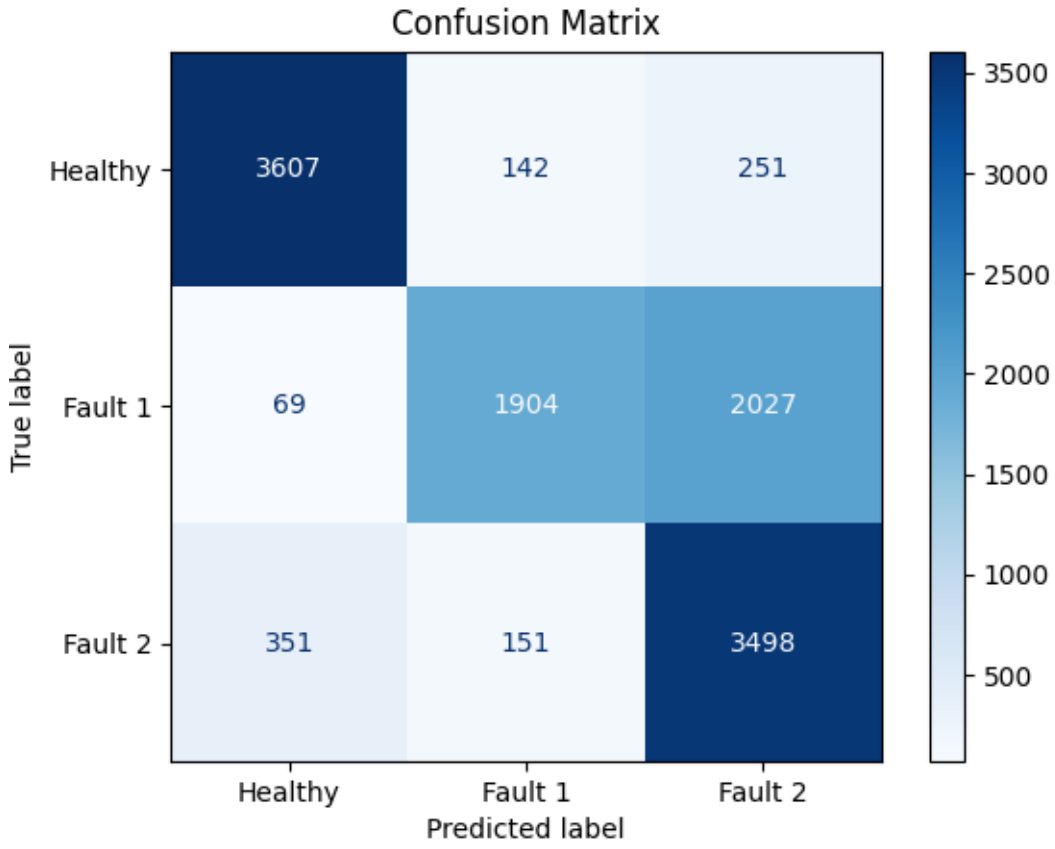


Figure 5.10 Confusion matrix for the 1000rpm load case on Day 8

Table 5.6 1000rpm load case model performance on Day 8

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.90	0.90	0.90	0.90
Fault 1	0.48	0.87	0.48	0.61
Fault 2	0.87	0.61	0.87	0.72
	Model Accuracy on Day 8 = 0.75			

From Table 5.6, the following observations were made:

- **Healthy**
    - Accuracy: 90% of the Healthy instances were correctly identified.
    - Precision: Of all instances predicted as Healthy, 90% were actually Healthy. This indicates good performance in avoiding false positives.
    - Recall: The model identified 90% of the actual Healthy cases. This shows that it effectively detected most Healthy cases.
    - F1-score: The balance between precision and recall for the Healthy category was good, reflected in the F1-score of 0.90.
    - The model performed well, with high accuracy, precision, recall, and F1-score.

- **Fault 1**
    - Accuracy**:** This suggests that only 48% of the instances predicted within the Fault 1 category were correctly classified, which indicates poor performance.
    - Precision: When the model predicted Fault 1, it was correct 87% of the time. This was fairly high, indicating that there were relatively few false positives for Fault 1.
    - Recall**:** The model only detected 48% of actual Fault 1 cases, meaning it missed more than half of them (low recall).
    - F1-score**:** The lower F1-score of 61% reflected the imbalance between good precision and poor recall, indicating that the model struggled to detect Fault 1 accurately.
    - The model struggles significantly with recall, meaning it failed to detect a substantial number of Fault 1 cases, despite good precision.
- **Fault 2**
    - Accuracy: This indicates that 87% of instances in the Fault 2 category were correctly classified, showing relatively good performance.
    - Precision: Of all instances predicted as Fault 2, only 61% were actually Fault 2, suggesting a moderate level of false positives.
    - Recall: The model correctly identified 87% of the actual Fault 2 cases, showing strong performance in detecting Fault 2.

- F1-score: This score of 72% indicated that the model had a reasonable balance between precision and recall for Fault 2, but there is room for improvement, particularly in precision.
- The model was good at detecting Fault 2 but struggled with precision, leading to some false positives. The F1-score indicates a better balance here than for Fault 1, but precision could still be improved.

The model performed reasonably well overall, its performance on Fault 1 needs improvement to ensure better detection and fewer missed cases. The difficulty in detecting and classifying Fault1 could also be explained by a higher change in the distribution of the samples compared to the samples used in the training phase. Most importantly, the model is capable of detection, concerning the Healthy category. This way, it is highly probable that a sample not categorized as healthy is a fault.

5.2.1 Comparison Study on Day 8

The performance of the proposed methodology is compared to the LSTM model whose confusion matrix is shown in Figure 5.11 and overall performance is summarized in Table 5.7 for the 700rpm load case. Figure 5.12 and Table 5.8 are respectively for the 1000rpm load case.

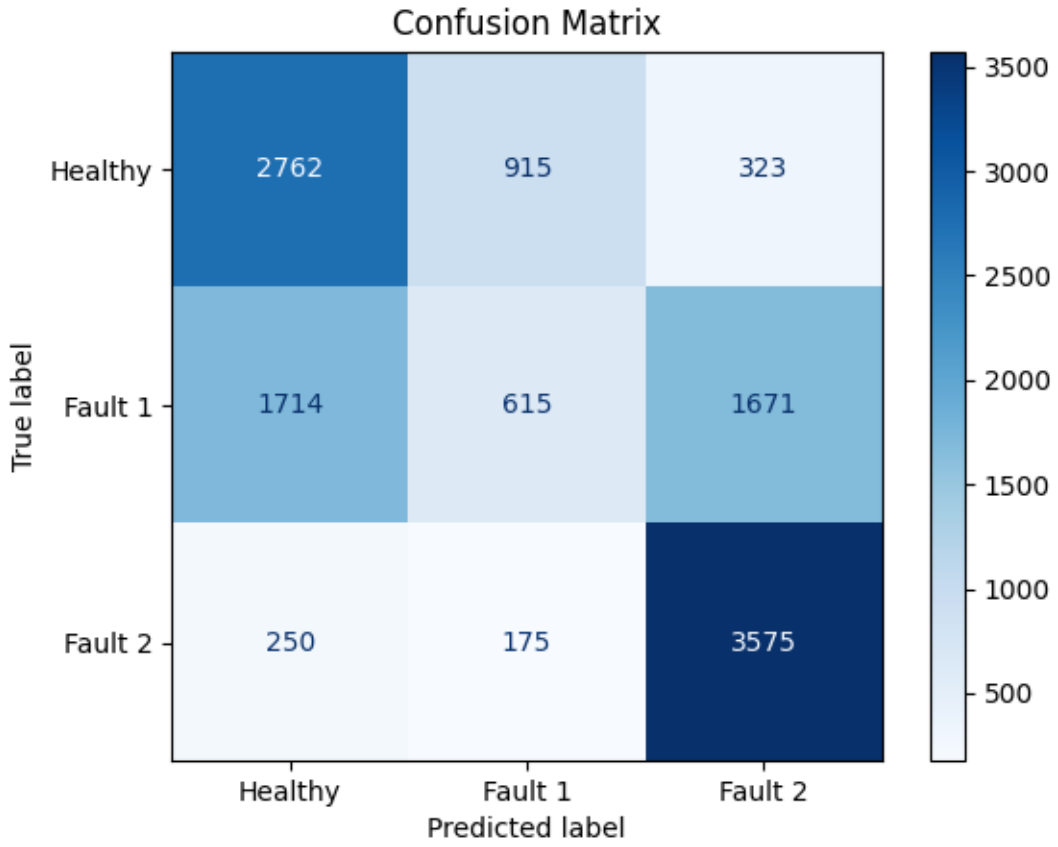


Figure 5.11 700rpm load case confusion matrix on Day 8 with LSTM

Table 5.7 700rpm load case performance on Day 8 with LSTM

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.69	0.58	0.69	0.63
Fault 1	0.15	0.36	0.15	0.22
Fault 2	0.89	0.64	0.89	0.75
	Model Accuracy on Day 8 = 0.58			

Comparing the two methods for the 700rpm load case, the following key observations could be made:

- **Healthy**
    - CNN had better accuracy (0.76) and recall (0.76) than LSTM, meaning it was slightly better at identifying healthy data
    - Both models showed fairly balanced precision and recall, with the CNN slightly outperforming in every metric for this category
- **Fault 1**
    - Both models struggled with Fault 1 but the CNN performed better in accuracy, recall, and F1-score
    - LSTM had a higher precision (0.36 vs. 0.33), indicating fewer false positives but still struggled overall with this fault category
- **Fault 2**:
    - The LSTM dominated in this category, with a significant improvement in accuracy (0.89 vs. 0.68), recall (0.89 vs. 0.68), and F1-score (0.75 vs. 0.63) compared to the CNN.
    - This suggested the LSTM model was far better at identifying and classifying Fault 2 instances.

Although the LSTM had a higher overall accuracy of 58% compared to 55% for the CNN, the CNN however offered a more balanced performance across the various health states.

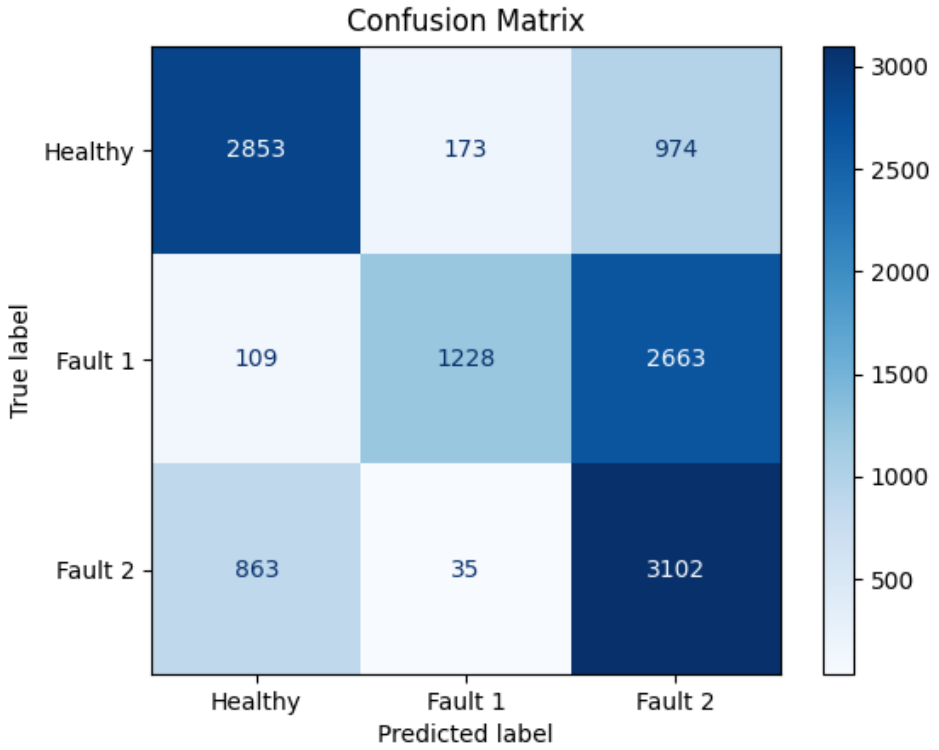


Figure 5.12 1000rpm load case confusion matrix on Day 8 with LSTM

Table 5.8 1000rpm load case performance on Day 8 with LSTM

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.71	0.75	0.71	0.73
Fault 1	0.31	0.86	0.31	0.45
Fault 2	0.78	0.46	0.78	0.58
	Model Accuracy on Day 8 = 0.60			

Comparing the two methods for the 1000rpm load case, the following key observations could be made:

- **Healthy**
    - CNN significantly outperformed LSTM in all metrics for the healthy category, with 0.90 accuracy, precision, recall, and F1-score
    - LSTM achieved reasonably good performance, but CNN demonstrated substantively better classification of healthy instances, making it superior in this regard
- **Fault 1**
    - CNN again outperformed LSTM, particularly in accuracy (0.48 vs. 0.31) and F1-score (0.61 vs. 0.45).
    - Both models exhibited high precision, with LSTM at 0.86 and CNN at 0.87, meaning both models were adept at minimizing false positives for this fault. However, CNN showed better recall (0.48 vs. 0.31), meaning it captured more of the true Fault 1 instances
- **Fault 2**
    - CNN achieved better overall accuracy (0.87 vs. 0.78) and F1-score (0.72 vs. 0.58) compared to LSTM.
    - LSTM was outperformed by CNN in recall (0.78 vs. 0.87) meaning the CNN captured more true Fault 2 instances

- The CNN showed superior precision (0.61 vs. 0.46), reducing false positives more effectively in the Fault 2 classification

With an overall accuracy of 75%, compared to 60% for LSTM, CNN demonstrated superior performance. It should be noted that even though CNN demonstrated its superiority in front of LSTM, both models did not perform very well on Day 8. To circumvent this shortcoming, another approach could be used to improve the classification accuracy across all categories. This will be discussed in the next section, in an attempt to improve the performance of the proposed methodology since it provided better results.

5.3 Transfer Learning Solution

The data collected from days 1 to 7 can be categorized as the source domain, while the data from Day 8 forms the target domain. Traditional (FDD) methods assume that the data remains independent and identically distributed (i.i.d.), which limits their efficiency when the data distribution changes. Since the target domain has a different distribution, it represents an environment where the quality of the data fluctuates compared to the source domain. From the perspective of conventional FDD strategies, the target domain appears as though it comes from a different engine. This is due to the day-to-day variations in internal combustion engine performance, even under identical load conditions. Transfer learning comes in an attempt to solve the problem by reducing the data distribution between the source and target domains

(day1-7 and day 8 respectively), so that it can pull out useful knowledge from the source domain and apply it to the target domain for improved classification performance in the target domain. It should also be noted that the source domain contains abundant data while the target domain contains few data. During the constitution of the engine data set as discussed in 3.3, the target domain was entirely labelled and as such, only a small portion of it will be used for knowledge sharing in the transfer learning as illustrated in Figure 5.13.

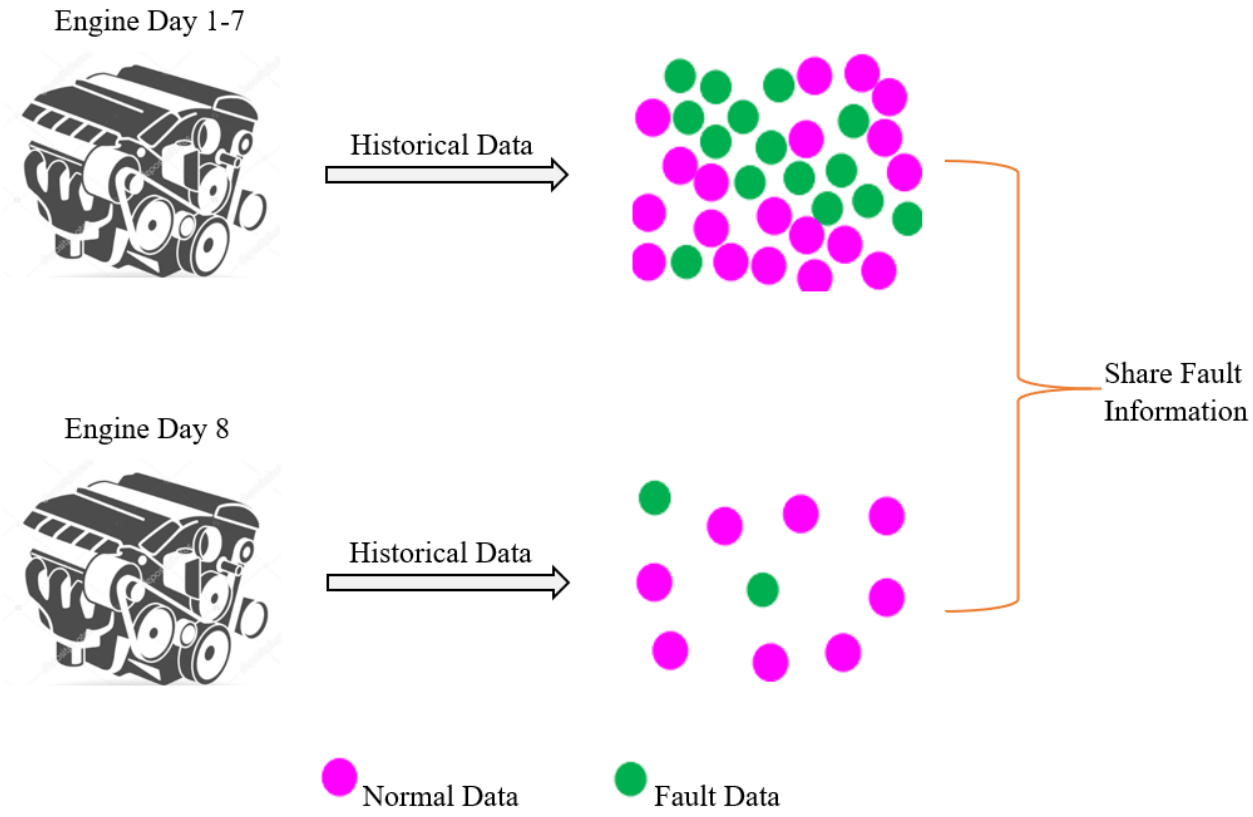


Figure 5.13 Fault knowledge sharing

Transfer learning can be defined as follows: Given a learning task T_s derived from the source domain D_s and a learning task T_t derived from the target domain D_t, transfer learning seeks to enhance the performance of the predictive function $f_T(\cdot)$ for the learning task T_t using the knowledge extracted from D_s and T_s, where $D_s \neq D_t$ and/or $T_s \neq T_t$. In addition, the size of D_s is significantly larger than the size of D_t[86]. In this research, the concept of transfer learning was combined with deep learning which has become a major breakthrough in the field of machine learning. The combination of the two is called deep transfer learning and is defined as follows: A transfer learning task defined by $\left(D_s, T_s, D_t, T_t, f_T(\cdot)\right)$ is a deep transfer learning task, where $f_T(\cdot)$ is a non-linear function that indicated a deep neural network. The deep transfer learning FDD solution illustrated in Figure 5.14 is divided into two main steps:

- Step 1: CNN model pre-train. In this step, the labelled engine data from the source domain is used to pre-train a CNN. This corresponds to the model generated in 5.1 for the two load cases.
- Step 2: CNN pre-train model finetuning. In this step, the parameters of the first several layers of the pre-trained CNN network (the first nine layers in the models generated for this research) are frozen and labelled data from the target domain (with few historical data) is used to finetune the parameters of the remaining layers of the pre-trained CNN network.

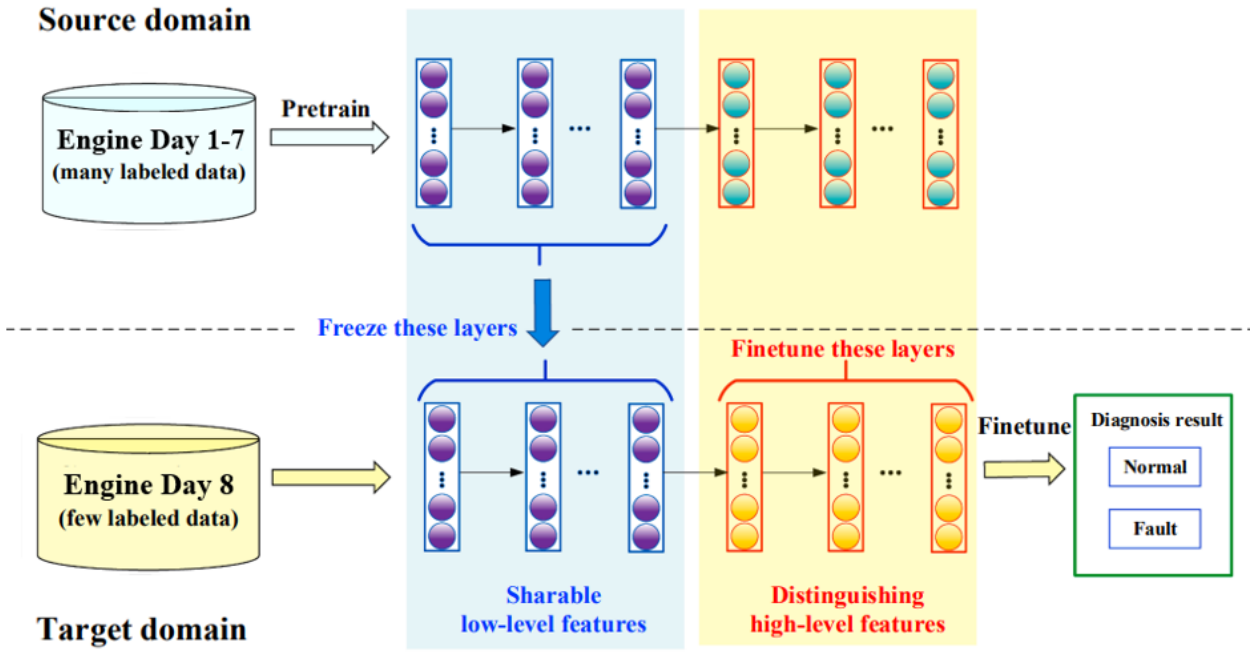


Figure 5.14 Architecture of the proposed solution

5.3.1 Using 5% of the Day 8

For both pre-trained networks corresponding each to a different load case, the last 6 layers were finetuned for just 10 epochs, using only 5% of the dataset acquired on day 8 for training (that is 600 out of 12000 samples for the 700rpm load case and 600 out of 12000 samples for the 1000rpm load case). The performance of the transfer learning was then evaluated on the rest of the dataset. The confusion matrix and model performance summary of the 700rpm load case are shown in Figure 5.15 and Table 5.9 respectively, and the confusion matrix and model performance summary of the 1000rpm load case are shown in Figure 5.16 and Table 5.10.

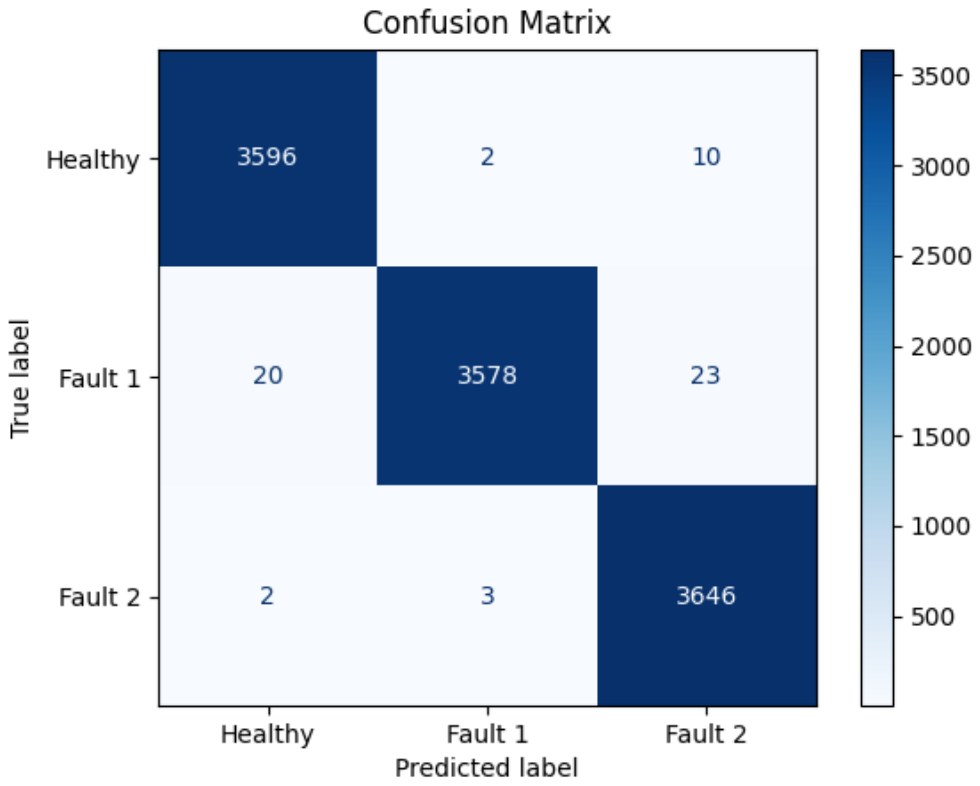


Figure 5.15 700rpm load case transfer learning model confusion matrix

Table 5.9 700rpm load case transfer learning model performance on Day 8

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.99	0.99	1.00	1.00
Fault 1	0.98	1.00	0.99	0.99
Fault 2	0.99	0.99	1.00	0.99
	Model Accuracy on Day 8 = 0.99			

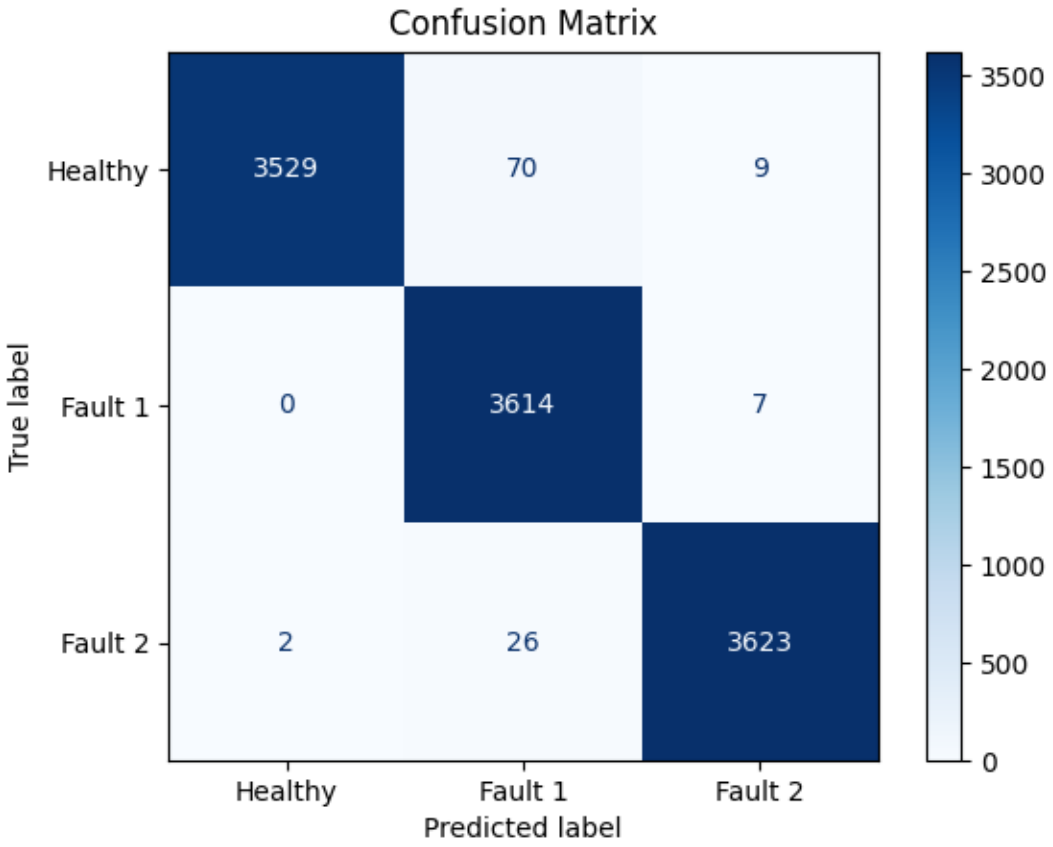


Figure 5.16 1000rpm load case transfer learning model confusion matrix

Table 5.10 1000rpm load case transfer learning model performance on Day 8

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.97	1.00	0.98	0.99
Fault 1	0.99	0.97	1.00	0.99
Fault 2	0.99	1.00	0.99	0.99
	Model Accuracy on Day 8 = 0.98			

The transfer learning solution provided better overall performance for all categories (Healthy, Fault 1 and Fault 2). It achieved on Day 8 an accuracy of 0.99 for the 700rpm load case compared to 0.55 for the traditional network and for the 1000rpm load case, an accuracy of 0.98 compared to 0.75 for the traditional network.

5.3.2 Morning and Afternoon Split of Day 8

Even though using only 5% of the Day 8 dataset gave excellent results, the randomness in the constitution of this 5% inserted some level of data distribution mix in a non-negligible amount. As section 3.3 of Chapter 3 mentioned, the data was gathered in the morning (AM) and the afternoon (PM), corresponding to 6000 samples each for Day 8. This indicates a difference in data distribution within the same day between the AM and PM datasets. Two scenarios were used to assess the performance of transfer learning. The first one is AM vs PM, where the AM data (split with 80% training and 20% validation) is used for finetuning and the model is tested on PM data. The second one is PM vs AM where the PM data (split with 80% training and 20% validation) is used for finetuning and the model is tested on AM data. Figure 5.17 and Table 5.11 show respectively the confusion matrix and classification report of the AM vs PM scenario for the 700rpm load case and the 1000rpm load case as represented by Figure 5.19 and Table 5.13. On the other hand, Figure 5.18 and Table 5.12 show respectively the confusion matrix and classification

report of the PM vs AM scenario for the 700rpm load case, while the 1000rom load case is represented by Figure 5.20 and Table 5.14.

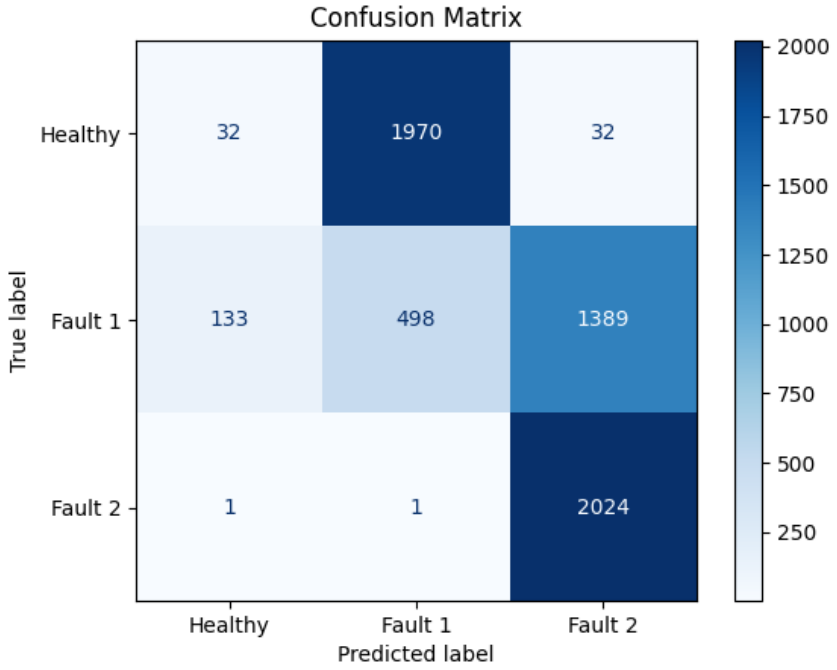


Figure 5.17 700rpm load case AM vs PM confusion matrix

Table 5.11 700rpm load case AM vs PM performance

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.02	0.19	0.02	0.03
Fault 1	0.25	0.20	0.25	0.22
Fault 2	1.00	0.59	1.00	0.74
	Model Accuracy on PM data = 0.42			

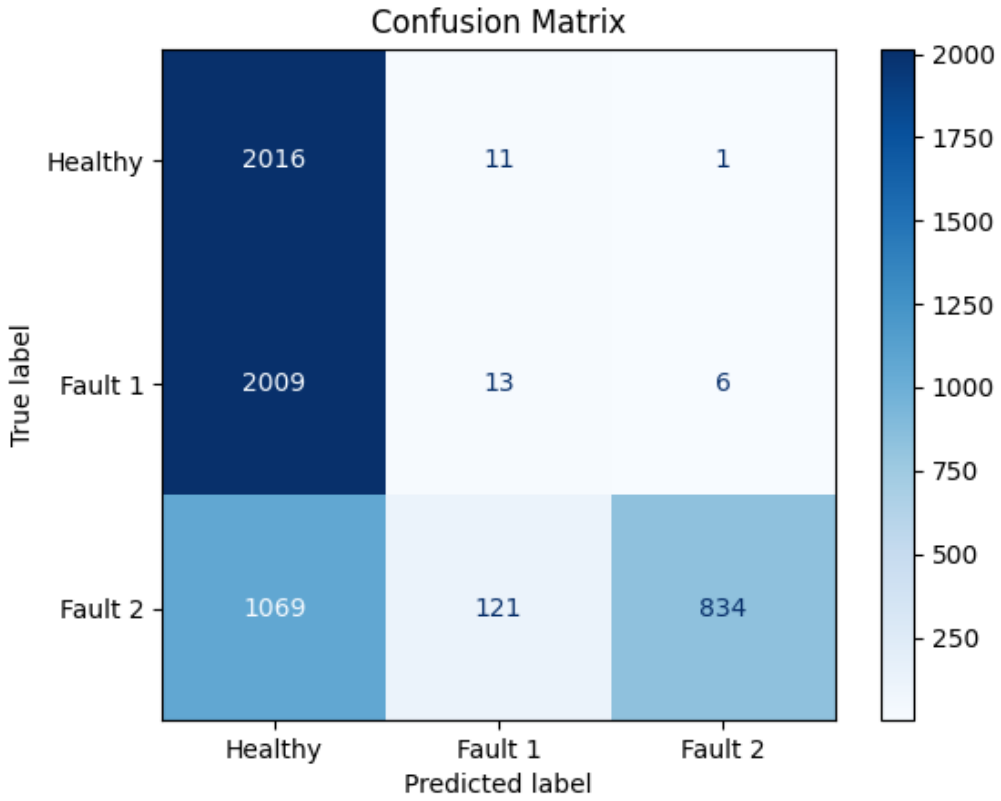


Figure 5.18 700rpm load case PM vs AM confusion matrix

Table 5.12 700rpm load case PM vs AM performance

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.99	0.40	0.99	0.57
Fault 1	0.00	0.09	0.01	0.01
Fault 2	0.41	0.99	0.41	0.58
	Model Accuracy on AM data = 0.47			

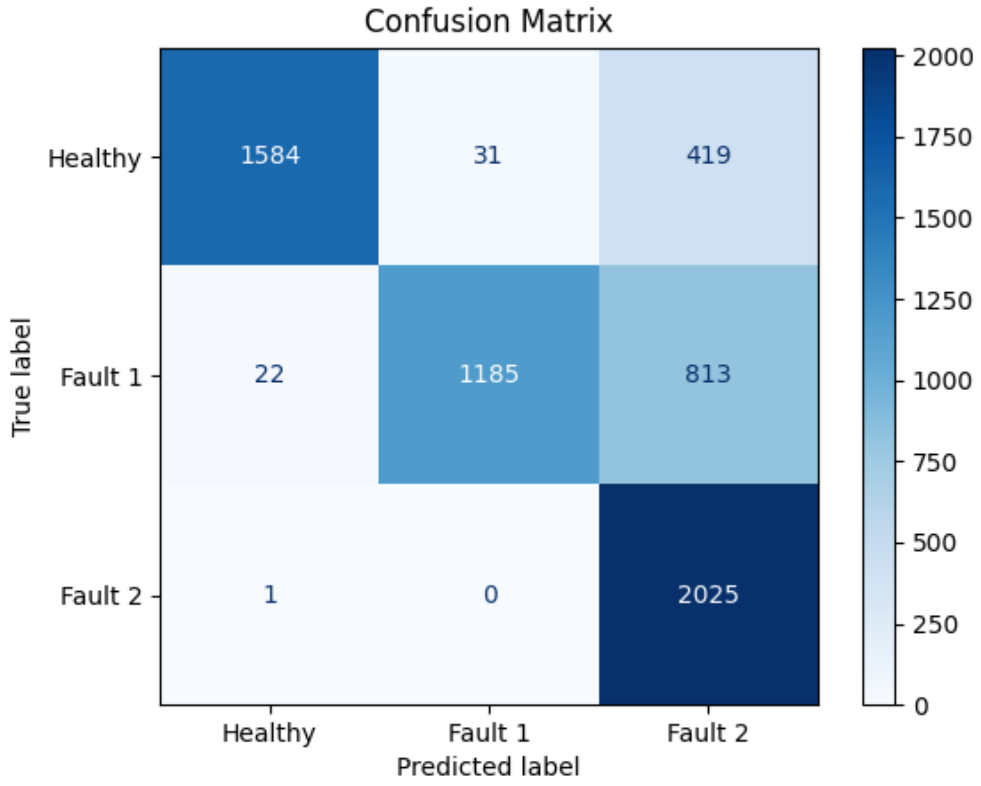


Figure 5.19 1000rpm load case AM vs PM confusion matrix

Table 5.13 1000rpm load case AM vs PM performance

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.78	0.99	0.78	0.87
Fault 1	0.59	0.97	0.59	0.73
Fault 2	0.99	0.62	1.00	0.77
	Model Accuracy on PM data = 0.78			

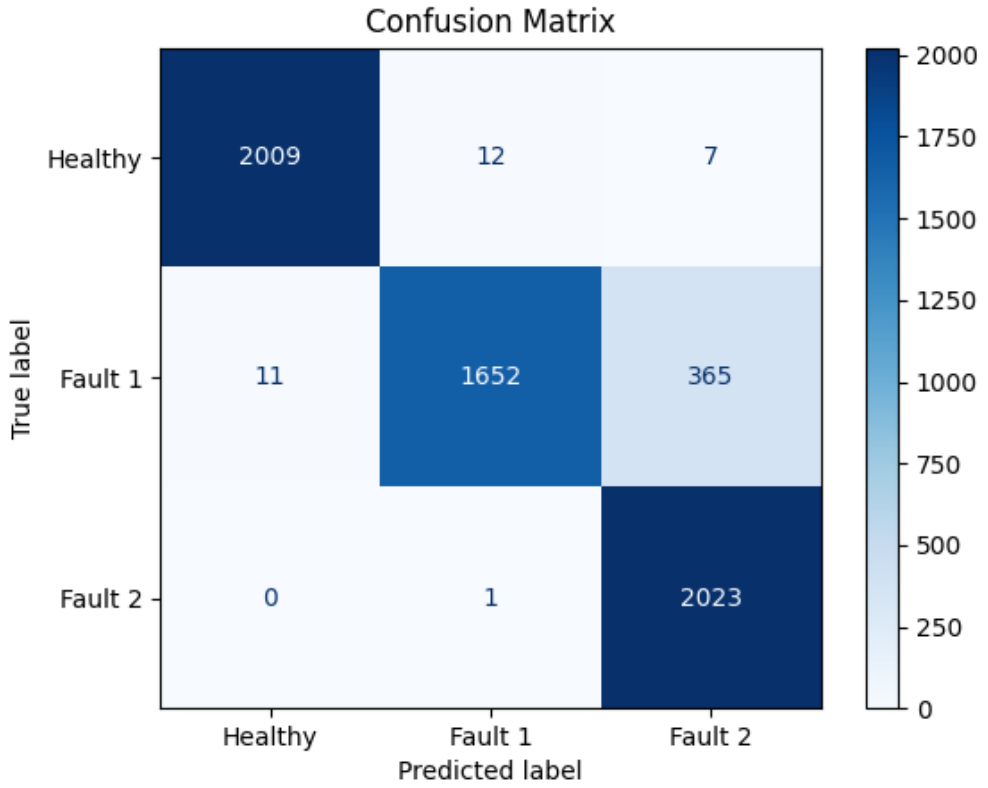


Figure 5.20 1000rpm load case PM vs AM confusion matrix

Table 5.14 1000rpm load case PM vs AM performance

Fault Categories	Performance Indicators			
	Accuracy	Precision	Recall	F1-score
Healthy	0.99	0.99	0.99	0.99
Fault 1	0.81	0.99	0.81	0.89
Fault 2	1.00	0.84	1.00	0.92
	Model Accuracy on AM data = 0.94			

The performance of the 700rpm load presented the same difficulties as its traditional counterpart, outlined in section 5.2 (which had a 55% accuracy). For the AM vs PM scenario, the accuracy was 42% and the model could at best detect Fault 2 with 100% accuracy. Regarding the PM vs AM scenario, the accuracy was 47% and the model could at best detect the Healthy category with an accuracy of 99%. The low score in the accuracy in addition to the other performance metrics indicates a high variability regarding this load case. On the other hand, the 1000rpm load case AM vs PM scenario provided slightly better results with an accuracy of 78% compared to its traditional counterpart in section 5.2 where an accuracy of 75% was obtained. The PM vs AM scenario had an accuracy of 94% which is far better than the performance discussed in section 5.2. This indicates that the PM data had a more general distribution compared to the AM data. The same trend is observed in 700rpm load cases with a smaller extent. The lower accuracy of Fault 1 for the 1000rapm load case is consistent with observations made in the previous section since it is the fault with the smallest gap.

Chapter 6: Conclusion and Future Work

The reliability of internal combustion engines is of prime importance in today's highly competitive manufacturing environment. The objective of this research is to employ deep learning for the development of a fault detection and diagnosis algorithm for internal combustion engine faults via engine block vibration. The proposed solution is data-driven, uses signals from knock sensors only, learns labelled data and accounts for perturbations in data quality.

Over the course of this research, Short Time Fourier Transform demonstrated its capacity to decipher the time-varying nature of knock sensor signals and its ability to provide useful features. STFT was coupled with Convolutional Neural Networks, which could perform fault detection and diagnosis by considering the extracted features as patterns and performing pattern recognition to solve the problem. The great capacity exhibited by CNN models is because they can automatically learn patterns in features from monitored input data and simultaneously identify health conditions based on those learned features. Using non-linear functions, these models can adaptively capture representational information from input signals and approximate complex non-linear functions with minimal error.

A large number of papers have portrayed the usefulness of CNN in the field of fault detection and diagnosis and as such, the solution proposed in this research was evaluated over experimental data from a test cell, where the signal of 4 knock sensors

mounted on a V8 type engine was acquired to constitute a dataset composed of three different spark-plug health states. The proposed solution performed very well on the initial dataset (days 1 to 7) with high accuracies over the three health states or fault categories. An interesting observation made was the difficulty the trained model had in the classification of the samples from the Day 8 dataset, with the 700rpm load case scenario providing poor results and the 1000rpm scenario providing fair results overall but most importantly capable of detection of the health category. The lower accuracies observed over the Day 8 dataset were most likely due to the phenomenon of domain shift, which is a change in the data distribution between the training set which is used to build a classification model algorithm and the dataset onto which the latter is evaluated. The 700rpm load case was particularly challenging because, at this load condition, the engine is in idling mode; a condition where the control strategy governing the engine operation induces even more variability. A comparative study was made using an LSTM-based model which faced the same challenges as the proposed solution but was outperformed by the latter. To resolve the domain shift problem, the concept of transfer learning was used to enhance the performance of the proposed solution. Hence, the model generated from the initial dataset was considered a pre-trained model. While its first few layers were frozen, its last layers were finetuned. This finetuning of the last layers is mostly because they are learning the high-level distinguishable features between domains while the first few layers learn the low-level sharable features between domains. As such, two fine-

tuning approaches were used. One consisted of finetuning the pre-trained model with 5% of the dataset used on Day 8 at random. Another approach was to split the Day 8 dataset into morning and afternoon halves and use one half for finetuning and the other half for testing in an alternating manner. The second approach was the closest to realistic conditions as it conserved a certain level of data distribution disparity in contrast to the first one where a non-negligible mix of data distribution was present. Finally, the scenario where the afternoon data was used for finetuning and the morning data for testing provided the best results with 94% accuracy, taking data distribution disparity as an essential condition for model evaluation. Another interesting observation was the increased stability of engine performance at high speeds. There was more variation in engine performance in idling mode (700rpm load case). Hence engine testing in idling conditions should be performed with careful attention. In addition, ambient air conditions affect engine outputs and thus mass airflow is a critical parameter to monitor as it linearly affects engine outputs.

6.1 Future Research

The aforementioned shift in data distribution opens a new ground for improvements. In industrial applications, a deep neural network trained on a machine with historical data may not perform well on a different machine of the same model, largely due to variations between machines and sensors. Even though the transfer learning solution adopted in this research was able to circumvent the problem via finetuning, there is still the use of labelled historical data from the target domain. The transfer learning solution could be improved by using an unbalanced dataset composed only of fault-free data. This is a condition where the finetuning is performed with a dataset that is unevenly distributed across classes. In this scenario, a pre-trained model could be finetuned with data coming from another machine that has only one class. Further improvement to the FDD solution could be done by employing domain adaptation, where the distribution discrepancy between source and target domains could be reduced. This solution would be semi-supervised as the data coming from the target domain would be unlabelled. Another direction for improvement could be domain generalization, where the model would be trained with labelled data coming from various (source) domains only. Using this concept, the model would learn domain invariant features and could be then evaluated on a target domain. These are some potential research topics which would be worth diving in. Other deep learning

strategies employing Domain Adversarial Neural Network (DANN) or Generative Adversarial Network (GAN) could be leveraged to fulfill those goals.

References

[1] E. G. Giakoumis, D. C. Rakopoulos, and C. D. Rakopoulos, "Combustion noise radiation during dynamic diesel engine operation including effects of various biofuel blends: A review," *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 1099–1113, Feb. 2016, doi: 10.1016/j.rser.2015.10.129.

[2] B. Pla, J. De la Morena, P. Bares, and A. Aramburu, "A supervised machine learning technique for combustion diagnosis using a vibration sensor signal," *Fuel*, vol. 343, p. 127869, Jul. 2023, doi: 10.1016/j.fuel.2023.127869.

[3] B. Pla, J. De la Morena, P. Bares, and A. Aramburu, "An unsupervised machine learning technique to identify knock from a knock signal time-frequency analysis," *Measurement*, vol. 211, p. 112669, Apr. 2023, doi: 10.1016/j.measurement.2023.112669.

[4] J. Sun, X. Zhang, Q. Tang, Y. Wang, and Y. Li, "Knock recognition of knock sensor signal based on wavelet transform and variational mode decomposition algorithm," *Energy Convers Manag*, vol. 287, p. 117062, Jul. 2023, doi: 10.1016/j.enconman.2023.117062.

[5] J. D. Naber and J. E. Johnson, "Internal combustion engine cycles and concepts," in *Alternative Fuels and Advanced Vehicle Technologies for Improved Environmental Performance*, Elsevier, 2014, pp. 197–224. doi: 10.1533/9780857097422.2.197.

[6] S. C. Government of Canada, "Vehicle registrations, by type of vehicle and fuel type." Accessed: Nov. 30, 2023. [Online]. Available: ttps://doi.org/10.25318/2310030801-eng

[7] B. Ashok, S. Denis Ashok, and C. Ramesh Kumar, "A review on control system architecture of a SI engine management system," *Annu Rev Control*, vol. 41, pp. 94–118, 2016, doi: 10.1016/j.arcontrol.2016.04.005.

[8] R. Isermann, *Engine Modeling and Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. doi: 10.1007/978-3-642-39934-3.

[9] Z. Wang, H. Liu, and R. D. Reitz, "Knocking combustion in spark-ignition engines," *Prog Energy Combust Sci*, vol. 61, pp. 78–112, Jul. 2017, doi: 10.1016/j.pecs.2017.03.004.

[10] J. Sun, X. Zhang, Q. Tang, Y. Wang, and Y. Li, "Knock recognition of knock sensor signal based on wavelet transform and variational mode decomposition algorithm," *Energy Convers Manag*, vol. 287, p. 117062, Jul. 2023, doi: 10.1016/j.enconman.2023.117062.

[11] B. O. Fernandez, P. R. Aguiar, F. A. Alexandre, M. A. Aulestia Viera, and E. C. Bianchi, "Study of Knock Sensors as Low-Cost Alternatives to Acoustic Emission Sensors," *IEEE Sens J*, vol. 20, no. 11, pp. 6038–6045, Jun. 2020, doi: 10.1109/JSEN.2020.2972778.

[12] W. Xi, Z. Li, Z. Tian, and Z. Duan, "A feature extraction and visualization method for fault detection of marine diesel engines," *Measurement (Lond)*, vol. 116, pp. 429–437, Feb. 2018, doi: 10.1016/j.measurement.2017.11.035.

[13] A. Taghizadeh-Alisaraei and A. Mahdavian, "Fault detection of injectors in diesel engines using vibration time-frequency analysis," *Applied Acoustics*, vol. 143, pp. 48–58, Jan. 2019, doi: 10.1016/j.apacoust.2018.09.002.

[14] R. X. Gao and R. Yan, *Wavelets: Theory and Applications for Manufacturing*. Boston, MA: Springer US, 2011. doi: 10.1007/978-1-4419-1545-0.

[15] R. Parhizkar, Y. Barbotin, and M. Vetterli, "Sequences with minimal time-frequency uncertainty," *Appl Comput Harmon Anal*, vol. 38, no. 3, pp. 452–468, May 2015, doi: 10.1016/j.acha.2014.07.001.

[16] S. Mallat, *A Wavelet Tour of Signal Processing*. Elsevier, 2009. doi: 10.1016/B978-0-12-374370-1.X0001-8.

[17] M. Ismail, "Industrial extended multi-scale principle components analysis for fault detection and diagnosis of car alternators and starters," McMaster University, Hamilton, 2015.

[18] M. Chafii, J. Palicot, and R. Gribonval, "Wavelet modulation: An alternative modulation with low energy consumption," *C R Phys*, vol. 18, no. 2, pp. 156–167, Feb. 2017, doi: 10.1016/j.crhy.2016.11.010.

[19] J. Chen *et al.*, "Wavelet transform based on inner product in fault diagnosis of rotating machinery: A review," *Mech Syst Signal Process*, vol. 70–71, pp. 1–35, Mar. 2016, doi: 10.1016/j.ymssp.2015.08.023.

[20] M. Upadhya, A. K. Singh, P. Thakur, E. A. Nagata, and D. D. Ferreira, "Mother wavelet selection method for voltage sag characterization and

detection," *Electric Power Systems Research*, vol. 211, p. 108246, Oct. 2022, doi: 10.1016/j.epsr.2022.108246.

[21] A. I. Megahed, A. Monem Moussa, H. B. Elrefaie, and Y. M. Marghany, "Selection of a suitable mother wavelet for analyzing power system fault transients," in *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, IEEE, Jul. 2008, pp. 1–7. doi: 10.1109/PES.2008.4596367.

[22] F. Hemmati, W. Orfali, and M. S. Gadala, "Roller bearing acoustic signature extraction by wavelet packet transform, applications in fault detection and size estimation," *Applied Acoustics*, vol. 104, pp. 101–118, Mar. 2016, doi: 10.1016/j.apacoust.2015.11.003.

[23] J. M. Juhani and R. Ibrahim, "Mother Wavelet Selection for Control Valve Leakage Detection using Acoustic Emission," in *2018 IEEE Conference on Systems, Process and Control (ICSPC)*, IEEE, Dec. 2018, pp. 224–227. doi: 10.1109/SPC.2018.8703973.

[24] J.-D. Wu and C.-H. Liu, "An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network," *Expert Syst Appl*, vol. 36, no. 3, pp. 4278–4286, Apr. 2009, doi: 10.1016/j.eswa.2008.03.008.

[25] A. Moosavian, M. Khazaee, G. Najafi, M. Khazaee, B. Sakhaei, and S. Mohammad Jafari, "Wavelet denoising using different mother wavelets for fault diagnosis of engine spark plug," *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, vol. 231, no. 3, pp. 359–370, Jun. 2017, doi: 10.1177/0954408915595952.

[26] K. N. Ravikumar, C. K. Madhusudana, H. Kumar, and K. V. Gangadharan, "Classification of gear faults in internal combustion (IC) engine gearbox using discrete wavelet transform features and K star algorithm," *Engineering Science and Technology, an International Journal*, vol. 30, p. 101048, Jun. 2022, doi: 10.1016/j.jestch.2021.08.005.

[27] C. Syms, "Principal Components Analysis," in *Encyclopedia of Ecology*, Elsevier, 2019, pp. 566–573. doi: 10.1016/B978-0-12-409548-9.11152-2.

[28] I. Jolliffe, "Principal Component Analysis," in *Wiley StatsRef: Statistics Reference Online*, Wiley, 2014. doi: 10.1002/9781118445112.stat06472.

[29] S. R. Haqshenas, "Multiresolution-multivariate analysis of vibration signals; application in fault diagnosis of internal combustion engines," M.A.Sc, McMaster University, Hamilton, 2012.

[30] S. Joe Qin, "Statistical process monitoring: basics and beyond," *J Chemom*, vol. 17, no. 8–9, pp. 480–502, Aug. 2003, doi: 10.1002/cem.800.

[31] H. H. Yue and S. J. Qin, "Reconstruction-Based Fault Identification Using a Combined Index," 2001, doi: 10.1021/ie000141.

[32] G. E. P. Box, "Some Theorems on Quadratic Forms Applied in the Study of Analysis of Variance Problems, I. Effect of Inequality of Variance in the One-Way Classification," *The Annals of Mathematical Statistics*, vol. 25, no. 2, pp. 290–302, Jun. 1954, doi: 10.1214/aoms/1177728786.

[33] J. E. Jackson, *A User's Guide to Principal Components*, vol. 587. Hoboken, NJ: Wiley Interscience, 2005.

[34] N. D. Tracy, J. C. Young, and R. L. Mason, "Multivariate Control Charts for Individual Observations," *Journal of Quality Technology*, vol. 24, no. 2, pp. 88–95, Apr. 1992, doi: 10.1080/00224065.1992.12015232.

[35] C. F. Alcala and S. J. Qin, "Reconstruction-based contribution for process monitoring," *Automatica*, vol. 45, no. 7, pp. 1593–1600, Jul. 2009, doi: 10.1016/j.automatica.2009.02.027.

[36] R. Dunia and S. Joe Qin, "A unified geometric approach to process and sensor fault identification and reconstruction," *Comput Chem Eng*, vol. 22, no. 7–8, pp. 927–943, Jul. 1998, doi: 10.1016/S0098-1354(97)00277-9.

[37] S. Yoon and J. F. MacGregor, "Principal-component analysis of multiscale data for process monitoring and fault diagnosis," *AIChE Journal*, vol. 50, no. 11, pp. 2891–2903, Nov. 2004, doi: 10.1002/aic.10260.

[38] S. Cao, Z. Hu, X. Luo, and H. Wang, "Research on fault diagnosis technology of centrifugal pump blade crack based on PCA and GMM," *Measurement*, vol. 173, p. 108558, Mar. 2021, doi: 10.1016/j.measurement.2020.108558.

[39] L. Hou, J. Zhang, and B. Du, "A Fault Diagnosis Model of Marine Diesel Engine Fuel Oil Supply System Using PCA and Optimized SVM," *J Phys Conf Ser*, vol. 1576, no. 1, p. 012045, Jun. 2020, doi: 10.1088/1742-6596/1576/1/012045.

[40] S. Xu, X. Jiang, J. Huang, S. Yang, and X. Wang, "Bayesian wavelet PCA methodology for turbomachinery damage diagnosis under uncertainty," *Mech Syst Signal Process*, vol. 80, pp. 1–18, Dec. 2016, doi: 10.1016/j.ymssp.2016.04.031.

[41] M. Z. Sheriff, M. Mansouri, M. N. Karim, H. Nounou, and M. Nounou, "Fault detection using multiscale PCA-based moving window GLRT," *J Process Control*, vol. 54, pp. 47–64, Jun. 2017, doi: 10.1016/j.jprocont.2017.03.004.

[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[43] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mech Syst Signal Process*, vol. 107, pp. 241–265, Jul. 2018, doi: 10.1016/j.ymssp.2017.11.024.

[44] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: A review," *Mech Syst Signal Process*, vol. 108, pp. 33–47, Aug. 2018, doi: 10.1016/j.ymssp.2018.02.016.

[45] D. Wang, "K-nearest neighbors based methods for identification of different gear crack levels under different motor speeds and loads: Revisited," *Mech Syst Signal Process*, vol. 70–71, pp. 201–208, Mar. 2016, doi: 10.1016/j.ymssp.2015.10.007.

[46] D. H. Pandya, S. H. Upadhyay, and S. P. Harsha, "Fault diagnosis of rolling element bearing with intrinsic mode function of acoustic emission data using APF-KNN," *Expert Syst Appl*, vol. 40, no. 10, pp. 4137–4145, Aug. 2013, doi: 10.1016/j.eswa.2013.01.033.

[47] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, "Applications of machine learning to machine fault diagnosis: A review and roadmap," *Mech Syst Signal Process*, vol. 138, p. 106587, Apr. 2020, doi: 10.1016/j.ymssp.2019.106587.

[48] Y. Lei, Z. He, and Y. Zi, "A Combination of WKNN to Fault Diagnosis of Rolling Element Bearings," *J Vib Acoust*, vol. 131, no. 6, Dec. 2009, doi: 10.1115/1.4000478.

[49] S. Dong, T. Luo, L. Zhong, L. Chen, and X. Xu, "Fault diagnosis of bearing based on the kernel principal component analysis and optimized *k* -nearest

neighbour model," *Journal of Low Frequency Noise, Vibration and Active Control*, vol. 36, no. 4, pp. 354–365, Dec. 2017, doi: 10.1177/1461348417744302.

[50] F. Li, J. Wang, B. Tang, and D. Tian, "Life grade recognition method based on supervised uncorrelated orthogonal locality preserving projection and K-nearest neighbor classifier," *Neurocomputing*, vol. 138, pp. 271–282, Aug. 2014, doi: 10.1016/j.neucom.2014.01.037.

[51] T. M. Mitchell, "GENERATIVE AND DISCRIMINATIVE CLASSIFIERS: NAIVE BAYES AND LOGISTIC REGRESSION," 2005. [Online]. Available: www.cs.cmu.edu/~tom/mlbook.html.

[52] J. Seshadrinath, B. Singh, and B. K. Panigrahi, "Vibration Analysis Based Interturn Fault Diagnosis in Induction Machines," *IEEE Trans Industr Inform*, vol. 10, no. 1, pp. 340–350, Feb. 2014, doi: 10.1109/TII.2013.2271979.

[53] R. H. Cunha Palácios, I. N. da Silva, A. Goedtel, and W. F. Godoy, "A comprehensive evaluation of intelligent classifiers for fault identification in three-phase induction motors," *Electric Power Systems Research*, vol. 127, pp. 249–258, Oct. 2015, doi: 10.1016/j.epsr.2015.06.008.

[54] J. Flett and G. M. Bone, "Fault detection and diagnosis of diesel engine valve trains," *Mech Syst Signal Process*, vol. 72–73, pp. 316–327, May 2016, doi: 10.1016/j.ymssp.2015.10.024.

[55] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. doi: 10.1017/CBO9780511801389.

[56] D. A. Pisner and D. M. Schnyer, "Support vector machine," in *Machine Learning*, Elsevier, 2020, pp. 101–121. doi: 10.1016/B978-0-12-815739-8.00006-7.

[57] R. Bhaumik *et al.*, "Multivariate pattern analysis strategies in detection of remitted major depressive disorder using resting state functional connectivity," *Neuroimage Clin*, vol. 16, pp. 390–398, 2017, doi: 10.1016/j.nicl.2016.02.018.

[58] X. Li, K. Wang, and L. Jiang, "The Application of AE Signal in Early Cracked Rotor Fault Diagnosis with PWVD and SVM," *Journal of Software*, vol. 6, no. 10, Oct. 2011, doi: 10.4304/jsw.6.10.1969-1976.

[59] N. Li, R. Zhou, Q. Hu, and X. Liu, "Mechanical fault diagnosis based on redundant second generation wavelet packet transform, neighborhood rough set and support vector machine," *Mech Syst Signal Process*, vol. 28, pp. 608–621, Apr. 2012, doi: 10.1016/j.ymssp.2011.10.016.

[60] Y. S. Wang, Q. H. Ma, Q. Zhu, X. T. Liu, and L. H. Zhao, "An intelligent approach for engine fault diagnosis based on Hilbert–Huang transform and support vector machine," *Applied Acoustics*, vol. 75, pp. 1–9, Jan. 2014, doi: 10.1016/j.apacoust.2013.07.001.

[61] M. Zhang, Y. Zi, L. Niu, S. Xi, and Y. Li, "Intelligent Diagnosis of V-Type Marine Diesel Engines Based on Multifeatures Extracted From Instantaneous Crankshaft Speed," *IEEE Trans Instrum Meas*, vol. 68, no. 3, pp. 722–740, Mar. 2019, doi: 10.1109/TIM.2018.2857018.

[62] Z. Li, S.-S. Zhong, and L. Lin, "Novel Gas Turbine Fault Diagnosis Method Based on Performance Deviation Model," *J Propuls Power*, vol. 33, no. 3, pp. 730–739, May 2017, doi: 10.2514/1.B36267.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[64] A. Zabihi-Hesari, S. Ansari-Rad, F. A. Shirazi, and M. Ayati, "Fault detection and diagnosis of a 12-cylinder trainset diesel engine based on vibration signature analysis and neural network," *Proc Inst Mech Eng C J Mech Eng Sci*, vol. 233, no. 6, pp. 1910–1923, Mar. 2019, doi: 10.1177/0954406218778313.

[65] T. Zheng, Y. Zhang, Y. Li, and L. Shi, "Real-time combustion torque estimation and dynamic misfire fault diagnosis in gasoline engine," *Mech Syst Signal Process*, vol. 126, pp. 521–535, Jul. 2019, doi: 10.1016/j.ymssp.2019.02.048.

[66] Y. S. Wang, N. N. Liu, H. Guo, and X. L. Wang, "An engine-fault-diagnosis system based on sound intensity analysis and wavelet packet pre-processing neural network," *Eng Appl Artif Intell*, vol. 94, p. 103765, Sep. 2020, doi: 10.1016/j.engappai.2020.103765.

[67] V. Kocaman, O. M. Shir, and T. Back, "Improving Model Accuracy for Imbalanced Image Classification Tasks by Adding a Final Batch Normalization Layer: An Empirical Study," in *2020 25th International*

Conference on Pattern Recognition (ICPR), IEEE, Jan. 2021, pp. 10404–10411. doi: 10.1109/ICPR48806.2021.9412907.

[68] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit*, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013.

[69] C. Qin *et al.*, "DTCNNMI: A deep twin convolutional neural networks with multi-domain inputs for strongly noisy diesel engine misfire detection," *Measurement*, vol. 180, p. 109548, Aug. 2021, doi: 10.1016/j.measurement.2021.109548.

[70] Y. Huangfu, E. Seddik, S. Habibi, A. Wassyng, and J. Tjong, "Fault Detection and Diagnosis of Engine Spark Plugs Using Deep Learning Techniques," *SAE Int J Engines*, vol. 15, no. 4, pp. 03-15-04–0027, Nov. 2021, doi: 10.4271/03-15-04-0027.

[71] S. M. Shahid, S. Ko, and S. Kwon, "Real-time abnormality detection and classification in diesel engine operations with convolutional neural network," *Expert Syst Appl*, vol. 192, p. 116233, Apr. 2022, doi: 10.1016/j.eswa.2021.116233.

[72] D. Binu and B. R. Rajakumar, "Introduction," in *Artificial Intelligence in Data Mining*, Elsevier, 2021, pp. 1–19. doi: 10.1016/B978-0-12-820601-0.00005-7.

[73] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Comput*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/neco_a_01199.

[74] W. De Groote, S. Van Hoecke, and G. Crevecoeur, "Prediction of follower jumps in cam-follower mechanisms: The benefit of using physics-inspired features in recurrent neural networks," *Mech Syst Signal Process*, vol. 166, p. 108453, Mar. 2022, doi: 10.1016/j.ymssp.2021.108453.

[75] P. Calvo-Bascones and M. A. Sanz-Bobi, "Advanced Prognosis methodology based on behavioral indicators and Chained Sequential Memory Neural Networks with a diesel engine application," *Comput Ind*, vol. 144, p. 103771, Jan. 2023, doi: 10.1016/j.compind.2022.103771.

[76] P. Han, A. L. Ellefsen, G. Li, V. Asoy, and H. Zhang, "Fault Prognostics Using LSTM Networks: Application to Marine Diesel Engine," *IEEE Sens J*,

vol. 21, no. 22, pp. 25986–25994, Nov. 2021, doi: 10.1109/JSEN.2021.3119151.

[77] H. Zhang, N. Lei, S. Liu, Q. Fan, and Z. Wang, "Data-driven predictive energy consumption minimization strategy for connected plug-in hybrid electric vehicles," *Energy*, vol. 283, p. 128514, Nov. 2023, doi: 10.1016/j.energy.2023.128514.

[78] H. Geraei, E. Seddik, G. Neame, E. (Yixin) Huangfu, and S. Habibi, "Machine Learning-Based Fault Detection and Diagnosis of Internal Combustion Engines Using an Optical Crank Angle Encoder," in *ASME 2022 ICE Forward Conference*, American Society of Mechanical Engineers, Oct. 2022. doi: 10.1115/ICEF2022-88851.

[79] Mehdi Sadeghkazemi, "Evaluation of Spark Plug Technologies in Spark Ignition Engines by Pareto Front Optimization," McMaster University, 2019.

[80] B. Ashok, S. Denis Ashok, and C. Ramesh Kumar, "A review on control system architecture of a SI engine management system," *Annu Rev Control*, vol. 41, pp. 94–118, 2016, doi: 10.1016/j.arcontrol.2016.04.005.

[81] E. Pipitone and A. Beccari, "Determination of TDC in internal combustion engines by a newly developed thermodynamic approach," *Appl Therm Eng*, vol. 30, no. 14–15, pp. 1914–1926, Oct. 2010, doi: 10.1016/j.applthermaleng.2010.04.012.

[82] T. Badawy, X. Bao, and H. Xu, "Impact of spark plug gap on flame kernel propagation and engine performance," *Appl Energy*, vol. 191, pp. 311–327, Apr. 2017, doi: 10.1016/j.apenergy.2017.01.059.

[83] Ö. Gültekin, E. Cinar, K. Özkan, and A. Yazıcı, "Multisensory data fusion-based deep learning approach for fault diagnosis of an industrial autonomous transfer vehicle," *Expert Syst Appl*, vol. 200, p. 117055, Aug. 2022, doi: 10.1016/j.eswa.2022.117055.

[84] R. Reshma and A. Jose Anand, "Predictive and Comparative Analysis of LENET, ALEXNET and VGG-16 Network Architecture in Smart Behavior Monitoring," in *2023 Seventh International Conference on Image Information Processing (ICIIP)*, IEEE, Nov. 2023, pp. 450–453. doi: 10.1109/ICIIP61524.2023.10537732.

[85] P. Fränti and R. Mariescu-Istodor, “Soft precision and recall,” *Pattern Recognit Lett*, vol. 167, pp. 115–121, Mar. 2023, doi: 10.1016/j.patrec.2023.02.005.

[86] M. Bai, X. Yang, J. Liu, J. Liu, and D. Yu, “Convolutional neural network-based deep transfer learning for fault detection of gas turbine combustion chambers,” *Appl Energy*, vol. 302, p. 117509, Nov. 2021, doi: 10.1016/j.apenergy.2021.117509.