Universidade Estadual de Campinas
Instituto de Computação

Amaury Bosso André

Edge Computing for Smart Machine Health Monitoring Systems

Computação de Borda para Sistemas Inteligentes de Monitoramento de Saúde de Máquinas

CAMPINAS
2019

Amaury Bosso André

Edge Computing for Smart Machine Health Monitoring Systems

Computação de Borda para Sistemas Inteligentes de Monitoramento de Saúde de Máquinas

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Jacques Wainer

Este exemplar corresponde à versão da Tese entregue à banca antes da defesa.

CAMPINAS
2019

Na versão final, esta página será substituída pela ficha catalográfica.

De acordo com o padrão da CCPG: "Quando se tratar de Teses e Dissertações financiadas por agências de fomento, os beneficiados deverão fazer referência ao apoio recebido e inserir esta informação na ficha catalográfica, além do nome da agência, o número do processo pelo qual recebeu o auxílio."
e
"caso a tese de doutorado seja feita em Cotutela, será necessário informar na ficha catalográfica o fato, a Universidade convenente, o país e o nome do orientador."

Na versão final, esta página será substituída por outra informando a composição da banca e que a ata de defesa está arquivada pela Unicamp.

Acknowledgments

I would like to express my deepest gratitude to Professor Jacques Wainer for being my supervisor, not only in this project, but since the graduation, my mastering degree and now for this project. I am very thankful of all the guidance and time spent during the whole process. The patience and inspiration through these years have really made the difference.

I would like to also thank SEMEQ and iX Consultoria for the opportunity of joining the large internal combustion engines monitoring project, specially to the power generation plant Gera Maranhão. The work reported in this thesis pressure reconstruction and combustion parameter estimation was supported by the Electric Power Sector of Technological Development and Research Program – R&D, regulated by Brazilian ANEEL (National Agency for Electric Energy Sector) with code number PD-6492-0215/2015.

The application for the machine health monitoring system was supported by SEMEQ, by making its database available for data extraction and by providing the hardware used for the development of the edge computing based architecture proposed in this work.

Resumo

Os sistemas de monitoramento de máquinas visam aprimorar a eficiência dos processos industriais, aumentando a produtividade e reduzindo o tempo inesperado de parada de máquina das linhas de produção. Porém, esses sistemas geralmente são projetados como redes que coletam uma enorme quantidade de dados das máquinas conectadas e disponibilizam esse grande volume de dados para interpretação e análise em alto nível, nos servidores em nuvem. Nesta tese, uma Rede Neural Convolucional com Wavelet (WCNN) é proposta como o modelo fundamental de um algoritmo de aprendizado profundo projetado para a extração de características do sinal de vibração, a ser implementado nos hardwares limitados dos sensores de vibração. Isso permite implementar o conceito de computação em borda, que direciona o processamento de dados computacionais para a borda de uma rede e é caracterizado em termos de alta largura de banda, baixa latência e acesso mais rápido às informações da rede. Com base no WCNN proposto, duas aplicações distintas são desenvolvidas para demonstrar a eficiência do sistema baseado em computação de borda. A primeira aplicação é para o monitoramento de grandes motores de combustão interna, com o Autoencoder Convolucional com Wavelet (WCAE) para extração de características do sinal e uma Rede Neural Convolucional com Wavelet (WCNN) para reconstrução de pressão no cilindro e estimativa de parâmetros de combustão. A segunda aplicação é para monitoramento de máquinas rotativas industriais, com Classificador Desbalanceado Convolutional com Wavelet (WCIC) para extração de características do sinal e detecção de falhas diretamente nos dispositivos de borda. Ambas as aplicações mostram resultados promissores para a implementação de sistemas práticos de monitoramento de máquinas no contexto da Internet Industrial das Coisas (IIot). Os benefícios da computação de borda são observados e as questões em aberto da literatura são consideradas e contempladas na arquitetura e nos modelos propostos.

Abstract

Machine health monitoring systems aim to enhance industrial processes efficiency, increasing productivity and reducing unexpected downtime of the production lines. But such systems are often designed as networks that gather a huge amount of data from connected machines and turn the big machinery data available for high level interpretation and analysis in the cloud servers. In this thesis, a Wavelet Convolutional Neural Network (WCNN) is proposed as the fundamental model of a deep learning algorithm designed for feature extraction of the vibration signal, to be embedded on limited vibration sensors hardware. It enables to implement the concept of edge computing, that directs computational data processing to the edge of a network and is characterized in terms of high bandwidth, low latency, and faster access to the network information. Based on the WCNN proposed, two distinct applications are developed to demonstrate the efficiency of the edge computing based system. The first application is for large internal combustion engines monitoring, with deep Wavelet Convolutional Autoencoder (WCAE) for feature extraction and a Wavelet Convolutional Neural Network (WCNN) for in-cylinder pressure reconstruction and parameter estimation. The second application is for industrial rotating machinery monitoring, with deep Wavelet Convolutional Imbalanced Classifier (WCIC) for feature extraction and fault detection directly on the edge devices. Both applications show promising results for implementing practical machine health monitoring systems in the Industrial Internet of Things (IIot) context. The edge computing benefits are observed and the literature open issues are considered and contemplated in the architecture and models proposed.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The industrial manufacturing is facing big innovations in terms of automation and data exchange. Called as the fourth industrial revolution, or Industry 4.0, this trend is driving the companies into the concept of smart factory, that is the combination of cyber-physical systems with Internet of Things and cloud or edge computing. It is also normally called as the Industrial Internet of Things (IIoT).

The spreading of low cost wireless sensors, with high connectivity, combined to advanced data analysis can enhance assets management because of higher and faster availability of information for decision making. And because of that, the IIoT enables an enhancement on industrial processes efficiency, increasing productivity and reducing unexpected downtime of the production lines.

As a key component in modern manufacturing systems, Machine Health Monitoring (MHM) systems offer robust solutions for fault detection. With vibration measurements, not only the detection is possible, but also failure diagnosis and the prediction of the components remaining useful life and working conditions. This makes MHM systems one of the key aspects that are driving industries to IIoT.

With that in mind, smart monitoring systems that combine machine health monitoring with deep learning algorithms are a major concern and has drawn substantial consideration of researchers. These recent reviews [21, 57] show how attractive are the deep learning and intelligent machine health monitoring, being a research trend in the area of machine learning.

But such systems are often designed as networks that gather a huge amount of data from connected machines and turn the big machinery data available for high level interpretation and analysis in the cloud servers. Edge computing, however, directs computational data, applications, and services to the edge of a network [34]. It is characterized in terms of high bandwidth, low latency, and faster access to the network information. These are very desirable characteristics for MHM systems and IIoT.

Despite the recent efforts in the literature, there are important issues that remain open in machine health monitoring through wireless sensor networks: (1) the sensor capacity for data acquisition - sophisticated techniques for failure diagnosis require high frequency and sufficient sampling time, specially for rolling element bearing components; (2) higher frequency vibration signals, to be transmitted as raw data, can overload the transmission capacity of sensor wireless networks, depending on the number of sensors and machines in

the monitoring system; (3) failure detection is performed in top level applications, usually in cloud servers, increasing latency and the response time for the monitoring system; (4) there is no flexibility in the network, and no priority can be established for data transmission or data analysis based on the signals content and the machine condition urgency.

This PhD thesis presents an Edge Computing based Smart Machine Health Monitoring system, where the signal processing and data analysis are performed on the edge devices. The architecture proposed aims to mitigate all the drawbacks previously mentioned with the development of Wavelet Convolutional Neural Networks (WCNN) to be embedded on vibration sensors. The deep learning algorithms proposed are designed for feature extraction of the vibration signals. The extracted features decreases significantly the sensor network load, aiming for fast processing and quick application response time, without loosing information for the real condition assessment of the monitored machines.

Two different applications of machine health monitoring are designed to test the efficiency of the edge computing based system proposed. The first application is an edge based system for large internal combustion engines monitoring, with deep wavelet convolutional autoencoder for feature extraction and a convolutional neural network for in-cylinder pressure reconstruction and parameter estimation. The second application is an edge based system for industrial rotating machinery monitoring, with deep wavelet convolutional neural network for feature extraction and fault detection directly on edge devices.

An overview of the main contributions of this thesis is presented in the section that follows.

1.1 Contributions Overview

The main contributions can be summarized as follows:

- *Edge Computing Based.* The proposal of a compact architecture for the Wavelet Convolutional Neural Network (WCNN) that is the base of implementation for the WCNN Autoencoder proposed for internal combustion engines monitoring, and the WCNN Imbalanced Classifier proposed for the industrial rotating machine monitoring. The compact architecture makes it possible to implement the models in the limited hardware of edge devices. The edge computing approach proposed leads to the following advantages:
  - *Fast processing.* Since features are extracted in the edge devices, less data is transmitted into the sensor network. With less, but more informative data, processing information is faster.
  - *Low latency.* With less data to transmit into the sensor network, and faster processing, the total latency decreases for information transmission to top servers.
  - *Faster response time.* Total time from the sensor acquisition until analysis and maintenance action can be decreased.

  - *Prioritization.* Information is processed closer to the edge devices, which permits the data to be flagged and receive special attention in top layers of the application.

- *Real measurements datasets.* Two different datasets were gathered in this work. One for large power generation engines, with vibration and in-cylinder pressure synchronized. It is a relevant dataset on a new application context. The other one, of vibration measurements from real industry components, is even more relevant because of the following characteristics:

  - *Big size.* Almost 300000 measurements.
  - *Imbalanced.* Less than 1% of fault samples.
  - *Diverse.* Many different kinds of components: motor, compressor, pump, bearing, gearbox, etc. From industries from different segments: chemical, foundry, food, power generation, stamping, etc. With all type of failures: unbalance, misalignment, wear, rolling element bearing failures such as inner race, outer race, etc.
  - *Labeled.* All samples were labeled by expert analysts.
  - *Incipient faults.* Fault samples contain incipient failures for early fault detection.

- *Pressure reconstruction for Internal Combustion engines.* The in-cylinder pressure reconstruction using the vibration signal as input is a challenging practical problem, that the architecture and model proposed can solve with very expressive results. The Wavelet Convolutional Autoencoder (WCAE) proposed in this thesis is generally applicable for pressure reconstruction and combustion parameters estimation on a new application context of large internal combustion engines of power generation plants.

- *Fault detection for rotating machines.* Machine health monitoring of rotating machines is one of the key aspects that are driving industries to IIoT. It is a practical problem that the architecture and model proposed in this thesis solve with very low false negatives rate, and the maximum accuracy possible, being generally applicable to any kind of machine on very different industry segments.

Chapter 2

Background

The following sections present the main works and concepts that are pertinent for the understanding of the architecture and the algorithms proposed in this thesis.

2.1 Predictive Maintenance

Machine health monitoring systems are part of a predictive maintenance strategy. Industrial maintenance comprehends a set of actions to ensure the proper working conditions for production line assets. Since industries are inserted in very competitive environments, to operate with maximum efficiency, in terms of quality, safety and reduced costs, is decisive in the company success.

There are two main kinds of maintenance: corrective and preventive [55]. In the first one, the maintenance interventions are performed after the equipment has a breakdown, while the second one aims to anticipate the failure occurrence. There are basically two different approaches for failure anticipation: by predetermined schedule of maintenance (calculated in terms of hours of work, cycles of usage, or even in elapsed time since the last intervention). Another way is through condition based monitoring, called as predictive maintenance.

In predictive maintenance, an intervention is triggered by a performance indicator or parameter monitoring, that shows an imminent degradation on the condition of operation of the machine. With predictive maintenance, it is possible to avoid unnecessary preventive maintenance schedules, executing them only when they are really necessary. Because of this characteristic, as stated in [11], intelligent maintenance systems are able to save up to 12% over scheduled repairs, reduce overall maintenance costs up to 30%, and eliminate breakdowns up to 70%, in average.

Besides that, there are some other key benefits from predictive maintenance, such as increased reliability of production processes, improvement of personal safety and of company image and increment of the overall performance of the company. But those benefits are achieved when it monitors a good parameter to identify deviation from normal condition operation. In this way, vibration analysis is the most effective technique used to detect mechanical failures in rotating machines [42].

2.2 Vibration Analysis

The principle of vibration analysis is the fact that the machines are excited by dynamic forces, and they result in vibratory signals. The frequency of vibration is dependent of the driving agents rotation speed. In the occurrence of misalignments between components, added masses in the rotor, gear problems or other kinds of components wear, specific frequencies are generated, usually as a function of the machine rotation. In reciprocating machines, such as internal combustion engines, the structure vibration caused by combustion in the cylinder contains abundant information about the combustion process and possible components failures.

Vibration analysis attracts a lot of scientific interest. Some applications are: structural health monitoring [46], where vibration data is used for structural condition monitoring and maintenance; detection of electrical failures such as broken bars in squirrel cage induction motors [51]; diagnosis of rotating machinery as wind turbine planetary gearbox [48]; combustion parameter estimation in internal combustion engines [9]; but a special attention is dedicated to rolling bearing faults, since in average, $45 - 55\%$ of breakdowns on industrial machinery happens because of the rolling element bearing failures [21].

The challenge in bearing diagnosis is that faults in inner or outer race, cage or on the rolling element have low energy and they appear in regions of high frequency in the spectrum. Therefore, they are easily masked by low signal-to-noise ratio, by the energy of other faults, or even by frequencies inherent in the operation of the machine itself, such as gear mesh frequencies, for example.

There are two main approaches for bearing fault detection [1]. The first one is a kurtogram based demodulation, proposed in [41]. This method, proposed by Randall and Antoni, corresponds to the following processing steps from the raw vibration data: a band-pass filter; the Hilbert transform that is a frequency-shift and decimation; the envelope signal is calculated and its Fourier transform obtain the squared envelope spectrum, that identifies bearing fault frequencies. The key point of this method is how to select the best frequency band for demodulation, in the first step. For that the fast kurtogram algorithm was proposed. Through the kurtogram it is possible to identify the frequency range that would best expose the characteristics of bearing failures.

However, there is an alternative path for achieving the squared envelope spectrum, that comprehends different steps, such as: calculate an autocorrelation function; apply a two dimensional Fourier transform; calculate the spectral correlation; integrate the spectral correlation over a spectral frequency band. In [40], it was discovered that integrating the spectral correlation over an informative spectral frequency band can generate the squared envelope spectrum, in which bearing fault frequencies can be detected.

The key point of this spectral correlation method is how to select the best frequency band for integration, in the last step. It used to depend on expertise and careful observations from the bi-spectral map. The work presented in [47] proposes to use the spectral kurtosis, for the characterization of the impulsiveness of bearing fault signals, and the Protugram and the Sparsogram for the characterization of the cyclostationarity of bearing fault signals.

All these works show that bearing diagnosis is one of the most challenging tasks in

machine health monitoring, because it requires high frequency vibration data, sufficient sampling time, and advanced signal processing techniques for good feature extraction and signal analysis in order to correctly identify rolling element bearing faults. But they also show that accurate signal processing for bearing diagnosis depend on the selection of the best frequency band for demodulation in the first approach and on the selection of the best frequency band for integration on the second approach.

2.3 Smart Monitoring System

The combination of machine health monitoring systems and machine learning or deep learning for feature extraction and failure detection creates the concept of Smart Monitoring Systems. To better understand monitoring systems, figure 2.1 shows the Open System Architecture for Condition Based Maintenance (OSA-CBM). It is a specification for a standard architecture in condition based maintenance systems [13].

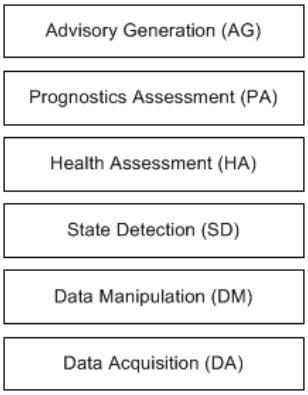

Figure 2.1: OSA-CBM blocks.

Lower levels consist of data acquisition and data processing. The first one, of Data Acquisition (DA), provides digital data from sensors or transducers, while the second one, of Data Manipulation (DM), process the signals to extract features that suggest parameters of degradation. The third module, of State Detection (SD), monitors the extracted features to detect anomalies. It has to be able to generate alerts, based on performance or security criteria.

Higher levels consists of interpretation and action supporting modules. Health Assessment (HA) module, makes the diagnosis over the detected state, for failure identification and localization. Prognostics Assessment (PA) makes an estimation on the Remaining Useful Life (RUL) of the component. The last module, of Advisory Generation (AG), recommend control or maintenance actions.

Usually in smart monitoring systems, except for data acquisition, all modules are implemented in top level applications. Due to the extensive computational resources necessary for data manipulation and machine learning algorithms training and execution, all the processing is executed in cloud servers.

Machine health monitoring systems have fully embraced the big data revolution [57], and data driven monitoring systems, specially the ones that implements machine learning

for automatic failure detection, are gaining more attention in industries nowadays.

2.4 Machine Learning for Condition Based Monitoring

Machine Learning have powerful pattern recognition tools, and these techniques have attracted great attention from many research fields in artificial intelligence, such as data mining and computer vision. As common rotating machinery fault diagnosis system consists of two key steps: feature extraction, and fault detection. And since fault detection is a failure pattern recognition, many artificial intelligence and machine learning algorithms were successfully implemented for rotating machinery fault diagnosis.

However, the accuracy of the proposed methods were always intrinsic related to the quality and representativeness of the extracted features. More conventional data-driven machine health monitoring systems usually consist of hand-crafted feature design, or feature extraction and selection.

In [3], a set of fourteen features from time and frequency domain are extracted from a dataset of industrial components. The extracted features were based on the expertise of human vibration analysts, and a combination of Support Vector Machine (SVM) an k-nearest neighbor (kNN) was proposed for classification with low false negative rates. The combination of SVM and kNN was later compared to other state-of-the-art methods for imbalanced classification using 33 standard datasets in [39].

According to [36], SVM has excellent performance in generalization, also with few training data, achieving high accuracy in classification tasks for rotating machinery fault diagnosis and condition monitoring.

Feature extraction from multi-domain aspect, such as time-domain, frequency-domain and time-frequency domain, was also presented in [54]. A Laplace score algorithm is introduced to select automatically the meaningful sensitive features, reducing redundant information from feature space. Than, a particle swarm optimization-based support vector machine (PSO-SVM) classification model is employed to implement the identification of multiple fault condition of rolling bearing.

Twenty-nine commonly used features for bearing fault diagnosis and performance assessment are extracted in [23]. Eleven time domain features, eight wavelet packet decomposition node energies, eight energy entropies when wavelet level is three, amplitude spectrum entropy and envelope spectrum entropy. Than a Nuisance attribute projection (NAP) is applied to the feature space to reduce the influence of operation conditions.

Besides that, according to the literature [36], many signal-processing methods have been applied for feature extraction to fault detection. Empirical mode decomposition (EMD) is a most commonly used time-domain method for feature extraction [35]. Time-frequency analysis such as short time Fourier transform (STFT), Hilbert-Huang transform (HHT), wavelet transform and wavelet packet transform (WPT) [53] are also often used for feature extraction.

But in all the previous works, designing the feature extractor has the drawback of requiring a great amount of human expertise and heavy signal processing techniques. Deep learning, however, provides an effective way to learn features automatically, without

depending on feature extractors, as detailed in the next section.

2.5 Deep Learning Feature Extraction

Deep Learning architectures can automatically extract multiple complex features from the input data without human engineers [21]. Convolutional Neural Network (CNN), is one of the deep learning algorithms that are getting increasing attention in recent works as a robust and efficient feature learning model for condition monitoring.

Convolutional neural networks have been proposed as the pattern recognition tool for fault detection, as in [8]. In this work, the CNN algorithm in the smart monitoring system detects faults in a gearbox, learning from a set of 256 predetermined features extracted from vibration signals. Not only as the pattern recognition tool, but as the feature extractor over vibration signals but converted in frequency domain, in [27]. And as a deep feature extraction from the raw vibration data, used for motor fault detection in [24].

Unidimensional convolutional neural networks, as the one proposed in this thesis are driving much attention for machinery fault detection. In [12] the proposed system takes directly raw time-series sensor data as input and it can efficiently learn optimal features. The developed architecture, has a compact architecture configuration, is cost effective and practical for small hardware implementation. Mainly because it has the ability to work without any predetermined transformation (such as FFT or DWT), hand-crafted feature extraction and feature selection.

Another unidimensional convolutional neural network is the one presented in [50]. Another deep learning algorithm that can automatically learn the feature classification diagnosis from the original vibration signals. It is compared with other feature extraction strategies, such as Empirical mode decomposition based neural networks (EMD-NN), and wavelet transform and wavelet packet transform neural networks (LWT-NN and WPT-NN). The results obtained by the 1dCNN are very promising results.

Besides convolutional neural networks, autoencoders are also used to outperform classical feature-engineering based methods. In [38], [45] and [29] different approaches with autoencoders of different characteristics are used to extract better signal features. These features are then provided to a separate classifier to diagnose the machine health.

In [45], the authors use stacked layers of autoencoders in an architecture for extraction of bearing characteristics, which are passed to a deep neural network to determine the severity of the failure. In [29], an autoencoder is trained in order to guarantee the extraction of characteristics with shift invariance. Shift invariance permits the identification of characteristics in different portions of the signal. In [38], layers are combined in the autoencoder of different sensors, to allow an architecture of extraction of multi modal characteristics.

The recent profusion of works proposing convolutional neural networks and autoencoders shows that these techniques have great potential for machine health monitoring and fault detection using vibration signals. However, some major challenges are still open in the literature. Which motivates the development of the models proposed in this thesis.

These challenges are discussed in the following sections.

2.5.1 Imbalanced Datasets

The vast majority of available vibration data are from normal equipment. There are usually few examples of failure conditions. Building a history of signals that highlight different problems in the equipment demands very long time of observation, or accelerated tests, which usually are not allowed for real machines in industry companies.

Many works in the area use tests made in laboratory that simulate failures found in the field [57, 15, 18]. To mitigate the challenge of imbalanced datasets, a combination of techniques is proposed in the literature, such as re-sampling of dataset examples, cost-sensitive learning and a modification in the loss function used for the model training.

In the literature, there are some works with different approaches for classification using imbalanced data. The work presented in [5] investigates the impact of class imbalance on classification performance for three benchmark datasets. It concludes that class imbalance has much effect on classification performance, and that oversampling on the minority class has better results in almost all analyzed scenarios.

The Synthetic Minority Over-sampling Technique is used in [22]. It classifies cropped patches of X-ray images, which is a very imbalanced dataset. A deep CNN was used, and SMOTE was the re-sampling method that helped the model to achieve the highest accuracy for the classification problem proposed.

Besides the dataset re-sampling strategies, cost-sensitive algorithms were also used to overcome the imbalanced dataset challenge. [4] presents a Cost-Sensitive Cross-Entropy Error Function for MLP neural networks to handle the common imbalanced classification problem. Work in [33] proposes a Cost Sensitive for deep CNN to deal with the class-imbalance problem based on the data statistics of the training set. Both articles conclude that the proposed pondering performs better or at least similarly to well-known classifiers.

But imbalanced datasets are not the only challenge of deep learning neural networks for feature extraction and pattern recognition in machine health monitoring systems. The learning metric should also be customized, to avoid false negatives, as explained in the next section.

2.5.2 Classification Recall

In Machine Learning, classification errors are defined as False Positives and False Negatives. Applied to the diagnosis of failures in industrial equipment, a False Negative happens when an equipment that has a defect is classified as normal. This category of error is catastrophic in industrial environments, as it may lead to a breakdown of the machine.

A breakdown is always associated with higher maintenance costs, longer time with the production stopped, not counting with the risks to the safety of the employees, and depending on the industry, even in risks to the population.

There are a number of researches that present very high accuracy results for machine health monitoring, or fault detection, such as [30, 49, 43, 37], just to name a few ones. Even

though the accuracy results are excellent, they do not comment about the presence of false negatives. Which in fact, in practical applications, should drive much more attention. Few works, as in [50], accuracy is compared in association to a metric of recall, that represents the model efficiency with false negatives.

This thesis proposes a smart machine health monitoring system with low false negatives rate restriction, using a combination of accuracy and recall metrics in the deep learning neural network. Besides that a modification in the loss function used in the model training is proposed to enhance results in comparison to traditional loss functions. The proposal of these strategies are described in details in section 3.4.

2.6 Wireless Sensor Network

Besides the theoretical background for the development of the research presented in this thesis in the previous sections, a more practical, but not less important background should be commented: of wireless sensor networks.

An important restriction to smart machine health monitoring systems based on edge computing, as proposed in this thesis, are low cost wireless sensors. The main characteristic of a vibration sensor is the quality of the measured signal. Unlike other sensors, where a temperature level, for example, or an overall value of some monitored parameter is sufficient to establish a condition of the equipment. In vibration analysis, as stated in section 2.2, to correctly diagnose faults, measurements are taken with high acquisition rates, because there are defects that manifest themselves only at high frequencies.

A comparative study on the market of wireless vibration sensors is presented in figure 2.2.

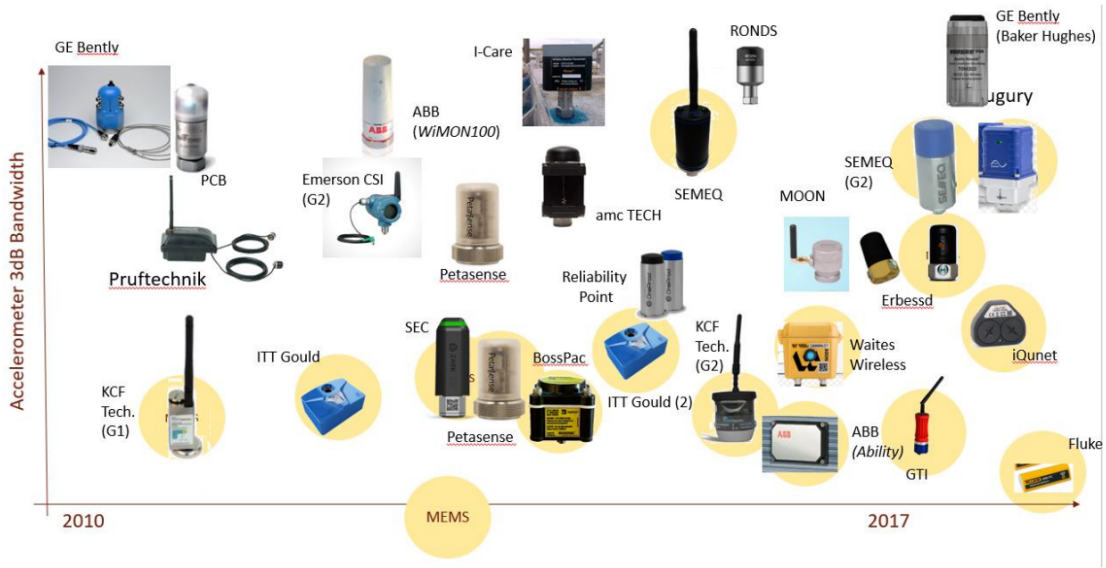

Figure 2.2: Comparative of vibration sensors

The x axis of the graph in figure 2.2 indicates the release year of each sensor, while the y axis shows their bandwidth. Sensors with higher frequency range, around $10kHz$

are at the top of the scale, while most sensors have a bandwidth of about $1kHz$ only. Because of the discussion made on section 2.2, these $1kHz$ sensors do not perform well for machine health monitoring systems.

The frequency range of the sensor is fundamental to its proper functioning and for complex applications as the one described in this thesis. However, the higher the frequency range observed by the sensor, the higher its acquisition rate should be. But with higher acquisition rate, the signal length that must be stored and transmitted by the sensor is bigger. This directly impacts in the battery autonomy, memory size, processing capacity, number of sensors per gateway, and consequently the total cost of the system.

The high frequency sensors, from figure 2.2, are almost all piezoelectric sensors. The ones based on MEMS (Micro Electro Mechanical Systems) accelerometers (highlighted in yellow on the graph), are in majority of low frequency. The use of MEMS sensors has great impact on consumption and especially on the price of sensors. The traditional piezoelectric are usually much more expensive and consume more power, which in practice, makes them not attractive to machine health monitoring systems.

Even though it may seem that there are many options in the vibration sensor market, only a very few of them have the desired requirements for machine health monitoring systems. In this thesis, we will use the vibration sensor developed in SEMEQ as the hardware reference for the development of the architecture and models proposed. The main technical characteristics of this sensor, relevant on the assumptions made in this thesis are presented in table 2.1.

Table 2.1: Sensor technical characteristics.

Processor	Arm	Cortex-M0
Memory	Flash	256KB
	Internal RAM	32KB
	External RAM	256Kb (32KB)
Conectivity	Bluetooth	BLE 4.2
Accelerometer	Range	10kHz
	Acquisition Rate	24KS
	Sensitivity	40mV/g
	Gain	0 a 48db

It has a high frequency acquisition, with $10kHz$ of frequency range, good sensitivity and big dynamic range. The models are proposed in this thesis taking into account the limited memory available and its low processing capacity.

Chapter 3

Proposed Architecture

In this chapter, the proposed architecture, and the algorithms for machine health monitoring system based on edge computing are discussed. The wavelet convolutional neural network and its modifications in activation function, metrics measurement and loss function are detailed in the sections that follows.

3.1 Edge Computing based Monitoring System

Monitoring systems based on wireless sensor networks are commonly divided into three distinct modules. One responsible to collect sensible data from the target equipments, a transmission layer, and a top module of analysis, commonly located at the cloud servers.

In vibration analysis, the collecting phase can be done by wireless sensors. As discussed in section 2.2, to fully accomplish the goal of failure detection and diagnosis, a high frequency and high sensitivity accelerometer sensor is necessary. Specially to monitor rolling element bearings, one of the main causes of breakdowns in the industry.

However, the need of higher sampling rate implies in more memory, more processing capacity, and mainly, more battery power for the signal transmission in the sensor. Transmission time drains battery in the sensor and overloads the transmission network, making it necessary to have more gateways per sensor to try to compensate the latency of transmission.

The proposed edge computing based architecture can be seen in figure 3.1.

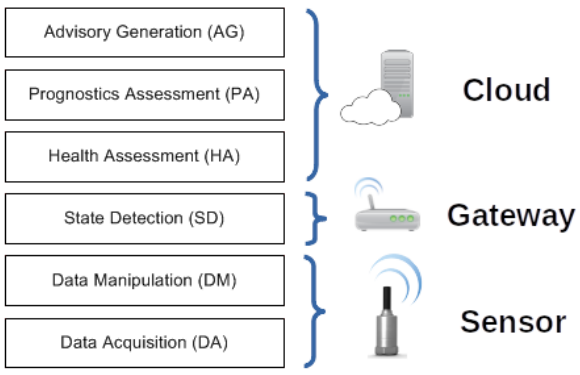

Figure 3.1: Edge Computing based Monitoring System

The new architecture proposed here is based on edge computing, where the processing resources and data analysis are performed on the edge devices, for low latency and minimum load on the network, as discussed in the related works [34, 52, 26]. It is based on the Open System Architecture for Condition Based Monitoring (OSA-CBM), described in figure 2.1.

The system proposed can collect and manipulate data inside the vibration sensor, with a Wavelet Convolutional Neural Network (WCNN). The WCNN, detailed in section 3.2.

When the WCNN is implemented in the form of the Wavelet Convolutional AutoEncoder (WCAE), as described in section 3.3, the encoder phase is embedded on the vibration sensor. Since the aim of the WCAE is to compress the original to an intermediate layer with lower dimension, the benefits of edge computing can be observed.

When the WCNN is implemented in the form of the Wavelet Convolution Imbalanced Classifier (WCIC), as described in section 3.4, the goal is to extract the best features inside the vibration sensor and transmit them for fault detection in the gateway. When no failure is detected, the full data transmission is discarded, saving resources in the network. Only in the case of a failure detection, the full signal is required by the gateway to the sensor, and transmitted for further analysis in the cloud application.

As the proportion of defective equipments are normally much lower in comparison to normal ones, a considerable reduction on data transmission is observed. Besides that, a risk grade is also calculated in the gateway device, and this information is used to prioritize transmission and analysis of the machine information in top level applications.

The wavelet convolution neural network presented in the next section is the proposed solution for data manipulation and feature extraction, that together with data acquisition, are the two bottom modules of the OSA-CBM adapted architecture, illustrated in figure 3.1.

3.2 Wavelet Convolutional Neural Network - WCNN

Convolutional neural networks, as discussed in section 2.5, have remarkable results in feature extraction, specially when dealing with images. In vibration analysis, the accelerometer measures an unidimensional signal that represents the vibration of the component over the time. Because of that, unidimensional convolutional neural networks are used for feature extraction in machine health monitoring systems that use vibration sensors, as in [50, 12].

Wavelet convolutional neural networks, as summarized in [2], are algorithms for analyzing a wide range of time-series, and it is a powerful tool for representing nonlinearities. The literature have different approaches to incorporate wavelets into neural networks. Some works [25, 7, 6], make a wavelet packet transform (WPT) with envelope analysis as a preprocessing phase for feature extraction. While other works, incorporate wavelet functions in the pooling and/or in the dropout layers [14, 16].

In this thesis, the wavelet function replaces *sigmoid* or *relu* in activation function for the convolution layers. Works such as [19, 32, 17] have proposed similar functionality for wavelet functions, because of their capacity of nonlinearities representation, even with the

convergence speed of wavelet neural network being slower [17]. In figure 3.2, a generic unidimensional wavelet convolutional neural network model is detailed.

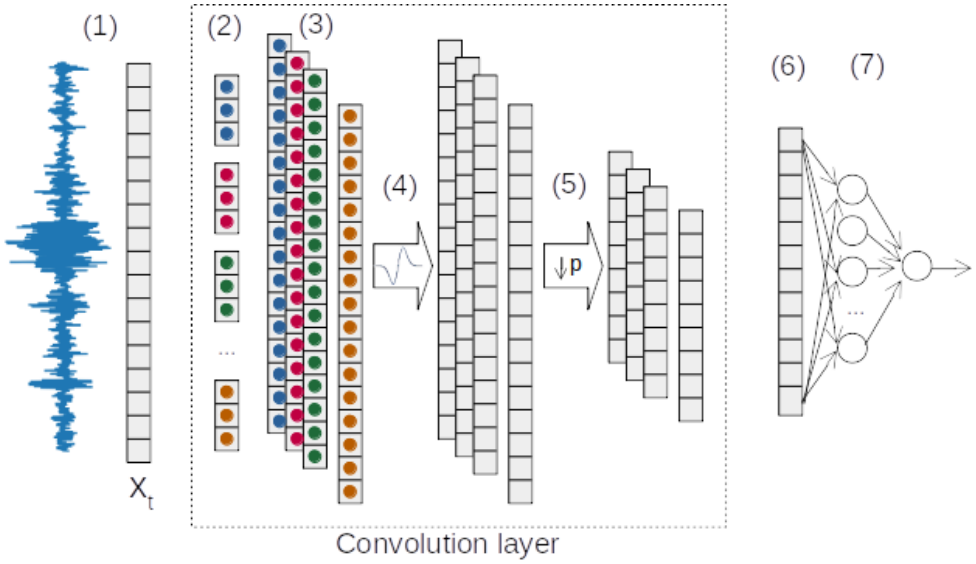

Figure 3.2: Generic WCNN model

Some aspects are relevant to observe and are enumerated in figure 3.2: the input is a unidimensional raw vibration signal (1); the convolution layer is presented in detail, with convolution operations based on the number and size of filters (2); the convolution result of the input data for each convolution filter is demonstrated in different colors, one by each convolution filter (3); the convolution is followed by a wavelet activation function (4) and a pooling layer that reduces the length of signals by a factor p (5); the resulting convolved arrays are flatten to a unidimensional vector (6); the convolved vector passes to a densely connected neural network (7).

Depending on the applications presented in this thesis, the number and size of filters can differ, as the number of convolution layers and the size of the densely connected neural network. But the structure follows the architecture proposed by the Wavelet Convolutional Neural Network in this section.

In the following sections, every phase on the generic convolution layer proposed are described in details.

3.2.1 Convolution

A convolution operation itself, is represented by the equation 3.1. It shows the convolution operation, denoted by $*$, of a filter f, with size of $2k+1$, where k is an integer greater than zero, to a unidimensional signal g, which has a length of n values.

$$(f * g)[n] = \sum_{m=-k}^{k} f[m].g[n+m] \tag{3.1}$$

An illustration of the convolution operation, implemented as a sliding window, is presented in figure 3.3.

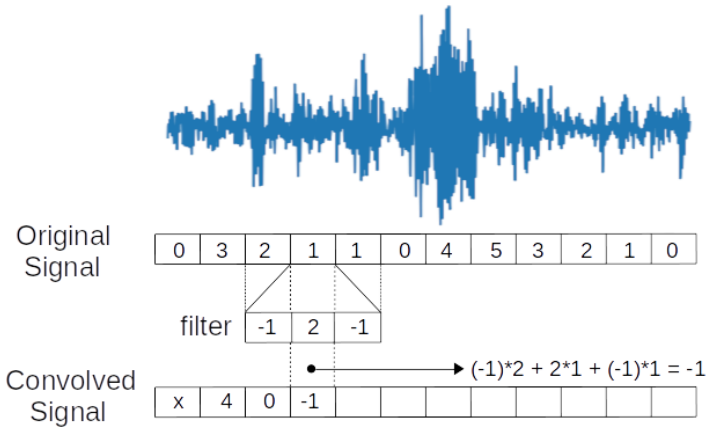

Figure 3.3: Convolution operation illustration

In mathematics, the convolution theorem, states that a convolution in time domain is equivalent to a point-wise multiplication in the frequency domain - see equation 3.2, where $\mathscr{F}$ denotes the Fourier transform. The consequence of these point-wise multiplications, when applied in frequency domain, are the enhancement or attenuation of certain frequencies in the original signal, depending on the value of filter applied.

$$(f * g)[n] = \mathscr{F}^{-1}\{\mathscr{F}\{f\}.\mathscr{F}\{g\}\} \tag{3.2}$$

Training a convolutional neural network corresponds to adjustments to the convolution filters weights. According to the convolution theorem, it affects to which frequencies of the original signal are going to be attenuate or enhanced, if we take in consideration a neural network with only one convolutional layer.

Following the same intuition used before, a second convolutional layer, stacked for deep learning, will attenuate or enhance frequencies from the signal already filtered in the first convolutional layer, making it possible to highlight modulated frequencies, for example.

Because of these properties, convolution neural networks tend to be very well suited for vibration feature extraction as the results in this thesis will show.

3.2.2 Wavelet Activation Function

The result of the convolution operation is passed to an activation unit. In this work we propose as activation a Gaussian wavelet function. The formulation of this wavelet activation function is presented in equation 3.3.

$$w(x) = \frac{x}{\sqrt{2\pi}} e^{-x^2/2} \tag{3.3}$$

Where w denotes the wavelet function and x the input value. Figure 3.2 shows the wavelet activation function calculated based on the equation 3.3.

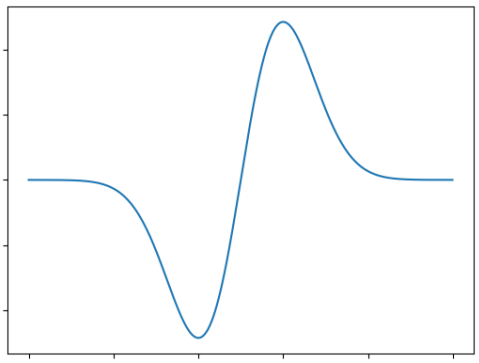

Figure 3.4: Wavelet activation function

For vibration signal feature extraction, the gaussian wavelet as activation function is able to give a suitable compromise between temporal and frequency selectivity in the time-frequency representation of the vibration signals [19]. The work presented in [19] also shows that the wavelet activation function, as the one proposed for the Wavelet Convolution Neural Network (WCNN) architecture in this thesis, can greatly enhance feature learning from the non-stationary vibration signals of electric locomotive bearings.

3.2.3 Pooling

Convolutional neural networks have a pooling layer after the convolution operation to reduce size of the features extracted in the network. The most common pooling strategy is a max-pooling layer, which preserves the local max value over the input features. For each p values, the pooling layer keeps the max values and discards the others. An example of a max-pooling layer is presented in figure 3.5.

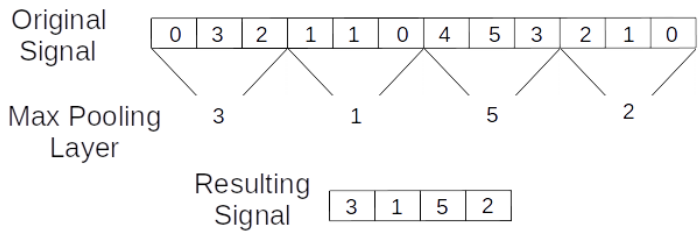

Figure 3.5: Max Pooling layer illustration

The Convolution Layer is then composed by the elements detailed before (convolution operation, wavelet activation function and the max-pooling layer). However, two different approaches are presented in this thesis, based on the generic Wavelet Convolutional Neural Network (WCNN) proposed in this section. The first one is an Autoencoder for unlabeled data, and the second one an Imbalanced Classifier for labeled data, of normal or defective industrial rotating equipments. Both approaches that are implemented with the generic WCNN are described in the following sections.

3.3 WCNN based Autoencoder - WCAE

As discussed in section 2.5, autoencoders are powerful tools for feature extraction. The encoder phase of the autoencoder maps the input data from a high-dimensional space into a low-dimension representation of the feature space. The decoder phase, in the other hand, reconstructs the original input data back from the corresponding low-dimensional feature representation. Figure 3.6 shows both phases.

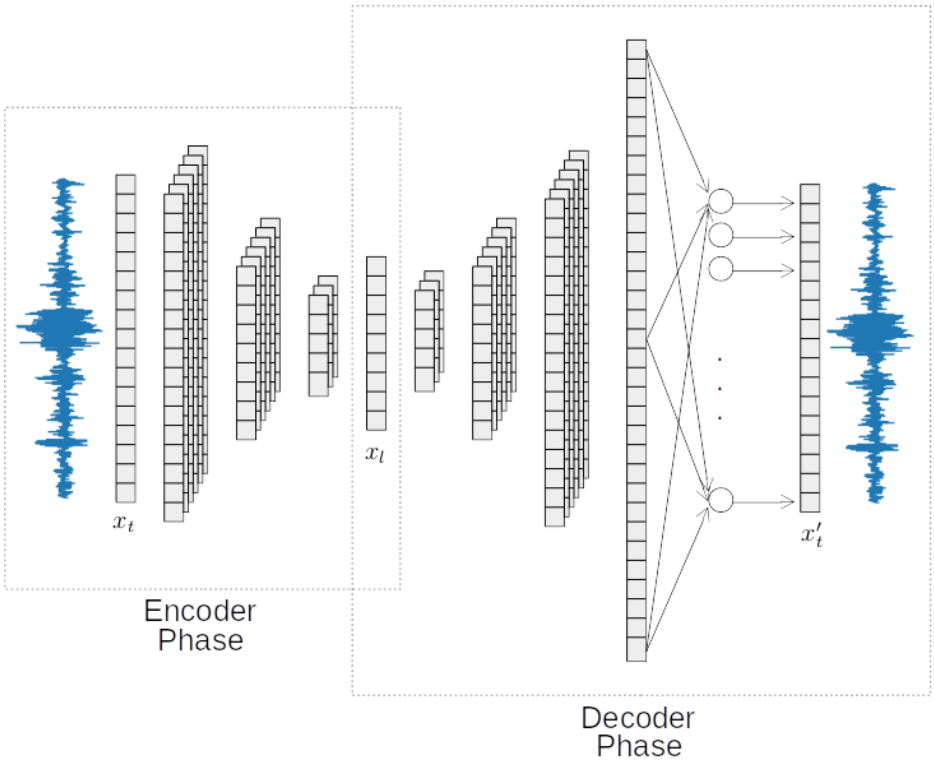

Figure 3.6: Wavelet Convolutional Autoencoder

From figure 3.6, the autoencoder takes as input a unidimensional raw vibration signal (denoted as x_t), makes a sequence of wavelet convolutional layers as described in the last section (with the convolution followed by a gaussian wavelet activation function and a max-pooling layer) and the result is flattened to a unidimensional array (denoted by x_l).

In the encoder phase, the unidimensional signal $x_t \in \mathbb{R}^D$ with dimension D is converted to a signal $x_l \in \mathbb{R}^d$ with dimension d where $d < D$. The decoder phase, implements the inverse operations of the encoder phase. The same convolution layers in reverse order, but instead of max-pooling by factor of p, an up-sampling of the same factor p is executed. The autoencoder is therefore, symmetric in its architecture.

After the convolution layers, the resulting signal is flattened and passed to a densely connected neural network, with one hidden layer, to reconstruct the original signal $x'_t \in \mathbb{R}^D$.

As stated by the operations described before, the goal of the autoencoder is to shrink the original signal to a compressed form and than reconstruct the original signal. To evaluate the efficiency of reconstruction, the Mean Absolute Error (MAE) metric is calculated and used as the loss function to train the autoencoder parameters. The MAE

metric represents the mean of the absolute difference from the reconstructed vibration signal x'_t in comparison to the real vibration measurement x_t, as equation 3.4 shows.

$$E_{mae} = \frac{\sum_{i=1}^{D} |x'_t[i] - x_t[i]|}{D} \tag{3.4}$$

The lower the MAE metrics result, better is the reconstruction performed by the WCAE. It means that the original signal x_t is compressed without significant loss of information to the lower dimension signal x_l. That is an important condition for the edge computing architecture proposed in this thesis.

The WCAE model described in this section is designed and implemented as part of the in-cylinder pressure reconstruction and combustion parameters estimation application, presented in section 4.1. More specific details, such as the number of convolution layers, the size of the convolution filters, the model dimensions, the training and testing strategies and the results obtained with the WCAE are also detailed in section 4.1.

3.4 WCNN based Imbalanced Classifier - WCIC

As discussed in section 2.5, an increasing number of papers in the recent literature is proposing unidimensional convolutional neural networks for machinery fault detection. The Wavelet Convolutional Imbalanced Classifier (WCIC) proposed here is based on the generic architecture of the WCNN described in section 3.2. The goal for the WCIC is to extract the most relevant features for fault detection, implemented through a binary classifier.

The structure of the WCIC proposed can be seen in figure 3.7.

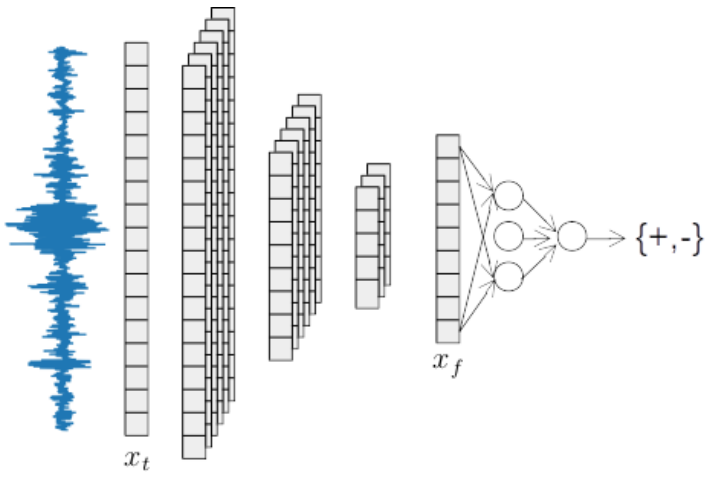

Figure 3.7: Wavelet Convolutional Imbalanced Classifier

From figure 3.7, the input vibration signal (denoted by x_t) is passed through a sequence of wavelet convolutional layers (with the convolution followed by a gaussian wavelet activation function and a max-pooling layer), as described in the section 3.2. Then the resulting array of extracted features (denoted by x_f) is the input of a densely connected

neural network. The hidden layers of the densely connected neural network have *ReLU* activation functions, described in equation 3.5 and illustrated in figure 3.8.

$$f(x) = x^{+} = max(0, x) \tag{3.5}$$

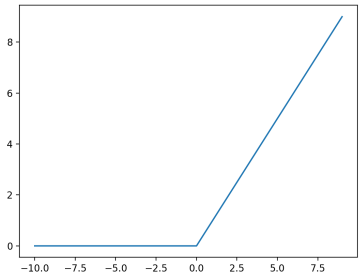

Figure 3.8: Rectified Linear Unit - ReLU

The output layer, however, has one neuron only and with a *Sigmoid* activation function, designed for the binary classification. The *Sigmoid* activation function is described in equation 3.6 and illustrated in figure 3.9.

$$f(x) = \frac{e^{x}}{e^{x} + 1} \tag{3.6}$$

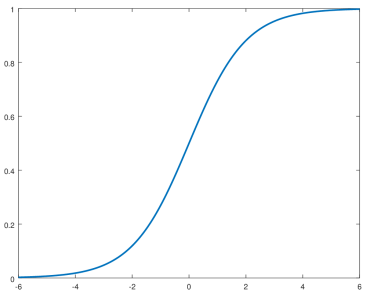

Figure 3.9: Sigmoid function

The goal for the WCIC as a binary classifier, is to state the machine condition of operation as one of the two following possibilities: normal or defective. From the *Sigmoid* function, the WCIC output values range from zero to one. The higher values of output means higher probability of defective vibration signals in the input.

The binary classification is executed by the comparison of the WCIC output value with a threshold. This threshold is one of the hyper parameter of the training model, since lower thresholds implies in more conservative boundary decisions, helping to avoid false negatives.

To train the model parameters and weights we use a loss function based on the output of the WICI. The most commonly used loss function is the Binary Cross Entropy, also called as the Log Loss function, defined by the equation 3.7.

$$L_{bc} = -(y \log(y_p) + (1 - y) \log(1 - y_p)) \tag{3.7}$$

From the equation 3.7, y is the real value of the machine classification (zero for normal and one for defective machines). The value denoted by y_p, is the predicted output value of the WCIC for the same machine. The binary cross entropy, or log loss, is calculated and increases as the predicted probability diverges from the real machine condition. Taking a closer look to the equation 3.7, we can separate it in two parts, as equation 3.8 shows.

$$\begin{aligned} a &= -y \log(y_p) \\ b &= -(1-y) \log(1-y_p) \\ L_{bc} &= a + b \end{aligned} \tag{3.8}$$

Since y is the real classification (zero for normal or one for defective machines), the first part of the equation is only considered in the final calculation of L_{bc} when the machine is defective ($y = 1$). And also, term a increases when false negatives occur. The second part of the equation has the opposite behavior. Term b is only considered in the final calculation of L_{bc} when the machine is normal ($y = 0$) and increases when false positives occur.

Usually cost sensitive convolutional neural networks ponder the contribution of both terms, based on the proportion of examples of each class in the training data set. Assuming that the number of normal examples in training dataset is bigger than the number of defective ones, and that c_1 and c_2 are the proportional cost of false negatives and false positives respectively, equation 3.9 is used.

$$\begin{aligned} a &= -y \log(y_p) \\ b &= -(1-y) \log(1-y_p) \\ L_{bc} &= c_1.a + c_2.b \end{aligned} \tag{3.9}$$

As we can see, the cost sensitive cross entropy is usually defined as a linear combination of the classes errors of classification. Instead of calculating the loss function as the binary cross entropy, the proposal of this thesis is to calculate the loss function as presented in equation 3.10 that follows.

$$\begin{aligned} a &= -3y(\sqrt[4]{y_p} - 1) \\ b &= -(1-y) \log(1-y_p) \\ L_{fr} &= c_1.a + c_2.b \end{aligned} \tag{3.10}$$

The loss function for low false negative also considers the two hyper parameters c_1 and c_2, that are the proportional cost of false negatives and false positives respectively. But term a of the equation 3.10 is calculated based on the fourth root of the predicted value of classification.

Figure 3.10 shows in the x axis the y_p value and in the y axis the resulting value of term a from the log and the fourth root losses functions.

For comparison purpose, the *Logcosh* loss function described in equation , is also

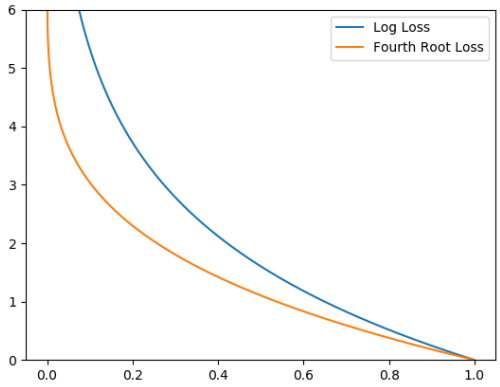

Figure 3.10: Comparison of Log and the Fourth root losses

implemented and tested. It is another commonly used loss function for imbalanced classification.

$$\begin{aligned} a &= -y\left((y_p - 1) \;+\; \log\left(1 + e^{2(y_p - 1)}\right) - \log\left(2\right)\right) \\ b &= -(1-y)\left((y_p) \;+\; \log\left(1 + e^{2(y_p)}\right) - \log\left(2\right)\right) \\ L_{lc} &= c_1.a + c_2.b \end{aligned} \tag{3.11}$$

Figure 3.11 presents a visualization of all analyzed losses, as functions of activation on positive samples.

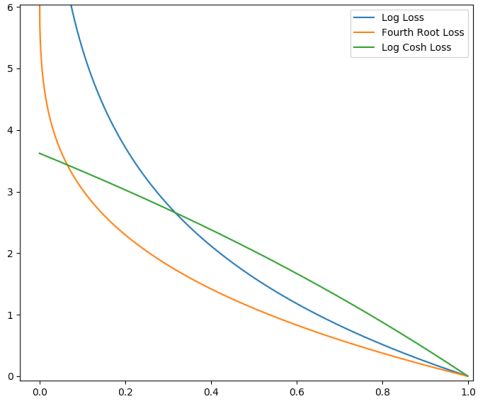

Figure 3.11: Visualization of analyzed losses on positive samples

The results for the comparison of the proposed loss function to the cost sensitive binary cross entropy, and the *Logcosh* loss functions are presented in the section 4.2.4.

Since the WCIC is a supervised learning algorithm, it must deal with the challenges discussed in section 2.5.1 of imbalanced datasets, and in section 2.5.2 of classifications recall. And for that, to evaluate the efficiency of classification, the following metrics are considered in this thesis.

- **Accuracy**: proportion of corrected predictions divided by the total;

$$acc = \frac{t_p + t_n}{t_p + f_p + t_n + f_n} \tag{3.12}$$

- **Recall**: proportion of correct fault detections divided by the sum of fault detections and false negatives;

$$rec = \frac{t_p}{t_p + f_n} \tag{3.13}$$

In the equations 3.12 and 3.13, t_p and t_n denotes true positives and true negatives, that are the number of correct predictions. While f_p and f_n are incorrect predictions of false positives and false negatives.

As discussed in section 2.5.2, false negatives are much worse than false positives for machine health condition monitoring. So, the model goal is to maximize the accuracy on the classification, with the restriction of a minimum value for the recall metrics, close to 1, in order to minimize the number of false negatives.

The higher are the metrics result, better is the classification performed by the WCIC. It means that the array of features x_f extracted from original signal x_t is calculated without significant loss of information. The fact that x_f has a very lower dimension than the original vibration signal x_t is an important condition for the edge computing architecture proposed in this thesis.

The WCIC model described in this section is designed and implemented as part of the rotating machinery health monitoring application, presented in section 4.2. More specific details, such as the number of convolution layers, the size of the convolution filters, the model dimensions, the losses functions comparison, the training and testing strategies, the acceptable restriction value for the recall metric, the threshold value for classification and the results obtained with the WCIC are also detailed in section 4.2.

3.5 Embedded Implementation of the WCNN

As discussed in section 3.1 where the Edge Computing based Monitoring System was presented, the Wavelet Convolution Neural Network (WCNN) is the solution for data manipulation and feature extraction, that together with data acquisition, are the two bottom modules of the OSA-CBM adapted architecture, illustrated in figure 3.1.

But the advantages of edge computing in this context is only possible if the embedded implementation of the WCNN can be done correctly and the execution in the limited hardware available in the edge computing devices can be efficiently performed.

Because of hardware restrictions, there are two major concerns in this kind of implementation: (1) the amount of memory used and (2) floating point operations.

The first limitation, is the memory available for calculations. The vibration sensor used for the implementation of the applications presented in this thesis have 32KB of RAM memory with an expansion memory of others 32KB in an external RAM, which give us the total amount of only 64KB for data acquisition and manipulation.

The memory usage is highly dependent on the model proposed in each application, which concerns of number and size of the convolution filters and the quantity of convolution layers. In each of the applications sections, 4.1 and 4.2, these parameters are detailed and the amount of memory necessary for data manipulation in the sensor is calculated.

The second limitation is about floating point operations. Usually devices such as the vibration sensors do not have a floating point unit. So operations that involve decimal numbers are considerably slower than integer operations. In practical applications, a conversion from floating point to integer values should be implemented, when there is no floating point unit available in the device hardware.

The conversion is performed through an empirically defined scaling factor. Every value, from the input signal through model weights and activation function values should be multiplied by the scaling factor and truncated to its integer part.

As most of these sensors devices work with a 12 bit ADCs (Analog to Digital Converters), the data representation of values are limited to 16 bits. Because of that, a scale factor with value of 10000, that corresponds to a precision of four decimal points, is a good choice.

Besides integer conversion, the activation functions should be interpolated and stored statically in the devices memory, because the calculation in every activation considerably slows the process computation. The wavelet values for the activation function proposed in the WCNN are calculated and stored in an static array. But the length of this array highly impacts on the performance of the neural network. The best trade-off on the array size and performance of calculation should be empirically defined.

Chapter 4

Applications

In this chapter, two different applications of machine health monitoring systems, based on edge computing, are detailed. Both applications implement the Wavelet Convolutional Neural Network (WCNN) proposed in the last chapter with very promising results.

The first application is a monitoring system for large internal combustion engines. An important parameter for condition based monitoring of these machines is the in-cylinder pressure signal, during the combustion process. But directly measure in-cylinder pressure has many drawbacks due to the harsh environment inside the combustion chamber.

Because of that, the use of indirect measurements have great potential, and a Wavelet Convolutional Autoencoder (WCAE), based on the one proposed in section 3.3 is developed. The input is the raw vibration data from the engine block, and the WCAE learns features to reconstruct the in-cylinder pressure signal for different operational conditions.

The second application is an industrial monitoring system, for very different kinds of rotating machines, such as induction motors, pumps, gearboxes, rolling element bearings, reducers, and compressors, in many industry segments such as: chemical, foundry, food, power generation, and stamping.

Automatic fault detection on rotating machines involves pattern recognition through a classification problem from features extracted out of vibration measurements. For that, a Wavelet Convolutional Imbalanced Classifier is proposed, based on the one presented in 3.4. The WCIC takes raw vibration data and historical information of the components to achieve good detection accuracy, with very low false negative rate, as discussed in 2.5.2.

4.1 Internal Combustion Engines

Condition-based monitoring of internal combustion engines, specially for large power generation plants, prevents unexpected breakdowns and consequently stoppage in power generation. As most defects found in internal combustion engines are directly related to the combustion process, the in-cylinder pressure provides useful information for the detection of problems in the combustion phasing, injection problems, leakages due to degradation in the piston ring or in the cylinder wall.

Combustion sensing is usually performed with an in-cylinder pressure sensor [31]. But directly measuring pressure has several drawbacks, enumerated as follows. (1) A high performance and highly resistant transducer is required due to the harsh environment that it is exposed inside the cylinder combustion chamber. (2) Pressure sensors are considerably more expensive in comparison to other type of sensors, such as accelerometers and tachometers. (3) The harsh environment significantly reduces pressure transducer lifetime when permanently mounted in the inspection point. (4) Combustion chamber deposits on the transducer through time is also an important drawback for constant direct measurements, because it heavily affects the values read by the sensor. All these factors reduce the efficiency, accuracy and mainly the economic feasibility of the constant monitoring process of large internal combustion engines using dedicated pressure sensors.

Indirect measurements have great potential for engine diagnosis, and some methodologies were presented in the literature with good accuracy. They use non-intrusive techniques and sensors, such as accelerometers and tachometers, instead of pressure transducers. These sensors are externally mounted on the engine block. The goal is to detect combustion events through vibration signals. In-cylinder pressure and vibration on the engine block are correlated. Events that occur in the cylinder pressure internal dynamics are transmitted to the engine external surfaces [31], such as valve opening and closing, fuel injection and burning, and mechanical abrasions.

In-cylinder pressure is the main tool for the detection of problems in the combustion process, so the edge computing based monitoring system proposed for internal combustion engines takes the vibration signal measured by the sensor to reconstruct the in-cylinder pressure, and estimate some parameters of combustion, such as maximum pressure peak value and its angle location in relation to the axis rotation.

Continuously measure the combustion parameters allows not only fault detection, but also permits a reduction in fuel consumption and in pollutant emissions. This because deviations from normal behavior in the combustion process can be observed earlier.

The proposal for internal combustion engines monitoring, using vibration signal, is a Wavelet Convolutional Autoencoder (WCAE), based on the generic Wavelet Convolutional Neural Network (WCNN), described in section 3.2. The aim of the autoencoder is to reduce the amount of data transmitted by the sensor, without loosing information for the pressure reconstruction and combustion parameter estimation done in the server side.

The proposed method is applied to data measured in combustion engines of a large power generation plant. The results confirm that pressure reconstruction and parameter estimation of the combustion process for those engines are performed with great success. The WCAE proposed is a new regressor generally applicable for pressure reconstruction

and combustion parameters estimation on a new application context of large internal combustion engines of power generation plants.

Before explaining the architecture of the Wavelet Convolutional Autoencoder, the algorithms developed for training, and the results and discussions on performance achieved, the experimental setup together with the data measurements and procedures are presented in the next section.

4.1.1 Experimental Setup

The experiments were carried out in a power generation plant in the state of Maranhão, Brazil. The GeraMaranhão is a thermoelectric power plant consisting of 38 Wärtsilä W20V32 engines (Figure 4.1). The Wärtsilä W20V32 is a vee engine with 20 cylinders divided in two banks: from A1 to A10, and from B1 to B10. Each engine has a generation capacity of $8,73MW$ at $60Hz$, while the whole power plant has a total capacity of $331,74MW$, and consumes more than 1600 ton/day of fuel oil.

Figure 4.1: Wärtsilä engines in the power generation plant.

More engine specifications are listed in table 4.1.

Table 4.1: Engine specifications.

Number of cylinders	20
Bore [mm]	320
Stroke [mm]	400
Con Rod [mm]	850
Rotation Speed [rpm]	720

The data acquisition system measures simultaneously vibration, pressure and the engine rotation speed, with 12 bits of resolution and with $24KS/s$ of sampling rate. These parameters were used to match the wireless sensor specification proposed to build the edge computing monitoring system for these motors.

The accelerometer used for vibration measurements is a piezoelectric one with $100mV/g$ of sensitivity, dynamic range of $\pm 50g$ peak and maximum shock protection of $5000g$ (peak). The accelerometer temperature range varies from -50 to 121^oC, while its frequency range response with $\pm 3db$ is 0.5 to $15000Hz$. To measure rotation speed, a laser photo tachometer with $0.25ms$ of answering time and work temperature range of -10 to 50^oC was used. In-cylinder pressure signal was acquired using an AVL sensor with $13mV/bar$ sensitivity, measuring range of 0 to $150bar$, and operating range of -10 to 110^oC.

Figure 4.2: Engine measurement

Five different engines were measured in the power generation plant. For each one, the flying wheel had to be marked with a reflexive tape for tachometer readings of rotation speed. To guarantee repeatability, the reflexive tape was positioned in the same place for every engine, on the TDC point of the first cylinder A1, assumed as the zero degree reference. The TDC (Top Dead Center) is the position that the piston is in his top position (when the ignition starts the combustion).

The angles, together with the cylinders sequence of combustion, were retrieved from the engine geometry, and can be seen in table 4.2.

Table 4.2: TDC angles for every cylinder

Cylinders angles in relation to reference cylinder A1									
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
0	232.5	155	77.5	670	310	465	540	592.5	387.5
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
60	292.5	215	137.5	729.5	370	525	574.5	652	447.5

All engines were measured in the same rotation speed of $720rpm$. Rotation speed is fixed because the frequency of the generator is dependent on its rotation speed. Differently from the rotation, the load for each engine could be adjusted, and measures were taken in 3 different load conditions: 50%, 75% and 100%.

Every engine had each one of its 20 cylinders measured, and 4 seconds of signal acquisition length were acquired. Since there is one combustion cycle every two rotations of the crankshaft, and the engine was running at $720rpm$, which corresponds to $12Hz$, there are 6 combustion cycles per second of signal. Once 4 seconds are measured, there are a total of approximately 24 cycles per signal. For 20 cylinders, in 3 load conditions, for 5 different engines, a total number of 7200 cycles were measured.

Figure 4.3 illustrate one measurement of the raw signals acquired simultaneously: in blue is the vibration signal measured on the engine's block; in red is the in-cylinder pressure; in black is the rotation signal, with a pulse for every crankshaft revolution. Figure 4.3 also shows a zoom view of one of the combustion cycles observed in the signals presented.

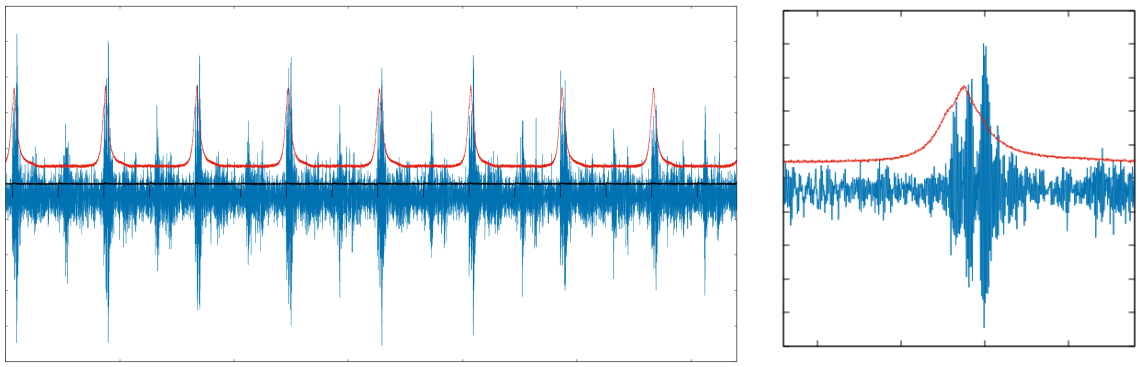

Figure 4.3: Waveforms acquired: vibration, in-cylinder pressure and the tachometer signals.

With the tachometer signal and the engine geometry shown in table 4.2 it is possible to retrieve the relative crankshaft position for each cylinder, because vibration and pressure signals are acquired simultaneously. The combustion cycle for every cylinder is centered in the Top Dead Center (TDC).

The algorithm for the combustion cycle determination and dataset preparation from the experimental data measured is presented in algorithm 1.

Algorithm 1: Dataset Preparation

Input: cylinder_angles ▷ TDC phase from reference as table 4.2 shows
Output: dataset ▷ samples of combustion cycles

Function `ParseData`(*cylinder_angle*)**:**
- dataset[] ← *nil*
- **for** *engine* ← 1 **to** 5 **do**
  - **for** *cylinder* ← 1 **to** 20 **do**
    - $acl, prs, tch \leftarrow ReadData(engine, cylinder)$ ▷ retrieve the 4 seconds length signals
    - $i \leftarrow 0$
    - **while** $i < length(tch)$ **do**
      - $i \leftarrow findNextPulse(i, tch)$ ▷ find the first tacho peak after i
      - $cylinder\_ref = i + 2000 * cylinder\_angle[cylinder]/360$
      - $acl\_cycle = acl[cylinder\_ref - 1000 : cylinder\_ref + 1000]$
      - $prs\_cycle = acl[cylinder\_ref - 1000 : cylinder\_ref + 1000]$
      - $dataset[] \leftarrow [acl\_cycle, prs\_cycle]$
    - **end**
  - **end**
- **end**
- **return** dataset[]

For example, figure 4.3 shows the measurement for cylinder A1, which has a zero degree phase in relation to the reference read by the tachometer. Because of that, the tachometer pulse corresponds to the exact position of cylinder A1 TDC. The rotation speed for engine is fixed in $720rpm = 12Hz$, and data were acquired with $24KS/s$. To slice the combustion cycle out of the measured signals one should take $24000/12 = 2000$ samples, 1000 before the tachometer pulse, and 1000 after the tachometer pulse, as can be observed in the zoom view of the combustion cycle in figure 4.3.

For others cylinders, like A2 for example, the tachometer pulse has 232.5 degrees of phase in relation to its TDC, from table 4.2. As calculated before, 2000 samples corresponds to a cycle of combustion, or 360 degrees. So, before slicing the combustion cycle, one should first shift the TDC reference in $2000 * 232.5/360$ samples for cylinder A2.

Every cycle, after synchronization would look like the one seen in figure 4.3. Note that, every combustion cycle is composed by the pair: a vibration signal x_t and a pressure signal p_t.

Once all cycles of combustion are retrieved by the algorithm proposed, the architecture models can be trained, as described in the following sections.

4.1.2 Monitoring System Architecture

The architecture for the internal combustion monitoring system, with pressure reconstruction and combustion parameter estimation proposed is presented in figure 4.4.

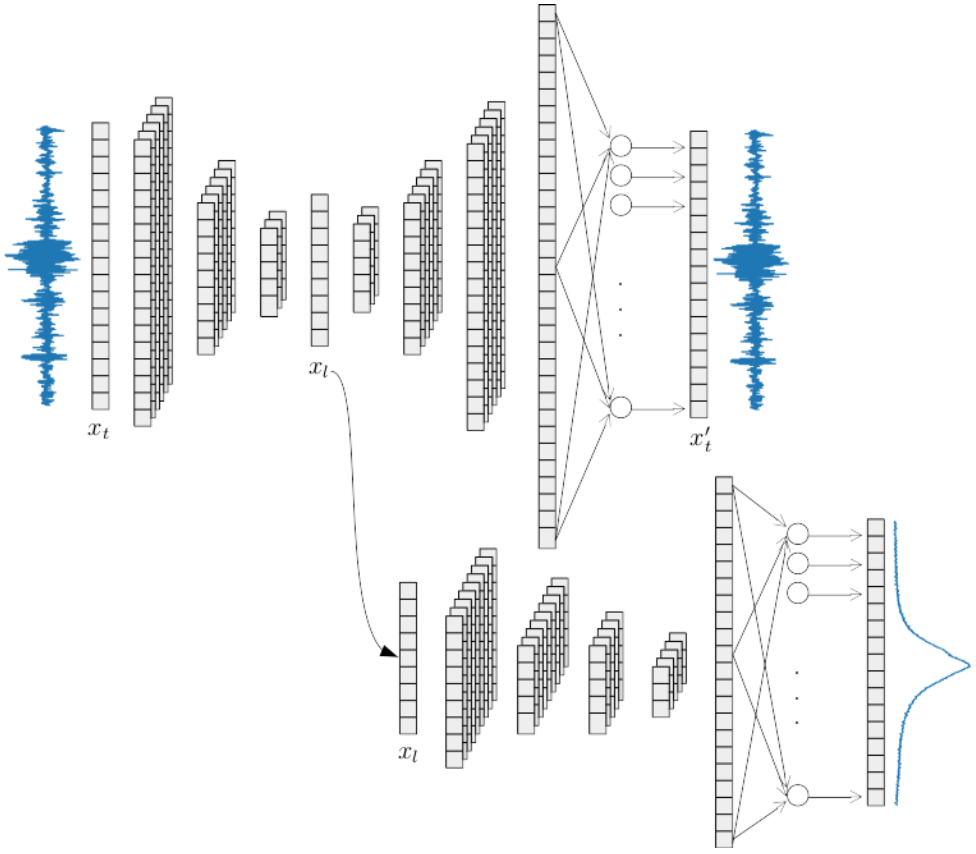

Figure 4.4: WCAE Monitoring System Architecture

The architecture presented in figure 4.4 is composed by three different phases:

1. The encoder phase of the autoencoder, that is responsible for robust feature extraction, and vibration data compression. It takes the original vibration signal x_t and compress it by a sequence of wavelet convolutional layers to a signal x_l. In this phase, the model proposed has a limited number of filters and their size are lowered to meet restrictions of hardware for embedded sensors;

2. The decoder phase of the autoencoder, that reconstructs the original vibration signal x_t. In the proposed model, x_t is compressed to a signal x_l, with smaller dimension than x_t. Than, the inverse sequence of wavelet convolutions, followed by a densely connected neural network are applied to reconstruct x'_t;

3. The deep wavelet convolutional neural network (DWCNN), responsible for pressure reconstruction and combustion parameter estimation. The DWCNN proposed takes as its input the intermediate layer of the autoencoder, that is the compressed form of the original vibration data (x_l). The output of this convolutional neural network (p'_t) is compared to the pressure signal (p_t), measured synchronously with x_t, in the same combustion cycle.

The autoencoder is trained first, and separately from the DWCNN. Each vibration sample, measured at the engine's block is used both as input and desired output for the autoencoder. The goal is to minimize the mean average error of reconstructing the original

vibration signal. As can be observed in figure 4.4 and in section 4.1.3 of the WCAE model dimensions, the intermediate layer of the autoencoder has much fewer samples compared to the original vibration signal.

The autoencoder training is finished when the model achieves a sufficient low loss value, with $E_{mae} < \epsilon$, where E_{mae} is the Mean Average Error metric defined in equation 3.4 and ϵ is a tolerance limit.

After the autoencoder training, then the convolutional neural network model proposed for the pressure reconstruction and combustion parameter estimation is trained. It takes the intermediate layer of the autoencoder as its input, for each vibration signal measured (x_t) and compressed by the autoencoder (x_l). The goal for the DWCNN training is to minimize the same metric of Mean Average Error, but calculated with its output (p'_t) in comparison to the pressure signal (p_t) acquired simultaneously to the original vibration sample (x_t).

The edge computing based monitoring architecture presented in figure 3.1, proposes that the data acquisition and data manipulation should be embedded in the sensor. In the model proposed in figure 4.4, the encoder phase of the autoencoder should be embedded in the sensor firmware, while the convolutional neural network for pressure reconstruction and parameter estimation should be in top level applications. To be able to embed in the sensor firmware, the model dimensions should respect the restrictions of the hardware capacity, as detailed in the next section.

4.1.3 Model Dimensions

The encoder part of the WCAE, presented in figure 4.4, has four consecutive combinations of wavelet convolutions and max-pooling layers. The first convolutional layer have 8 filters, while the others 3 layers have 4 filters, all of them with 3 values length. The max-pooling layers have downsize factor of 2. For the encoder phase and the decoder phase to be completely symmetrical, the input combustion cycle is limited to 1748 samples. That happens because the dimensions are truncated when downsized and up-sampled in the autoencoder layers.

As described before, the decoder part of the WCAE is symmetrical to the encoder phase, intercalating the four convolutions with up-sampling layers, with up-size factor of 2. These layers are flattened to an array of 6672 samples. This array is passed for one hidden layer of a fully connected network with 1748 nodes using $ReLU$ activation function. All convolutional layers, both from the encoder and the decoder part, have the wavelet activation function as proposed in section 3.2.

Figure 4.5 that follows shows in details the model dimensions explained before.

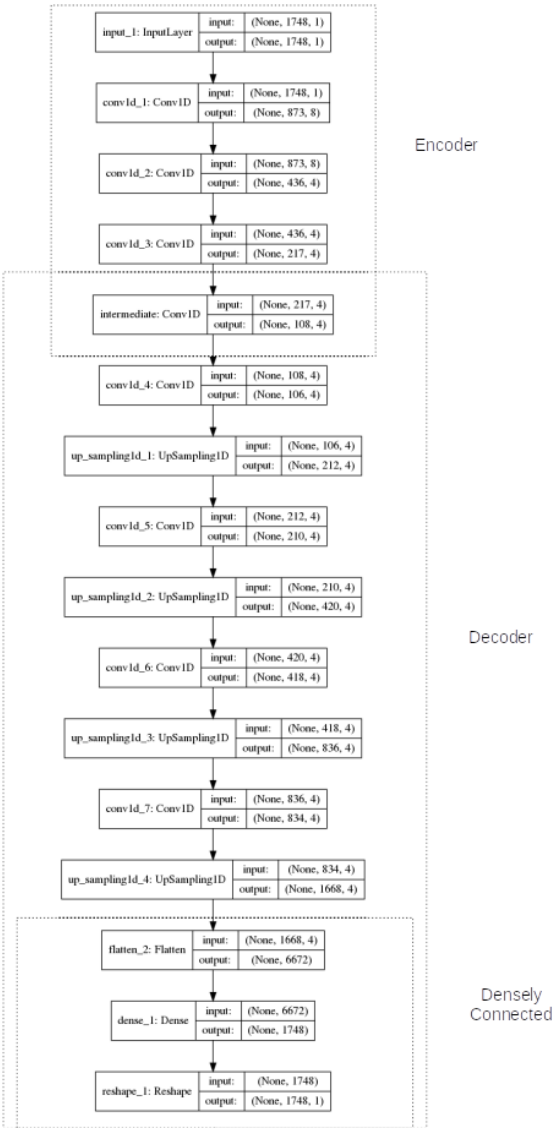

Figure 4.5: WCAE Model Dimensions

For the encoder phase, the small number of filters and their small size, results in a total of only 236 trainable parameters (convolution filter values). As discussed in section 3.5, each input value and each parameter uses a 16 bit representation, or 2 bytes. Calculating the memory usage in each layer we have:

- $1748 \cdot 2 = 3496$ bytes for the vibration input signal;
- $8 \cdot 873 \cdot 2 = 13968$ bytes for the first convolution and max pooling layer;
- $4 \cdot 436 \cdot 2 = 3488$ bytes for the second convolution and max pooling layer;
- $4 \cdot 217 \cdot 2 = 1736$ bytes for the third convolution and max pooling layer;
- $4 \cdot 108 \cdot 2 = 864$ bytes for the fourth convolution and max pooling layer;

Adding to the memory needed for signal representation $236 \cdot 2 = 472$ bytes for the model weights manipulation and $1000 \cdot 2 = 2000$ bytes for the wavelet static array storage, the

total memory usage for the encoder phase is of 26024 bytes. That is a very feasible value, since normally the vibration sensor have at least 32KB of RAM memory, as discussed in section 3.5.

The deep wavelet convolutional neural network proposed in figure 4.4 has dimensions explained in details in figure 4.6 that follows.

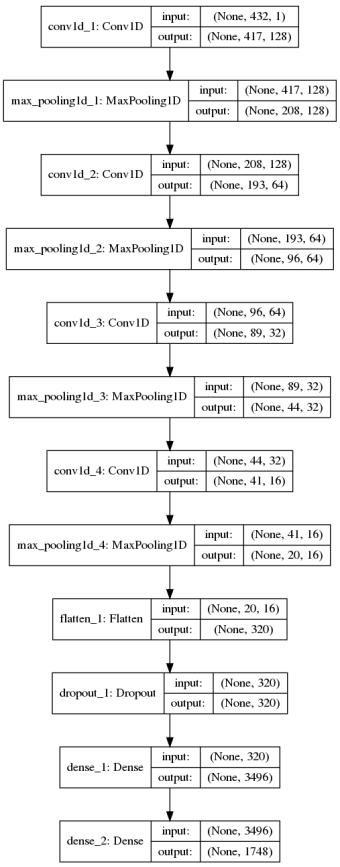

Figure 4.6: DWCNN model dimensions

For pressure reconstruction and parameter estimation, the DWCNN has a sequence of four convolutional layers, that have 128 filters with 16 values length, than 64 filters also with 16 values length, than 32 filters with 8 values length and than 16 filters with 4 values length. The convolutional layers are intercalated by max-pooling layers with downsize factor of 2. The result of these layers are than flatten to an array and passed to a two hidden layers of fully connected network with $ReLU$ activation function, containing 3496 and 1748 nodes, respectively.

As discussed before, there are two distinct phases of training: first the WCAE training, and then the model training of the DWCNN proposed for pressure reconstruction and parameter estimation.

The next sections will present: the results of the autoencoder training in 4.1.4; the results for pressure reconstruction in 4.1.5; and the results for combustion parameter estimation in 4.1.6.

4.1.4 Results for the WCAE

The autoencoder model, proposed in the architecture presented in figure 4.4, was trained with all the 7200 cycles of combustion retrieved from the signals acquired. As discussed before, the model takes as input and output the vibration signal x_t measured from engine block.

As discussed in section 3.3, the loss function designed for the model training is the the Mean Average Error metric, defined in the equation 3.4.

For a better understanding on the autoencoder results efficiency, some other metrics will also be calculated. The error metrics used to compare the results from the autoencoder are:

- MAE (*Mean Absolute Error*) - Represents the mean of the absolute difference from the reconstructed vibration signal (x'_t) in comparison to the real vibration measurement (x_t). The result is expressed in g's, the unit of the accelerometer readings. This metric should be minimized.

$$MAE = \frac{\sum_{i=1}^{n} |x_t - x'_t|}{n} \tag{4.1}$$

- R^2 (Coefficient of Determination) - Represents the proportion of variation from the reconstructed vibration signal (x'_t) in comparison to the real vibration measurement (x_t). From the formulation in equation 4.2, we can see that the closer the R^2 metric is to the value of 1, more alike are the signals x_t and x'_t.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (x_t - \overline{x_t})^2}{\sum_{i=1}^{n} (x_t - x'_t)^2} \tag{4.2}$$

- RMSp (*Root Mean Square proportion*) - Represents the proportion of the root mean square of the reconstructed vibration signal (x'_t) over the the real vibration measurement (x_t). This metric is calculated in percentage. So the expected value is to be closer to 100% if the signals have the same energy.

$$RMSp = 100 \cdot \frac{\sqrt{(\sum_{i=1}^{n} {x'_t}^2)/n}}{\sqrt{(\sum_{i=1}^{n} {x_t}^2)/n}} \tag{4.3}$$

- PKPKp (*Peak to Peak proportion*) - Represents the proportion of the peak to peak of the reconstructed vibration signal (x'_t) over the the real vibration measurement (x_t). This metric is calculated in percentage. So the expected value is to be closer to 100% if the signals have the peak amplitude.

$$PKPKp = 100 \cdot \frac{max(x'_t) - min(x'_t)}{max(x_t) - min(x_t)} \tag{4.4}$$

To train the autoencoder model, we use a ten-fold cross validation, with ten percent split. The autoencoder training algorithm is presented in algorithm 2.

Algorithm 2: Training WCAE Model

Input: $dataset[], epochs$ ▷ combustion cycles dataset and epochs of training
Output: $model$ ▷ trained model

Function `TrainModel`$(dataset[], epochs)$**:**
  for $seed \leftarrow 1$ **to** 10 **do**
    $X_{train}, X_{test}, y_{train}, y_{test} \leftarrow SplitSamples(dataset[], 0.1, seed)$ ▷ 10% split
    $model \leftarrow FitModel(X_{train}, X_{test}, y_{train}, y_{test}, epochs)$
    $mae, r2, rmsp, pkpkp \leftarrow CalcMetrics(model, X_{test}, y_{test})$ ▷ from equations 4.1 to 4.4
  end

One of the ten fold cross validation history of training is presented in figure 4.7. It shows the loss as the blue line and validation loss as the orange line, per epoch of training.

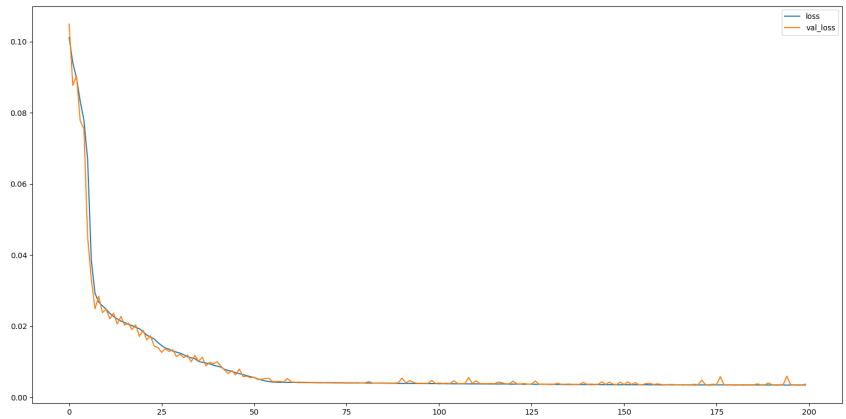

Figure 4.7: Training history in one validation round.

Table 4.3 shows the metrics results for every round in the 10-fold cross validation as defined in algorithm 2.

Table 4.3: Autoencoder results for 10-fold cross validation.

fold	MAE		R^2		RMSp		PKPKp	
	mean	std dev	mean	std dev	mean	std dev	mean	std dev
1	0.675777	0.265737	0.997808	0.003057	0.999626	0.000434	0.962248	0.027484
2	0.662832	0.252267	0.997877	0.002768	0.999846	0.000439	0.96155	0.028529
3	0.655948	0.260347	0.997899	0.0028	0.999769	0.000457	0.958415	0.028697
4	0.670507	0.266278	0.997806	0.003073	0.999896	0.000398	0.970157	0.027171
5	0.666399	0.260181	0.997834	0.002988	0.999732	0.000414	0.964852	0.027522
6	0.656626	0.259312	0.997882	0.002844	0.999804	0.000444	0.9602	0.028503
7	0.650971	0.257196	0.997906	0.002782	0.999817	0.000428	0.962424	0.028606
8	0.654256	0.257582	0.997897	0.002819	0.999821	0.000434	0.960986	0.02842
9	0.659446	0.262338	0.997848	0.002975	0.999857	0.0004	0.966618	0.027435
10	0.658894	0.261195	0.997857	0.002936	0.999802	0.000423	0.963829	0.027534

Table 4.4 shows the average mean and standard deviation of vibration reconstruction in each error metric chosen, after the ten-fold cross validation test, with ten percent split.

Table 4.4: Autoencoder results.

Error Metric	mean	std dev
MAE [g]	0.7	0.3
R^2	0.998	0.003
RMSp [%]	99.98	0.04
PKPKp [%]	96.3	2.8

The results in table 4.4 show that the autoencoder proposed succeeded in compressing the information from the vibration signal used as input to an intermediate layer with less than a quarter of the size of the input. The vibration signals used, that corresponds to a cycle of combustion as showed in figure 4.3, have dimension of 1748 samples. The intermediate layer of the autoencoder proposed has two arrays of size 216, that results in 432 samples.

Examples of vibration signals reconstructed by the WCAE are presented in figure 4.8.

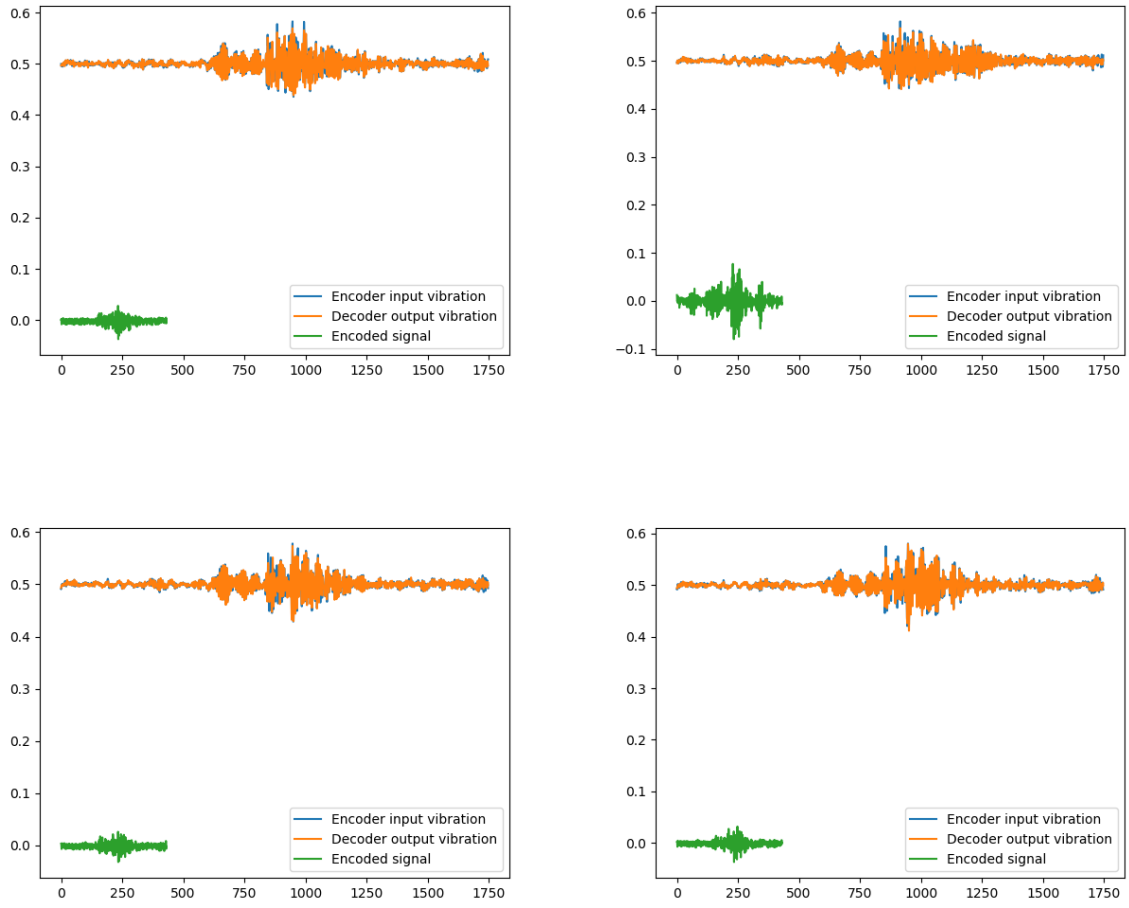

Figure 4.8: Some Autoencoder results examples.

4.1.5 Results for Pressure Reconstruction

The architecture proposed in this work is a Wavelet Convolutional AutoEncoder (WCAE) for in-cylinder pressure reconstruction and combustion parameter estimation, using raw

vibration data as input. In this section, the results of a Deep Wavelet Convolutional Neural Network (DWCNN) is presented for the reconstruction of the pressure signal, while next section shows the results for parameter estimation. The pressure reconstruction uses the features extracted from the intermediate layer of the autoencoder trained for the last section.

The combustion cycles retrieved from the waveforms acquired, as shown in figure 4.3, are used to calculate the results for the DWCNN. The error metrics used to measure the accuracy of the model are:

- MAE (*Mean Absolute Error*) – Represents the mean of the absolute difference from the reconstructed pressure (p'_t) in comparison to the real pressure acquired (p_t). The result is expressed in bar, the unit of the pressure readings.

$$MAE = \frac{\sum_{i=1}^{n} |p_t - p'_t|}{n} \tag{4.5}$$

- R^2 (Coefficient of Determination) – Represents the proportion of variation from the reconstructed pressure in comparison to the real pressure acquired.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (p_t - \overline{p_t})^2}{\sum_{i=1}^{n} (p_t - p'_t)^2} \tag{4.6}$$

For comparison purpose, the same DWCNN is used to reconstruct pressure signal, using the original (x_t) vibration and the features extracted from the intermediate layer of the autoencoder (x_l). The number of hidden layers, filter size, pooling factor, gaussian wavelet activation function and dense layers size are the same. The difference between them are the data dimensions only, as seen in figure 4.9.

The DWCNN training algorithm is presented in algorithm 3. As described in section 4.1.1, five different engines were measured.

Algorithm 3: Training DWCNN Model

Input: $dataset[], encoder, epochs$ ▷ combustion cycles, encoder model and epochs of training
Output: $model\_orig, model\_interm$ ▷ trained model with x_t and x_l

Function `TrainModel(`$dataset[], encoder, epochs$`):`
  for $engine \leftarrow 1$ **to** 5 **do**
    $X_{train}, X_{test}, y_{train}, y_{test} \leftarrow SplitSamples(dataset[], engine, seed)$ ▷ select engine for test
    $model\_orig \leftarrow FitModel(X_{train}, X_{test}, y_{train}, y_{test}, epochs)$
    $mae, r2 \leftarrow CalcMetrics(model, X_{test}, y_{test})$ ▷ from equations 4.5 and 4.6
    $Xf_{train} \leftarrow encoder(X_{train})$ ▷ encoder model trained before
    $Xf_{test} \leftarrow encoder(X_{test})$ ▷ encoder model trained before
    $model\_orig \leftarrow FitModel(Xf_{train}, Xf_{test}, y_{train}, y_{test}, epochs)$
    $maef, rfs2 \leftarrow CalcMetrics(model, Xf_{test}, y_{test})$ ▷ from equations 4.5 and 4.6
  end

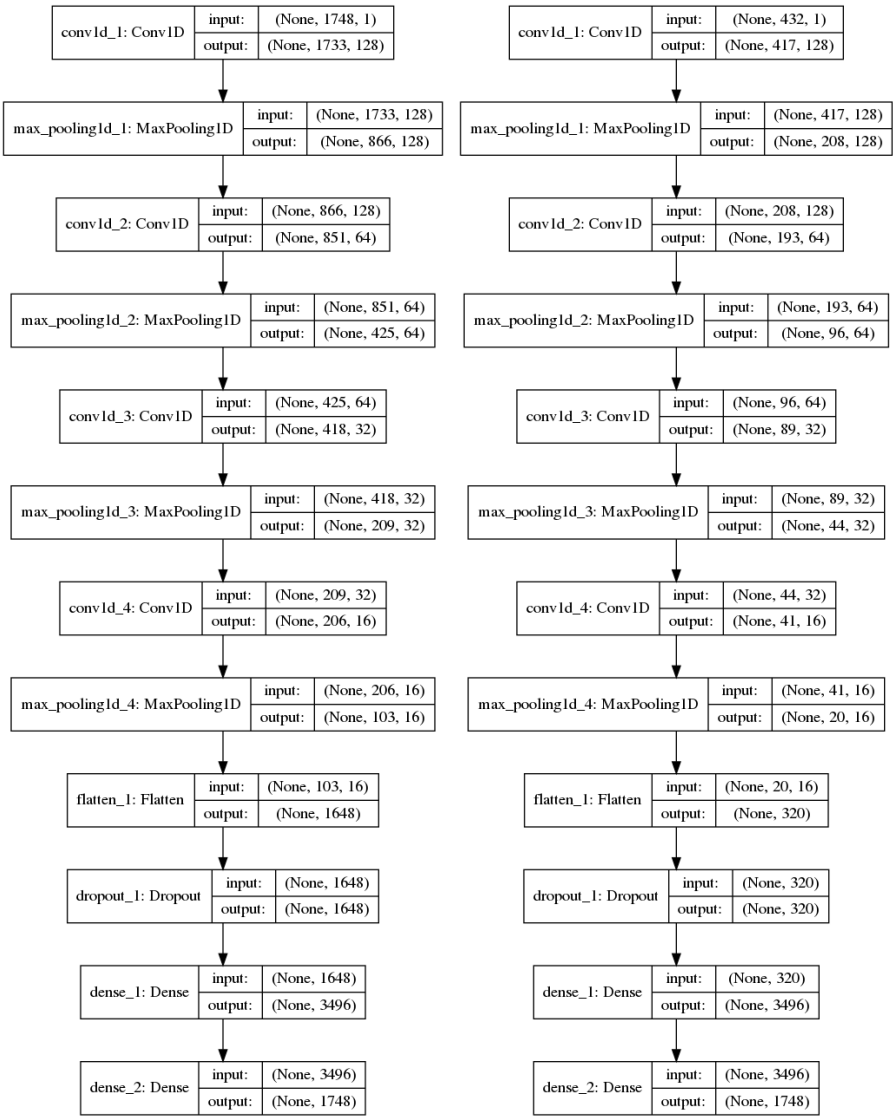

Figure 4.9: Dimensions comparison in the DWCNN architecture.

To calculate the results for pressure reconstruction, the training dataset were build with four engines, while the testing dataset with the the fifth engine measurements. This strategy is repeated 5 times, choosing one different motor as the testing dataset in each round.

Table 4.5 shows the metrics results for every round of training as defined in algorithm 2 for the model trained from original vibration signal.

Table 4.5: DWCNN training results from original vibration.

round	MAE		R^2	
1	0.718084	0.530003	0.994099	0.016952
2	0.644886	0.939123	0.992081	0.066757
3	0.590362	0.727071	0.994229	0.042679
4	0.727145	0.502590	0.994492	0.012976
5	0.502051	0.868839	0.994561	0.062219

Table 4.6 shows the metrics results for every round of training as defined in algorithm 2 for the model trained from the compressed vibration signal.

Table 4.6: DWCNN training results from compressed vibration.

round	MAE		R^2	
1	0.610418	0.516385	0.994839	0.012059
2	0.623506	0.980883	0.992124	0.073962
3	0.608474	0.729459	0.994057	0.043897
4	0.605140	0.471459	0.995161	0.010500
5	0.626747	0.901324	0.993344	0.063098

Table 4.7 shows the average mean and standard deviation for pressure reconstruction in each error metric chosen.

Table 4.7: Pressure reconstruction errors

	from x_t		from x_l	
Error Metric	mean	std dev	mean	std dev
MAE [bar]	0.63	0.71	0.61	0.71
R^2	0.99	0.04	0.99	0.04

Since data were paired for the results with the DWCNN from original vibration and from the intermediate layer, and each pair was chosen randomly and independently, the Wilcoxon signed-rank test were calculated over the error metrics. The null hypothesis for the Wilcoxon test states that the difference between the pairs follows a symmetric distribution around zero.

Calculating the p-value for MAE and R^2, we have p-value$_{MAE} = 0.68$ and p-value$_{R^2} = 0.89$. Since those are high values, we fail to reject the null hypothesis. This statistical test shows that there is no considerable difference of using the original vibration signal or the encoded signal for the pressure reconstruction problem.

The result for the Wilcoxon test also shows that there is no significant loss of information with using the compressed vibration signal (x_l) from the intermediate layer in comparison to using the original vibration signal (x_t) for pressure reconstruction.

Some examples of reconstructed pressure signals are presented in figures 4.10,4.11 and 4.12.

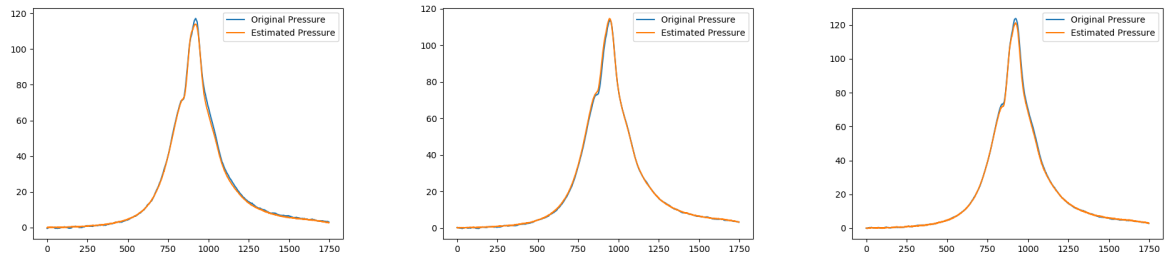

Figure 4.10: Pressure reconstruction results - 50% load.

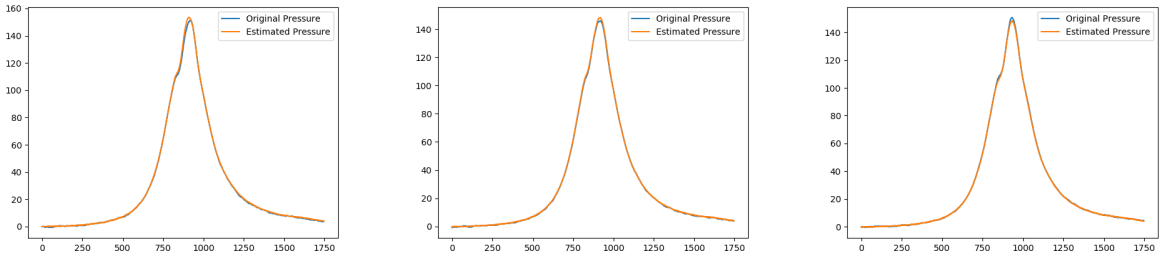

Figure 4.11: Pressure reconstruction results - 75% load.

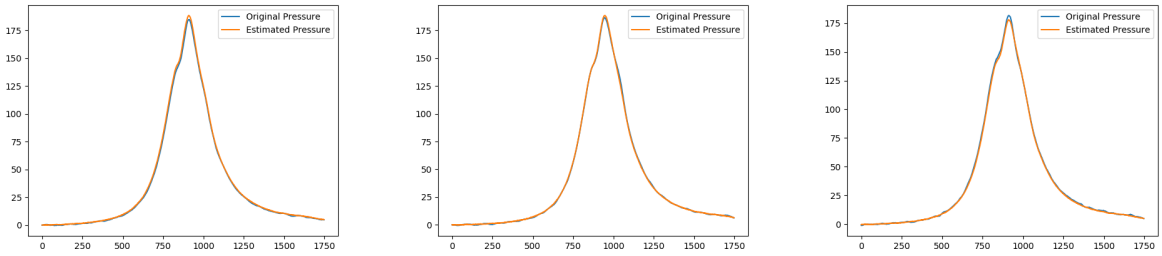

Figure 4.12: Pressure reconstruction results - 100% load.

4.1.6 Results for Combustion Parameter Estimation

Besides pressure reconstruction, combustion parameters such as maximum pressure peak and its correspondent angle during the combustion cycle can be calculated from the measured pressure signal and compared with the estimated pressure curve calculated from the model trained in the last section. To analyze the results for parameter estimation, the following metrics are used:

- $P_{max}AE$ (Pmax Absolute Error) - Represents the absolute difference from the maximum value of the reconstructed pressure (p'_t) in comparison to the maximum value of the real pressure acquired (p_t).

$$P_{max}AE = \frac{\sum_{i=1}^{n} |max(p_t) - max(p'_t)|}{n} \tag{4.7}$$

- $P_{max}RE$ (*Pmax Relative Error*) – Represents the relative error from the difference of maximum value of the real (p_t) and reconstructed pressure (p'_t) over the maximum value of the real pressure acquired (p_t).

$$P_{max}RE = \frac{\sum_{i=1}^{n} |max(p_t) - max(p'_t)|/max(p_t)}{n} \tag{4.8}$$

- $P_{ang}AE$ (*Pangle Absolute Error*) – Represents the absolute difference from the location in angles of the maximum value from the reconstructed pressure (p'_t) in comparison to the location in angles of the maximum value from the real pressure acquired (p_t).

$$P_{ang}AE = \frac{\sum_{i=1}^{n} |argmax(p_t) - argmax(p'_t)|}{n} \tag{4.9}$$

The functions max and $argmax$ are functions that calculate the maximum and location of the maximum values of a signal. The results for the metrics defined before are presented in table 4.8. They were calculated separately from the engines load during the signals acquisition, and also with all loads computed together.

Table 4.8: Combustion parameters estimation

	50% load		**75% load**		**100% load**		**Total**	
Parameter	mean	std dev	mean	std dev	mean	std dev	mean	std dev
$P_{max}AE$ [bar]	1.2	0.1	0.88	0.06	1.5	0.1	1.22	0.08
$P_{max}RE$ [%]	0.95	0.08	0.59	0.04	0.89	0.07	0.81	0.05
$P_{ang}AE$ [^{0}]	0.57	0.02	0.59	0.02	0.48	0.08	0.53	0.02

The results in table 4.8 shows that absolute error on the maximum value of pressure are low, corresponding to less than one percent of the peak value. Accuracy to determine the location from the maximum peak of pressure is also high, around of half degree of deviation from the real location.

In figure 4.13 the estimated versus the real maximum value of pressure peaks can be observed.

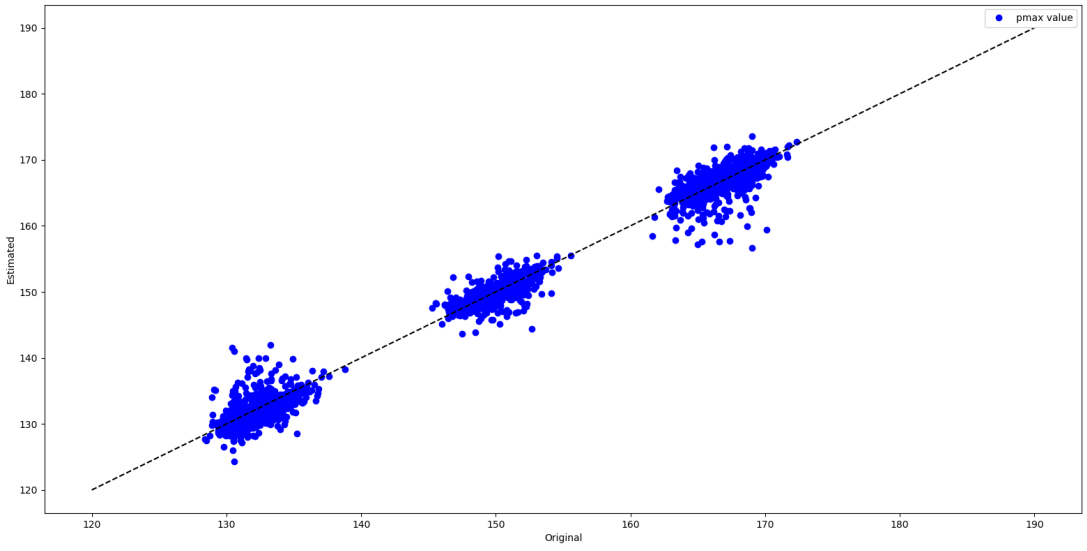

Figure 4.13: Pressure peak values from real and estimated pressure signals.

From figure 4.13, the x axis represent the original pressure (p_t) maximum peak, and in the y axis the reconstructed pressure (p'_t) maximum peak. The dotted line represents the optimal result. The fact that the resulting dots are distributed close to the dotted line, confirms the good results presented in table 4.8.

It is also interesting to see the maximum peak values form three distinct agglomerations, that are relative to the three load conditions acquired. That shows that the good model results are load invariant.

In figure 4.14, the estimated and real location of maximum peaks are shown.

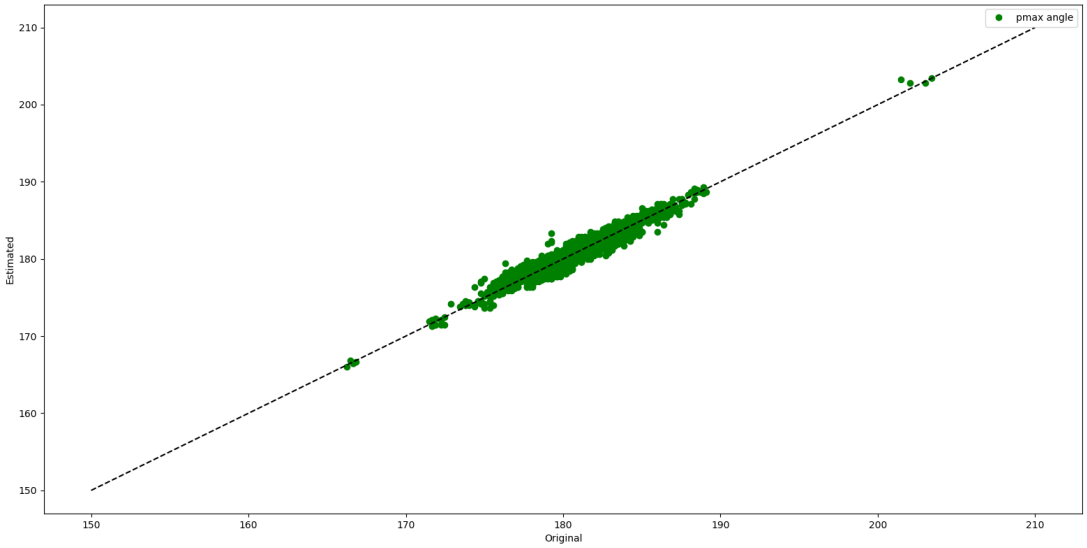

Figure 4.14: Pressure peak angles from real and estimated pressure signals.

From figure 4.14, the x axis represent the original pressure (p_t) maximum peak angle, and in the y axis the reconstructed pressure (p'_t) maximum peak angle. The dotted line represents the optimal result. The fact that the resulting dots are distributed close to the dotted line also confirms the good results presented in table 4.8.

The results for combustion parameter estimation can be compared to the literature. In [9], instead of the maximum peak they estimate the position where 50% of fuel mass was burned over an engine cycle is reached (MFB50). They estimate in a small three cylinder engine, for six different rotation speeds, and got 0.77 ± 0.24 degrees of deviation for their estimation.

In [10] they also estimate the MFB50, and absolute error on the angle estimation was calculated using the root mean squared error RMSE metric. The value obtained in this article was of 0.0743 degrees.

In [28] the maximum peak of pressure is estimated for high power combustion engines. The error in percentage for 6 load conditions were: -1.98, -2.85, 3.08, -0.86, -4.93, 2.53. In average the percentage error is 2.7 ± 1.3, which is higher than the results obtained in this work. The DWCNN proposed in this thesis is more accurate for parameter estimation.

In [20] the maximum pressure peak and its location is also estimated in a four-cylinder diesel engine, for three different rotation speeds. Results for P_{max} estimation, in percentage, were: 5.1 ± 0.6, 4.4 ± 0.4 and 0.9 ± 0.4. The results for P_{ang} estimations, in degrees, were: -1.05 ± 0.17, 1.86 ± 0.61 and 0.64 ± 0.44. Again, our results have better accuracy.

4.1.7 Results Discussion

The pressure reconstruction results, presented in section 4.1.5, show that there is no significant loss of information when using the compressed vibration signal (x_l), from the intermediate layer of the Wavelet Convolutional Autoencoder (WCAE) proposed in this

thesis, in comparison to using the original vibration signal (x_t). Also, the model dimensions explained in section 4.1.3 show that the encoder phase of the proposed autoencoder has limited size parameters, feasible to be embedded in vibration sensor firmware, with limited hardware available.

Because of those conclusions, the architecture proposed in section 4.1.2, that defines the monitoring system architecture for internal combustion engines, can be implemented following the characteristics presented in section 3.1, that presents the edge computing based monitoring system proposed in this thesis, based on the Open System Architecture for Condition Based Monitoring (OSA-CBM), illustrated in figure 2.1.

Once the WCAE for the vibration signal compression and the DWCNN for pressure reconstruction and combustion parameter estimation are both trained, the monitoring system can be implemented as the following figure 4.15 illustrates.

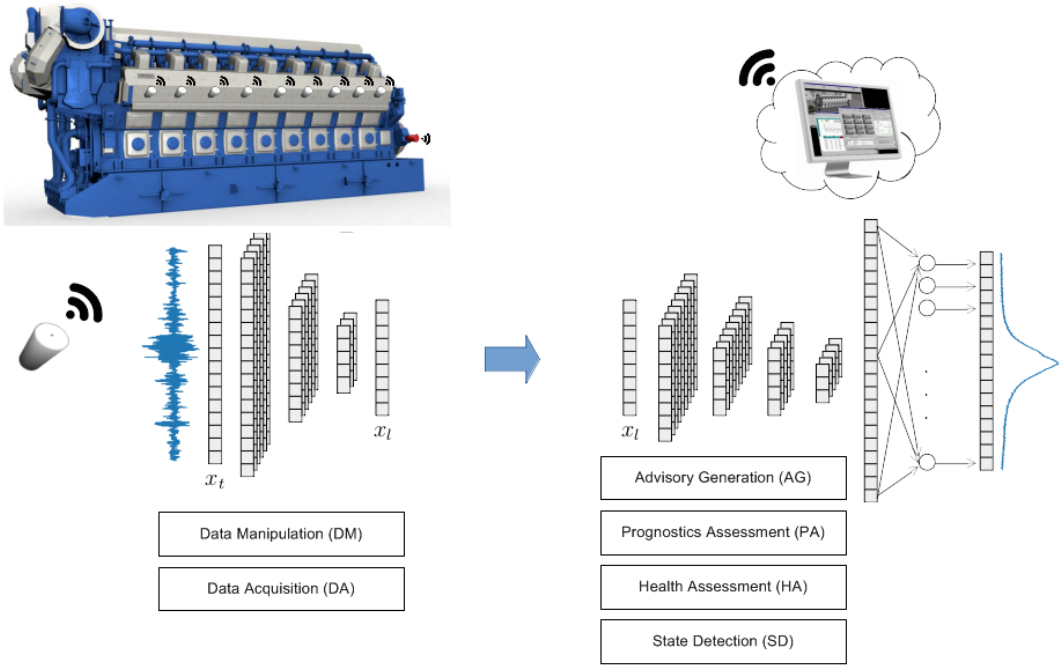

Figure 4.15: Pressure monitoring System.

The in-cylinder pressure reconstruction from raw vibration data proposed in this thesis is a robust way of indirect measure a parameter widely used in evaluating the combustion process. Due to the smaller cost of sensors used for this indirect measurements, their considerably longer life-time, the robustness and the accuracy of the methodology presented, is possible to constantly monitor large power generators, preventing breakdowns, excessive fuel consumption and pollutant emission.

Let us calculate an estimative of the benefits on adoption of the technology proposed in this thesis for the power generation plant in Brazil, where the experiments were carried out. As discussed in section 4.1.1 of experimental setup, this plats has 38 Wärtsilä W20V32 engines that run with fuel oil type B1.

Considering an average value of US$ 2000 for the pressure transducer, and that each engine has 20 cylinders, the total cost of instrumentation would exceed US$ 1.5 million

using pressure transducers only. Considering the results presented in this project, allowing the replacement of this sensor by vibration accelerometers, which cost up to a maximum of US$ 200, the cost of instrumentation would fall to around US$ 150000.

In addition to the cost of instrumentation, the life of the monitoring system using vibration sensors is much longer. This would enable a return on investment prior to system degradation, which would not be possible using pressure transducers.

Once continuous motor monitoring is possible, any malfunctions that would lead to an unexpected failure and consequently an unscheduled shutdown of the equipment can be avoided. A break would impact in large maintenance costs and a possible power outage to the community that depends on the power generation.

In addition to avoiding unplanned shutdowns, the precise estimation of combustion parameters, such as the peak maximum and its location, are an important feature to detect if the engine is tuned for good fuel consumption. Deviation from the expected TDC implies that the engine is not optimized for consumption.

Let us assume that the continuous monitoring could reduce in 1% the fuel consumption in the power generation plant. Given that the daily consumption of the plant is about 1600 tonnes of fuel oil for the 38 engines, each engine consumes an average of 42 tonnes per day. The specific density of the fuel oil used (OCB1) is 0.98 g/cm^3 [1]. Therefore the consumption of an engine is about 43 thousand liters per day. One hundredth of that is 430 liters a day. Given the cost of US$ 1.8 a gallon (3.8 liters) [2], the savings would be around US$ 203 a day. With a hundredth increase in efficiency for a single engine, the return on instrumentation investment would be after 1 year and 10 months.

But if we take a closer look to the figure 4.14, it is possible to see that there are cylinder with high deviation from its expected location. That makes us believe that the return on the investment could be much faster than our estimation.

[1] http://www.abq.org.br/cbq/2015/trabalhos/4/8288-21602.html – Acessado em dez/2017

[2] http://www.indexmundi.com/pt/pre%E7os-de-mercado/?mercadoria=%c3%b3leo-combust%c3%advel&meses=60 – Acessado em dez/2017

4.2 Industrial Rotating Machinery

Rotating machinery play an important role in industry, driving all kinds of processes in many industry segments such as: chemical, foundry, food, power generation, stamping, etc. Being able to early detect faults in these components can not only improve their reliability, but also avoid catastrophic damage, big economic losses and even great environmental disasters. In this way, vibration analysis is the most effective technique to detect mechanical defects in rotating machinery [42].

Fault detection is a binary classification problem, where false positives and false negatives does not have the same cost for real applications. False determining that a component is defective, will incur in a scheduled stop in the industry process, to unnecessary maintenance, which is bad. But, the opposite scenario, diagnosing a faulty component as normal, will lead to an unexpected failure, with big economic losses. Besides the economic impact, depending on what segment the industry operates in, a breakdown can lead to even more drastic consequences, including safety risks.

Automatic fault diagnosis involves pattern recognition through a classification problem from features extracted out of vibration measurements [56]. As discussed in 2.4, feature extraction, from more conventional machine health monitoring systems, were heavily based on specialists expertise or heavy signal processing techniques. More recently, as discussed in section 2.5, deep learning models have being developed for feature extraction, instead of classical feature-engineering based methods.

The deep learning model proposed in this thesis is a Wavelet Convolutional Neural Network (WCNN), based on the generic model described in section 3.2. The edge computing based system for rotating machinery health monitoring, based on the architecture presented in 3.1, takes the vibration signal measured by the sensor and extract relevant features for fault detection. The aim of the WCNN is to reduce the amount of data transmitted by the sensor, without loosing information for the fault classification performed in the gateway.

The proposed method is applied to data measured in real equipments from industries around the world. The experimental data and its preprocessing is explained in the next section. The results confirm that feature extraction and fault detection are performed with great success. The WCIC proposed is a new classifier generally applicable for many different equipment types and industry segments.

Before explaining the architecture of the Wavelet Convolutional Imablanced Classifier, the algorithms developed for training, and the results and discussions on performance achieved, the experimental setup together with the data measurements and procedures are presented in the next section.

4.2.1 Experimental Data

The data used for the experiments reported in this thesis was retrieved from a large database of a Predictive Maintenance Company in Brazil, called SEMEQ. This company implements the health monitoring system of many industries around the world, executing not only vibration, but also oil, electric, thermographic, and ultra-sound analysis on many

different types of rotating machinery. It operates in the predictive maintenance field from 25 years. The experimental data, retrieved from the database, consists of measurements executed from July of 2018 until March of 2019.

The most common types of components retrieved from the database are: Electric Motors, Reducers, Compressors, Gear Boxes and Servomotors. But many others more were considered, such as Pumps, Fans, Blowers, Turbines, etc. Around 35 thousands of components are measured in each month. But, in despite of all these measurements, the failure rate is smaller than 1%, which confirms the challenge for model training stated in section 2.5.1. The approximate average number of components with failures each month is around 300 samples only.

All these data, were manually analyzed by experts in vibration analysis, from the predictive maintenance program implemented by the companies. Their analysis will be used here as the ground truth for the database extracted data. To illustrate, the different kinds of failures diagnosed and the total number of samples per month, are summarized in table 4.9.

Table 4.9: Failures in datasets.

	Aug 18	Sep 18	Oct 18	Nov 18	Dec 18	Jan 19	Feb 19	Mar 19
Looseness	216	145	211	175	167	145	184	178
Imbalance/Misalignment	2	2	1	0	0	4	6	3
Wear	31	25	19	23	35	28	27	44
Electrical	12	10	12	12	16	16	18	14
Rolling Element Bearing	25	17	18	18	26	20	30	33
Other	53	41	24	35	42	37	34	35
Total	39017	39707	34820	34864	34925	36635	36829	36529

The total amount of measurements retrieved from the database is of 293326 vibration samples over these 8 months of data. Due to the discussion in sections 2.5.1 and 3.4, of imbalanced datasets, a random under sampling of normal examples is proposed in this thesis. The data extraction and datasets composition is presented in Algorithm 4, that follows.

Algorithm 4: Dataset Generation

Input: month, year ▷ date of measurement
Output: train_dataset, test_dataset ▷ vibration samples datasets

train_dataset[] ← *nil*
test_dataset[] ← *nil*
Function `GetData`(*month,year*)**:**
  tmp ← *ReadData*(*AddMonth*(0,month,year)) ▷ retrieve all measurements for the ref. month
  test_dataset[] ← tmp ▷ all measurements are considered
  for $i \leftarrow 1$ **to** 6 **do**
    tmp ← *ReadData*(*AddMonth*(−i,month,year)) ▷ retrieve measurements from the past
    defect ← *FilterStatus*(*tmp*, 1) ▷ filter samples with status 1 (defective)
    normal ← *FilterStatus*(*tmp*, 0) ▷ filter samples with status 0 (normal)
    train_dataset[] ← defect
    train_dataset[] ← *RandomSelect*(normal, *limit*, $2 * size(defect)$)
  end
  return dataset[]

Algorithm 4 shows that, for fault detection of an equipment measured in March, for

example, the dataset samples are retrieved from the six months before (September to February). From this historical data, all defective samples are used, while only a portion of the normal samples are considered.

The random under sampling strategy is used, and it is one of the most simple algorithms for dataset balancing. No considerable advantage was notice in fault detection results on this application for more complex under sampling strategies, as the ones described in section 2.5.1. Although the random under sampling is performed for balancing the training dataset, we limited the dataset balancing to a proportion of a third of defective samples and two thirds of normal ones, as algorithm 4 shows. These decisions were empirically defined and no exhaustive comparison was made.

Data Processing

In the algorithm 4, the $ReadData()$ function retrieves from the database each sample measured in the reference month. The vibration sample is stored in the database in a waveform setup, with 4096 lines and $16384 KS/s$ of sampling rate. This setup, corresponds to 0.25 seconds of the vibration signal length. An example of a waveform sample can be seen in figure 4.16.

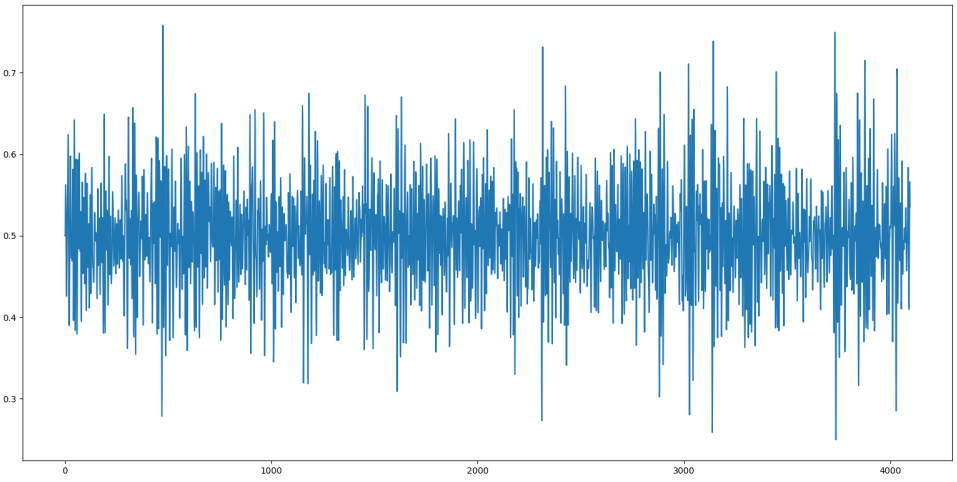

Figure 4.16: Waveform of Acceleration Sample

The data acquisition system used for vibration measurement uses an accelerometer with $100 mV/g$ of sensitivity, dynamic range of $\pm 50g$ peak and maximum shock protection of $5000g$ (peak). The accelerometer frequency range response with $\pm 3db$ is 0.5 to $15000 Hz$.

So, the waveform measured is an acceleration signal, and has its amplitude converted to g unit. However, in order to better visualize low frequencies for vibration analysis, the acceleration signal is usually integrated to a velocity representation.

The acceleration signal gives higher importance to high frequencies, what is useful specially for gearing condition assessment. In the other hand, velocity signals give higher importance to lower frequencies than the acceleration. Velocity is more related to the destructive force of vibration.

To convert acceleration to the velocity signal, an integration must be calculated. It is done by a numerical integration with Beeman's algorithm, based on the equation 4.10

that follows.

$$v(t + \Delta t) = v(t) + \frac{1}{6}(2a(t + \Delta t) + 5a(t) - a(t - \Delta t)) \tag{4.10}$$

The resulting velocity signal for the acceleration showed in figure 4.16 is presented in figure 4.17. The unit of the velocity signal is in mm/s.

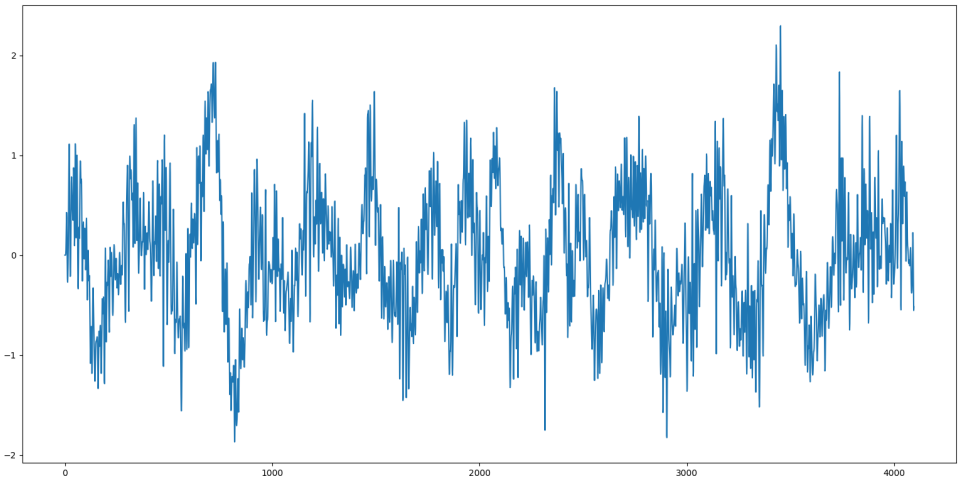

Figure 4.17: Waveform of Velocity Sample

From the acceleration and velocity waveforms, some global levels are also calculated. The first one, of root mean square (rms), indicates the signal energy. The rms value is calculated by the equation 4.11.

$$rms = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_t^2} \tag{4.11}$$

From equation 4.11, x_t is the input signal (acceleration or velocity), and N corresponds to the input signal length.

Another global level calculated is the peak-to-peak ($pkpk$) value, that indicates the maximum amplitude of impacts in the signal. The $pkpk$ value is calculated using the equation 4.12.

$$pkpk = max(x_t) - min(x_t) \tag{4.12}$$

Historical Data Processing

In predictive maintenance, the main objective is to predict faults by trending the vibration and identifying its behavior over time. So, in practical vibration analysis applications, two main aspects are considered in order to correctly identify possible faults in rotating machinery: the frequency analysis of the actual condition; the evolution of the vibration condition over the time.

For the second approach, the global levels calculated (rms and $pkpk$) are compared to historical values of the same component, as algorithm 5 shows.

Algorithm 5: Historical Dataset Generation

```
Input: month                                              ▷ date of measurement
Output: dataset[]                                         ▷ vibration samples
dataset[] ← nil
Function GetData(month):
    measures[] ← getMeasures(month)
    foreach measure ∈ measures[] do
        acl ← measure
        vel ← integration(measure)
        rms_acl, pkpk_acl ← global(acl)
        rms_vel, pkpk_vel ← global(vel)
        ref_rms_acl ← ∞
        ref_pkpk_acl ← ∞
        ref_rms_vel ← ∞
        ref_pkpk_vel ← ∞
        for i ← 1 to 6 do
            tmpa ← getHistMeasure(month − i)          ▷ from the same component
            tmpv ← integration(tmpa)
            tmp_rms_acl, tmp_pkpk_acl ← global(acl)
            tmp_rms_vel, tmp_pkpk_vel ← global(vel)
            if i = 1 then
                prv_rms_acl ← tmp_rms_acl
                prv_pkpk_acl ← tmp_pkpk_acl
                prv_rms_vel ← tmp_rms_vel
                prv_pkpk_vel ← tmp_pkpk_vel
            end
            if tmp_rms_acl < ref_rms_acl then
                ref_rms_acl ← tmp_rms_acl
            end
            if tmp_pkpk_acl < ref_pkpk_acl then
                ref_pkpk_acl ← tmp_pkpk_acl
            end
            if tmp_rms_vel < ref_rms_vel then
                ref_rms_vel ← tmp_rms_vel
            end
            if tmp_pkpk_vel < ref_pkpk_vel then
                ref_pkpk_vel ← tmp_pkpk_vel
            end
            dataset[] ←[acl,rms_acl,rms_acl/prv_rms_acl,rms_acl/ref_rms_acl
            pkpk_acl,pkpk_acl/prv_pkpk_acl,pkpk_acl/ref_pkpk_acl
            rms_vel,rms_vel/prv_rms_vel,rms_vel/ref_rms_vel
            pkpk_vel,pkpk_vel/prv_pkpk_vel,pkpk_vel/ref_pkpk_vel]
        end
    end
    return dataset[]
```

Algorithm 5 shows in details the implementation of the function *ReadData*() previously presented inside algorithm 4.

The comparison is made to a reference value, that is the minimum value measured in the last six months of the components operation, and to a more recent historical value, that is the minimum value measured in the last month of the components operation.

The global levels measured are compared and the sample results in the acceleration waveform and and array of 12 positions, from the global levels comparatives, described as:

- Three values of: the *rms*, *rms* divided by the *rms* of the short term historical, and the *rms* divided by the *rms* of the long term historical; all from the acceleration signal;
- Than three values of: *pkpk*, *pkpk* divided by the *pkpk* of the short term historical, and the *pkpk* divided by the *pkpk* of the long term historical;all from the acceleration signal;
- Three values of: the *rms*, *rms* divided by the *rms* of the short term historical, and the *rms* divided by the *rms* of the long term historical; all from the velocity signal;
- Than three values of: *pkpk*, *pkpk* divided by the *pkpk* of the short term historical, and the *pkpk* divided by the *pkpk* of the long term historical;all from the velocity signal;

4.2.2 Monitoring System Architecture

The architecture for the monitoring system proposed is presented in figure 4.18.

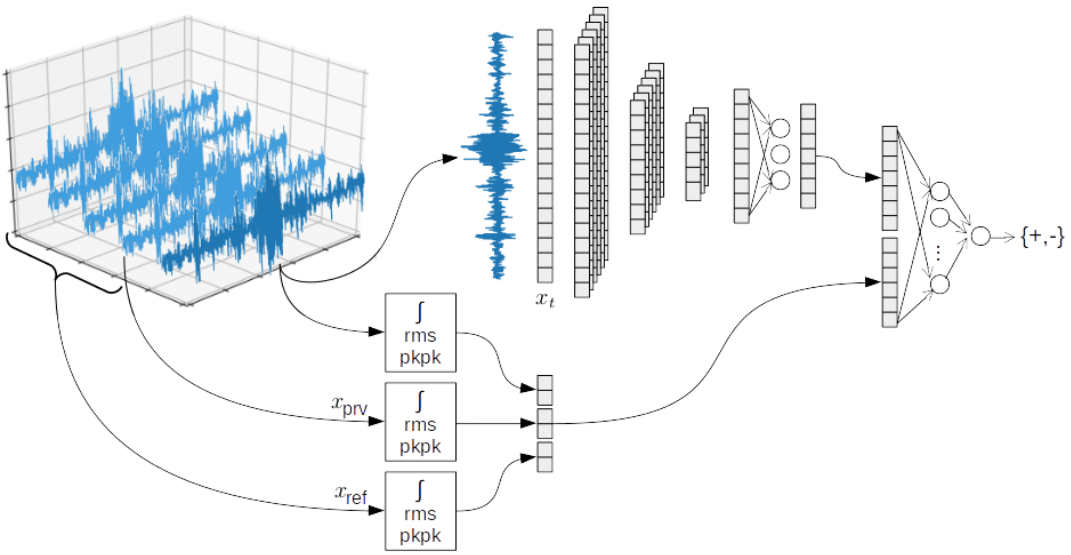

Figure 4.18: WCNN Architecture for Rotating Machine Monitoring

Figure 4.18 shows that the raw vibration signal x_t is used as input of a Wavelet Convolutional Neural Network (WCNN), as the one proposed in section 3.2, used in

this context for feature extraction. The raw vibration signal is passed the WCNN, with convolution, wavelet activation functions, max pooling layers and a densely connected neural network to extract a reduced number of features. The number of layers and its dimensions and characteristics are detailed in the following section 4.2.3.

Note that, the acceleration signal is the WCNN input to extract vibration features, since one of the main challenges, as discussed before, is the fault detection in higher frequencies, with low energy as inner or outer race faults in rolling element bearings, for example.

From figure 4.18, the result of feature extraction from the WCNN proposed is concatenated to the global levels, compared to historical values. The global values used are the *rms* and *pkpk*, both from acceleration and velocity, as described in the last section.

The edge computing based monitoring architecture presented in figure 3.1, proposes that the data acquisition and data manipulation should be embedded in the sensor. In the model proposed in figure 4.18, the WCNN and the global levels are implemented in the sensor firmware. However, the densely connected neural network that performs the fault detection is implemented in the gateway firmware. To be able to embed in these devices firmware, the model dimensions should respect restriction of the hardware capacity, as detailed in the next section.

4.2.3 Model Dimensions

The first path of feature extraction, is composed of three consecutive combinations of wavelet convolutional and max-pooling layers. The first and the last convolutional layer have 2 filters, while the middle one has 4 filters, all of them with 3 values length. All convolutional layers have gaussian wavelet activation functions. The max-pooling layers have downsize factor of 2. These layers are flattened to an array of 1020 samples. This array is passed for one layer of a fully connected network with 8 nodes using *relu* activation function.

The resulting array from the wavelet convolution neural network has eight values, and is concatenated with the array of global levels, detailed in the last section, which has 12 values. The resulting array has 20 values, and that one is transmitted to the gateway for fault detection.

The fault detection neural network, implemented in the gateway has a hidden layer with 16 nodes, a dropout layer and one output node with *sigmoid* activation function. This architecture proposes and extended version of the Wavelet Convolutional Imbalanced Classifier (WCIC) proposed in section 3.4.

Figure 4.19 shows the dimensions mentioned for the architecture proposed.

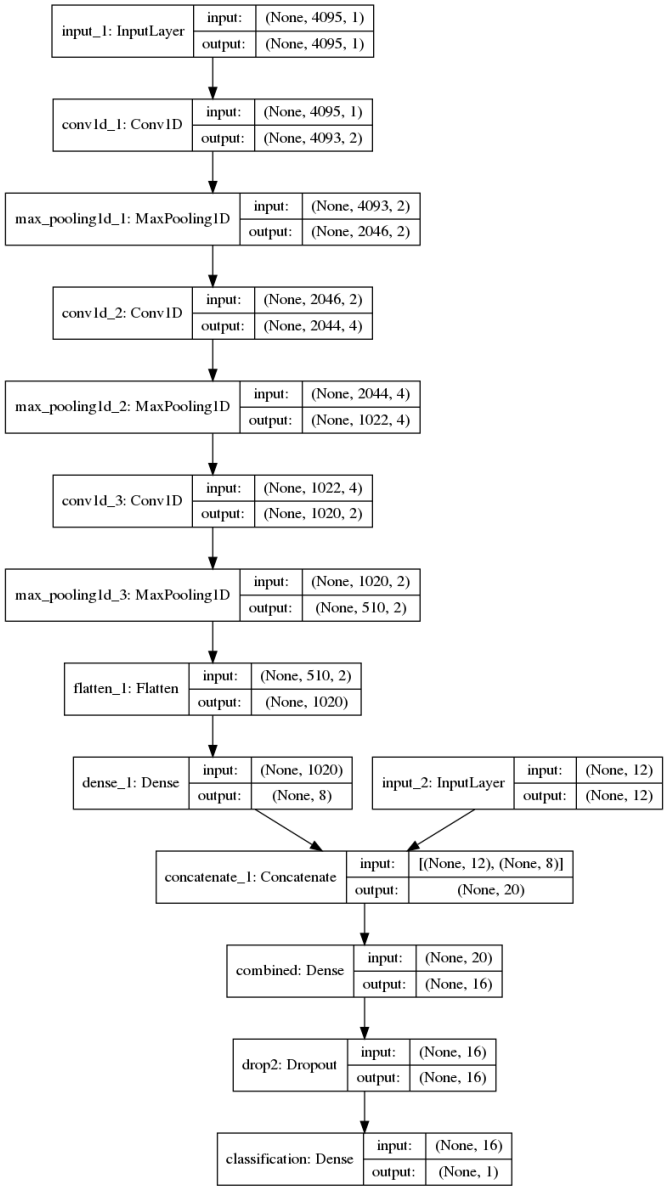

Figure 4.19: WCIC Model Dimensions

In the WCNN, part of the WCIC that should be embedded in the sensor firmware, there are a total of 8230 trainable parameters of convolution filter values and the densely connected network weights. As discussed in section 3.5, each input value and each parameter uses a 16 bit representation, or 2 bytes. Calculating the memory usage in each layer we have:

- $4095 \cdot 2 = 8190$ bytes for the vibration input signal;
- $2 \cdot 2046 \cdot 2 = 8184$ bytes for the first convolution and max pooling layer;
- $4 \cdot 1022 \cdot 2 = 8176$ bytes for the second convolution and max pooling layer;
- $2 \cdot 510 \cdot 2 = 2040$ bytes for the third convolution and max pooling layer;
- $20 \cdot 2 = 40$ bytes for the output features;

Adding to the memory needed for signal representation $8230 \cdot 2 = 16460$ bytes for the model weights manipulation and $1000 \cdot 2 = 2000$ bytes for the wavelet static array storage, the total memory usage for the encoder phase is of 45050 bytes. That is feasible value, even though the vibration sensor proposed in section 3.5 have only 32KB of RAM, it has an external RAM memory of others 32KB, which are sufficient to allocate the 40KB needed for the architecture proposed.

4.2.4 Results for the WCIC

The results obtained by the fault detection routine described in the architecture presented in figure 4.18 are calculated based on the following assumption: a normal component, without any failure is defined as a negative result, while the presence of a fault is considered a positive result. In that way, two metrics are used to calculate the model efficiency:

- **Accuracy**: proportion of corrected predictions divided by the total;

$$acc = \frac{t_p + t_n}{t_p + f_p + t_n + f_n} \tag{4.13}$$

- **Recall**: proportion of correct fault detections divided by the sum of fault detections and false negatives;

$$rec = \frac{t_p}{t_p + f_n} \tag{4.14}$$

In the equations above t_p and t_n denotes true positives and true negatives, that are the number of correct predictions. While f_p and f_n are false positives and false negatives, that are the number of incorrect predictions. As discussed in section 2.5.2, false positives are normal condition components diagnosed as to be faulty, and false negatives are the opposite: defective components diagnosed as to be normal.

False negatives are much worse than false positives in the context of rotating machinery condition monitoring, as discussed before. So, the model goal is to maximize both metrics: higher accuracy limited to a high value of recall. In this work, it is assumed that the minimum tolerance for false negatives is of 1%, that means that the minimum value accepted by the recall metric is 0.99.

The training strategy for the model proposed consists of retrieving the samples from the historical of six months. Than a split of this dataset is executed, of 80% of samples used for training and 20% for testing. That is done in a 20-fold cross validation routine. The metrics of accuracy and recall are calculated, and only the models that reaches recall values higher than 0.99 have the accuracy considered.

Algorithm 6 shows the above strategy in details.

Algorithm 6: Training Model

Input: $Samples[], epochs$ ▷ six months historical data and epochs of training
Output: $model$ ▷ trained model

Function `FitModel`$(X_{train}, X_{test}, y_{train}, y_{test}, epochs)$**:**
    $model \leftarrow nil$
    $bestAcc \leftarrow 0$
    for $e \leftarrow 1$ **to** $epochs$ **do**
        model $\leftarrow Fit(X_{train}, X_{test}, y_{train}, y_{test})$
        $acc, rec \leftarrow CalcMetrics(model, X_{test}, y_{test})$ ▷ from equations in 4.13 and 4.14
        if $rec > 0.99$ **then**
            if $acc > bestAcc$ **then** $bestModel \leftarrow model$
    end
    return $bestModel$

Function `TrainModel`$(Samples[], epochs)$**:**
    $bestModel \leftarrow nil$
    $bestAcc \leftarrow 0$
    for $seed \leftarrow 1$ **to** 20 **do**
        $X_{train}, X_{test}, y_{train}, y_{test} \leftarrow SplitSamples(Samples[], 0.2, seed)$
        $model \leftarrow FitModel(X_{train}, X_{test}, y_{train}, y_{test}, epochs)$
        $acc, rec \leftarrow CalcMetrics(model, X_{test}, y_{test})$ ▷ from equations in 4.13 and 4.14
        if $acc > bestAcc$ **then** $bestModel \leftarrow model$
    end
    return $bestModel$

In algorithm 6, the function Fit uses the Fourth Root Loss function, presented in equation 3.10 from section 3.4. This function is used to calculate the loss in each epoch and adjust the model weights.

From the function of $TrainModel$, one can see that the process of fitting the model is repeated through different splits of the historical samples. The function $SplitSamples$ splits the samples randomly and only the best model result is considered for validation assessment, described as follows.

As described in section 4.2.1, the experimental data retrieve consists of measurements of Aug-18 to March-19. Because of that, we can validate the WCIC results for two reference months: Feb-19 - using as the training dataset measurements from Aug-18 to Jan-19; and Mar-19 - using as the training dataset measurements from Sep-18 to Feb-19.

Results for Feb-19

As mentioned before, to calculate the results for Feb-19, the model is trained using data from Aug-18 to Jan-19. Figure 4.20 shows the training history for the best model trained, out of the 20-fold cross validation routine explained in algorithm 6. The best model has the higher result of accuracy, limited to a recall value of at least 0.99 for the testing dataset.

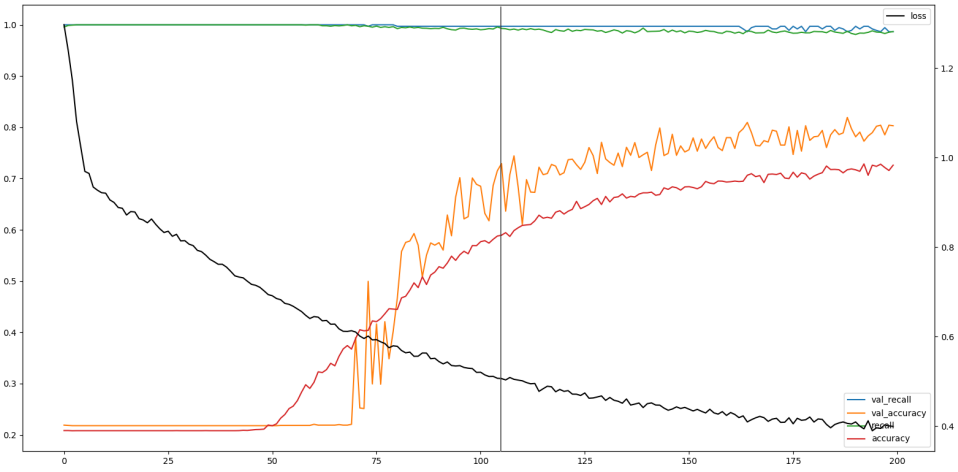

Figure 4.20: Training history of Aug-18 to Jan-19.

In figure 4.20 we can see the evolution over the epochs of training of: the loss value in black; the accuracy for the training and testing datasets (80% and 20% splits) in red and orange, respectively; and the recall values for the training and testing datasets in green and blue, respectively. The vertical line is the epoch where the the conditions of the best model trained were reached: higher accuracy in the testing dataset, with recall value of at least 0.99.

Table 4.10 shows the validation results. As mentioned before, to estimate the classification values we use data not considered in the training and testing algorithm proposed. All the 36829 measurements made on Feb-19 are classified and the results are presented in the following table.

Table 4.10: Results for February 2019 using training history of Aug-18 to Jan-19.

	Accuracy	**Recall**	**False Negatives**
Test (20% split)	0,731	0,992	3
February 19	0,660	0,986	4

Together with Feb-19 results, for comparison purpose, we present also the result obtained in the test dataset (20% split of measurements made from Aug-18 to Jan-19).

Results for Mar-19

Similar to the results presented in the above section, we calculated the results for Mar-19. The model was trained using data from Sep-18 to Feb-19, and figure 4.20 shows the training history for the best model trained, out of the 20-fold cross validation routine. The criteria for the best model is the same, as the higher result of accuracy, limited to a recall value of at least 0.99 for the testing dataset.

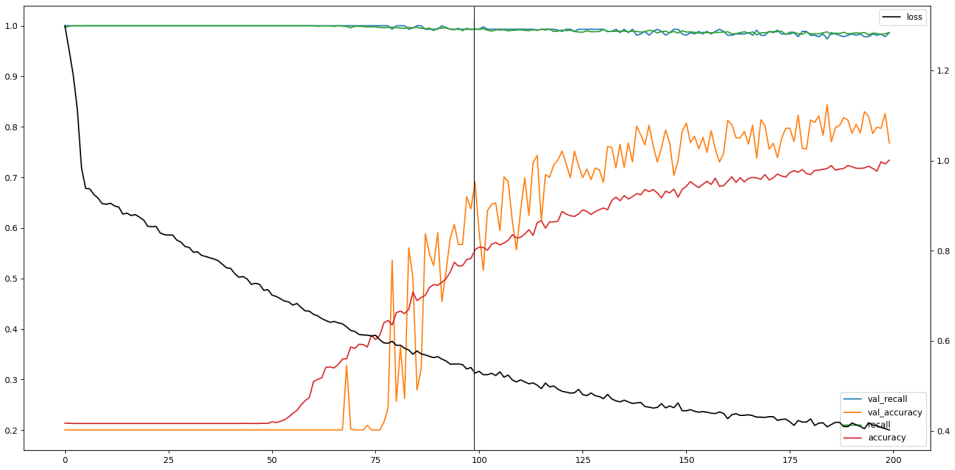

Figure 4.21: Training history of Sep-18 to Feb-19.

The same color representation is used in figures 4.20 and 4.21. And the vertical line is the epoch where the the conditions of the best model trained were reached.

The results for Mar-19 can be seen in table 4.11, where all 36529 measurements are classified. A comparison is made with the results calculated in the testing dataset (20% split of measurements made from Sep-18 to Feb-19).

Table 4.11: Results for March 2019 using training history of Sep-18 to Fev-19.

	Accuracy	**Recall**	**False Negatives**
Test (20% split)	0,725	0,990	4
March 19	0,665	0,986	4

The results before, for Feb-19 and Mar-19, were calculated with the WCIC proposed in the architecture presented in figure 4.18. This WCIC is based on the generic WCNN proposed in section 3.2, with convolution filters, wavelet activation function and the fourth root loss function.

In the next section, a comparison is made of the proposed model to more conventional alternatives, such as the $ReLU$ instead of the wavelet activation function, and the Log and Logcosh instead of the fourth root loss function.

4.2.5 Comparing results to conventional CNN

This section makes a comparison of the proposed model and conventional cost sensitive convolutional neural networks. For comparison purposes, we used the same architecture proposed in 4.2.2, the same model and dimensions presented in 4.2.3, and the same strategy of training proposed in Algorithm 6 for all networks.

The results were calculated with both Wavelet and $ReLU$ activation functions for the convolution layers in the architecture proposed. For each one, the loss function used to train the models were the conventional *log* loss, the *log cosh* loss and the proposed *fourth root* loss, as discussed in 3.4.

The following results are from all different splits used in algorithm 6, of the 20-fold cross validation described. For every random split, the six combinations of activation and

loss functions were applied, so the tests are paired. The results for training dataset of Aug-18 to Jan-19 applied to Feb-19 validation are presented in table 4.12.

Table 4.12: Comparison results for February 2019.

ReLU Activation Function									**Wavelet Activation Function**								
Log Loss			**Log Cosh**			**Fourth Root**			**Log Loss**			**Log Cosh**			**Fourth Root**		
acc	rec	fn	acc	rec	fn	acc	rec	fn	acc	rec	fn	acc	rec	fn	acc	rec	fn
0.62	0.99	4	0.61	0.99	3	0.68	0.98	5	0.64	0.98	5	0.58	0.99	2	0.66	0.98	6
0.74	0.97	10	0.76	0.97	9	0.52	0.99	2	0.74	0.96	11	0.72	0.97	8	0.66	0.99	4
0.76	0.97	8	0.82	0.95	14	0.70	0.97	9	0.79	0.95	14	0.82	0.94	18	0.68	0.98	5
0.66	0.98	5	0.65	0.99	4	0.63	0.99	4	0.67	0.98	5	0.68	0.98	5	0.70	0.98	6
0.59	0.99	3	0.54	0.99	2	0.64	0.98	5	0.58	0.99	2	0.55	0.99	2	0.67	0.98	7
0.71	0.98	6	0.64	0.99	2	0.64	0.99	4	0.49	1.00	1	0.69	0.98	5	0.66	0.99	4
0.70	0.98	6	0.71	0.98	6	0.62	0.99	4	0.71	0.98	6	0.70	0.98	6	0.64	0.99	4
0.64	0.99	3	0.72	0.98	5	0.69	0.97	8	0.70	0.99	4	0.63	0.99	4	0.71	0.98	6
0.74	0.97	8	0.73	0.97	9	0.69	0.98	6	0.65	0.98	5	0.73	0.98	7	0.71	0.97	8
0.55	0.99	2	0.76	0.97	10	0.55	0.99	2	0.79	0.95	13	0.69	0.98	7	0.69	0.98	7
0.68	0.98	6	0.68	0.98	6	0.68	0.98	6	0.69	0.98	6	0.68	0.98	5	0.67	0.98	6
0.61	0.97	8	0.73	0.97	8	0.68	0.98	7	0.73	0.97	8	0.70	0.98	5	0.40	0.99	2
0.64	0.99	4	0.71	0.98	5	0.63	0.98	5	0.77	0.97	9	0.68	0.98	7	0.75	0.97	9
0.71	0.98	6	0.71	0.98	6	0.71	0.98	6	0.72	0.98	6	0.72	0.98	6	0.72	0.98	6
0.67	0.98	5	0.67	0.98	5	0.67	0.98	5	0.68	0.98	5	0.68	0.98	5	0.68	0.98	5
0.78	0.95	15	0.66	0.98	6	0.66	0.98	6	0.66	0.98	6	0.63	0.99	4	0.63	0.99	4
0.72	0.98	7	0.68	0.98	5	0.61	0.98	5	0.67	0.98	5	0.64	0.98	5	0.67	0.98	5
0.65	0.98	5	0.65	0.98	5	0.65	0.98	5	0.69	0.98	5	0.68	0.98	6	0.69	0.98	6
0.75	0.96	11	0.64	0.98	6	0.64	0.98	6	0.64	0.98	6	0.69	0.98	6	0.61	0.99	4
0.77	0.97	10	0.79	0.97	10	0.77	0.97	10	0.70	0.98	5	0.79	0.97	10	0.70	0.98	5

The results for training dataset of Sep-18 to Feb-19 applied to Mar-19 validation are presented in table 4.13.

Table 4.13: Comparison results in percentage for March 2019.

ReLU Activation Function									**Wavelet Activation Function**								
Log Loss			**Log Cosh**			**Fourth Root**			**Log Loss**			**Log Cosh**			**Fourth Root**		
acc	rec	fn	acc	rec	fn	acc	rec	fn	acc	rec	fn	acc	rec	fn	acc	rec	fn
0.62	0.99	4	0.60	0.99	4	0.63	0.99	4	0.66	0.98	6	0.57	0.99	4	0.67	0.98	5
0.72	0.97	10	0.70	0.97	10	0.69	0.98	6	0.69	0.98	6	0.69	0.97	10	0.68	0.98	6
0.79	0.96	13	0.80	0.96	13	0.68	0.98	6	0.79	0.96	13	0.79	0.96	13	0.72	0.98	7
0.81	0.94	19	0.75	0.97	10	0.72	0.98	7	0.75	0.97	10	0.80	0.96	13	0.71	0.98	7
0.67	0.97	10	0.67	0.97	10	0.70	0.98	6	0.66	0.98	6	0.67	0.97	10	0.70	0.98	7
0.62	0.99	4	0.59	0.99	4	0.71	0.97	9	0.59	0.99	4	0.61	0.99	4	0.72	0.97	9
0.67	0.98	6	0.59	0.99	4	0.48	0.99	3	0.55	0.99	4	0.58	0.99	4	0.68	0.98	6
0.71	0.97	10	0.69	0.97	10	0.64	0.99	4	0.72	0.97	10	0.72	0.96	11	0.68	0.98	6
0.72	0.96	11	0.68	0.97	10	0.71	0.98	6	0.70	0.97	10	0.70	0.97	10	0.70	0.97	10
0.72	0.97	9	0.72	0.97	10	0.70	0.98	6	0.69	0.97	9	0.72	0.97	10	0.70	0.98	6
0.67	0.97	10	0.67	0.97	10	0.67	0.97	10	0.70	0.97	10	0.60	0.99	4	0.60	0.99	4
0.70	0.97	10	0.66	0.97	10	0.70	0.97	10	0.72	0.97	10	0.72	0.97	10	0.72	0.97	10
0.75	0.97	10	0.54	0.99	4	0.67	0.98	5	0.56	0.99	4	0.57	0.99	4	0.63	0.99	4
0.63	0.99	4	0.63	0.99	4	0.63	0.99	4	0.63	0.99	4	0.54	0.99	4	0.54	0.99	4
0.75	0.97	10	0.75	0.97	10	0.75	0.97	10	0.64	0.99	4	0.64	0.99	4	0.64	0.99	4
0.73	0.97	10	0.74	0.96	12	0.68	0.98	5	0.69	0.98	5	0.73	0.96	11	0.68	0.98	6
0.79	0.94	19	0.73	0.97	8	0.73	0.97	8	0.79	0.96	12	0.65	0.99	4	0.65	0.99	4
0.74	0.96	12	0.73	0.96	11	0.63	0.99	4	0.73	0.97	10	0.72	0.97	10	0.67	0.98	6
0.64	0.99	4	0.64	0.99	4	0.64	0.99	4	0.62	0.99	4	0.58	0.99	4	0.52	0.99	3
0.66	0.97	8	0.67	0.97	10	0.66	0.97	8	0.71	0.97	10	0.69	0.97	9	0.52	0.99	4

Table 4.14 shows the mean and standard deviation for the results using *ReLU* as activation function, while 4.15 shows the results using Wavelet as activation function. They both aggregate the results from February and March of 2019.

Table 4.14: Comparison results for *ReLU*.

ReLU Activation Function									
	Log Loss			**Log Cosh**			**Fourth Root**		
	acc	rec	fn	acc	rec	fn	acc	rec	fn
average	0.695	0.972	8.125	0.686	0.975	7.350	0.662	0.980	5.875
stddev	0.061	0.013	3.988	0.066	0.011	3.175	0.056	0.007	2.127

Table 4.15: Comparison results for Wavelet.

Wavelet Activation Function									
	Log Loss			**Log Cosh**			**Fourth Root**		
	acc	rec	fn	acc	rec	fn	acc	rec	fn
average	0.683	0.976	6.950	0.675	0.976	6.900	0.661	0.981	5.675
stddev	0.067	0.011	3.194	0.068	0.012	3.470	0.067	0.006	1.817

If we consider the most conventional model (*ReLU* with *Log* loss), the accuracy scores are in average a little higher: 69.5% versus 66.1% of the proposed model (*Wavelet* with *Fourth Root* loss), and the standard deviation from accuracy results are: 6.1% versus 6.7%, respectively.

The recall results, however, present the opposite behavior. The recall results are a little higher for the proposed method, with 0.981 versus 0.972 of the conventional model. The standard deviation from recall results were: 0.006 versus 0.013, respectively.

In a closer look, the comparison of the results in tables 4.14 and 4.15 shows that the Wavelet Activation function presents better results in average for the recall metric, and consequently, for the total number of false negatives. However, accuracy tend to be slightly lower for all loss function tested to train the wavelet convolutional neural network.

Even though accuracy and recall results seems to differ, we should consider the Wilcoxon Signed-rank statistical test to analyze if this difference has significance. It is considered that the tests were paired because of the 20-fold cross validation results presented in tables 4.12 and 4.13.

The definition for the null hypothesis is that conventional convolutional neural network with *ReLU* activation function trained with the *Log* loss function and the proposed wavelet convolutional neural network trained with the *Fourth Root* loss have the same distribution for the results. That means that, if the hypothesis is confirmed, there is no difference of using one model instead of the other.

As parameters for the test we use a 0.01 level for significance. Calculating the Wilcoxon Signed-rank for the **accuracy** results we have the value of z equals -2.527 and the $p - value$ is 0.0114. The result is not significant at $p < 0.01$. That means that accuracies are likely to be the same for both models.

The same way, calculating the Wilcoxon Signed-rank for the **recall** results we have the value of z equals -2.9516 and the $p - value$ is 0.00318. We can conclude that the result is significant at $p < 0.01$. So the recall results are different depending on the model used. As stated in table 4.15, and statistically showed by the Wilcoxon test, the WCIC proposed performs better in terms of a lower number of false negatives.

For the application described, when false negatives are catastrophically worse than false positives, the proposed model should be considered a better solution.

4.2.6 Comparing results to the literature

The results presented in the sections before do not show very high accuracy levels, as the results in the literature. But many assumptions made were not considered in most of the related work.

For the model proposed in this thesis, the application described of real industry health monitoring system must deal with very imbalanced datasets and with a false negative restriction near to zero, in order to mitigate catastrophic failures. In the literature, these very practical restrictions are commonly relaxed.

For example, the work presented in [44] proposes an autoencoder for deep feature learning, with a non-standard loss function, the correntropy. The experiments were executed in a gearbox fault test rig and in a locomotive roller bearing test rig. But as they have control of the experiments, the datasets were generated in a balanced way.

Besides that, even though the results presented have better accuracy in comparison to the standard autoencoder and SVM algorithms, no differentiation is made about false negatives. For the gearbox experiment presented in the article, accuracy was around 94.05% in average and 1.34% of standard deviation. However, for one of the possible gearbox failure, the spalling fault, there was 2.5% of false negatives.

The article presented in [37] detects faults in reciprocating compressors which have nonlinear and non-stationary vibration signals. It also proposes an autoencoder, but to overcome the noise influence in vibration signal. They applied the proposed method to a natural gas reciprocating compressor in China. Although the application is on a real industry component, the dataset was generated in a balanced way.

The results had also higher accuracy than some peer algorithms, specially in the presence of low SNR. But, again, no differentiation is made on false negatives. With no noise, all methods achieved 100% of accuracy. With $5db$ of noise, the proposed method achieved 99.3% of accuracy, but 0.5% of false negatives.

Another method, applied in a gearbox benchmark for mechanical fault diagnosis, was presented in [50]. It proposed an unidimensional CNN for deep feature extraction and fault classification. The method proposed achieved accuracy of about 99.3% and recall of also 99.3%.

In [12], the authors present a compact architecture of unidimensional CNN. It was tested for bearing fault diagnosis from two commonly used real benchmark vibration datasets. It achieved accuracies higher than 93% on both datasets. Other metrics beyond accuracy were calculated. The recall metric was around 94% and 93% for datasets used.

These results are all in controlled experiments, or from some known benchmarks, and suffer from some major simplifications. Besides the imbalance of class samples, and the lack of recall metrics the main aspect that should be analyzed before purely comparing the results is about the fault severity.

For example, let us analyze one benchmark of real rolling element bearing vibration measurements, widely used, as in [12]. The experiments made to create the benchmark data were test-to-failure ones. In those kind of experiments, the data are collected until the bearings worked for over the designed lifetime, until the occurrence of a failure such as: inner race, outer race, cage or rolling element fail. In this case, time-to-failure tests

were about 2 days of straight measurements.

Because of that, the severity of failure considered as a fault in most of the literature related works are much higher than the one we assume in this thesis. For example, when classifying the rolling element condition training with the benchmark created with test-to-failure experiments, the maximum prediction of failure possible is of 2 days.

That is the big difference from the dataset used in this thesis: it is retrieved from real machine health monitoring systems. As described in section 2.1, the advantage of predictive maintenance is to reduce costs with unnecessary maintenance stops. To make this strategy possible, the classification model should make an early fault detection, with time to correctly program the maintenance. Detecting an imminent failure, as most related works do, can avoid breakdowns, but does not reduce maintenance cost or minimize production line stops.

Because of that, the dataset retrieved from SEMEQ historical data have much more incipient failures than the ones that are simulated in test rigs, or from conventional benchmarks. And that makes the classification problem much more challenging, and the accuracy results smaller, but completely more useful in practical applications.

4.2.7 Results Discussion

The results presented in section 4.2.4 show that the binary classification of real industry rotating machinery can be done with a low false negative rate. Also, the model dimensions explained in section 4.2.3 show that the wavelet convolutional neural network part of the WCIC proposed has limited size parameters, feasible to be embedded in vibration sensor firmware, with limited hardware available.

Once the WCIC model is trained, the monitoring system can be implemented as the following figure 4.22 illustrates.

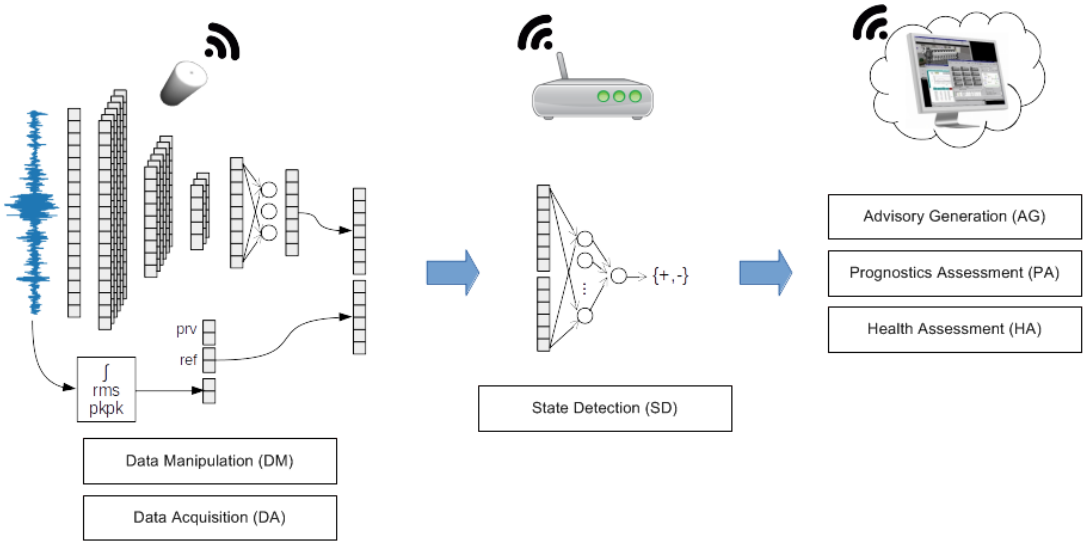

Figure 4.22: Rotating machinery monitoring system.

The architecture proposed in section 4.2.2, that defines the monitoring system architecture for rotating machinery, can be implemented following the characteristics presented

in section 3.1, that presents the edge computing based monitoring system proposed in this thesis, based on the Open System Architecture for Condition Based Monitoring (OSA-CBM), illustrated in figure 2.1.

Note that for the implementation of the proposed method embedded in the vibration sensor, the previous and the reference measurements global levels were calculated and stored. Only the actual vibration measurement is acquired, the global levels calculated and compared to the previous values as algorithm 5 shows.

The binary classification neural network can be implemented in the gateway hardware for state detection. In the case of a normal classification, the features are sent to the cloud server for logging purpose. For normal equipments, instead of transmitting the 4095 samples, only 20 features are transmitted. For defective machines, besides the 20 features, the full signal must also be transmitted in order to be analyzed in the cloud Health and Prognostics Assessment modules.

The network load can be reduced by the WCIC proposed. The average accuracy is around 66%, the other 34% is composed mainly of false positives, since recall metric was restricted to 0.99. As false positives are normal equipments classified as defective, they transmit the features and the full signal, in a total of 5015 samples.

The failure rate of rotating machines are about 1%, as stated in section 4.2.1. So, from the 66% of corrected predictions, 1% are of true negatives, which also transmit 5015 samples. So, the transmission data can be calculated as: $(0.65 * 20 + (0.01 + 0.34) * 5015)/5015 = 0.3526$. That means a reduction of 64.7% on the network load.

Once continuous monitoring is possible, any malfunctions that would lead to an unexpected failure and consequently to breakdown of the equipment can be avoided. A break would impact in large maintenance costs and a possible power outage to the community that depends on the power generation. As discussed in section 2.1 based on data presented in [11] intelligent maintenance systems are able to eliminate breakdowns up to 70%, in average. Besides avoiding breakdowns, the WCIC can save up to 12% over scheduled repairs and reduce overall maintenance costs up to 30%.

Chapter 5

Conclusion

The development of the Wavelet Convolutional Neural Network (WCNN) as proposed in this thesis, makes it possible to implement edge based smart machine health monitoring systems. Two very distinct applications were presented, one for large internal combustion in-cylinder pressure reconstruction and parameter estimation and one for rotating machinery fault detection. In both applications the results show the feasibility of the architecture and the models proposed.

We worked on some major open issues, and the thesis succeeded in proposing an architecture that makes use of high capacity sensors, with high frequency and data sampling, sufficient for sophisticate vibration analysis, such as rolling element bearing fault detection. And the models proposed could reduce the amount of data transmitted, without loosing sensitive information to the applications proposed.

By reducing the amount of data transmitted by the sensors, the edge computing benefits are achieved, such as lower latency, faster response time and prioritization. Since the proposed models have compact architectures, it is possible to implement them in the limited hardware available on the edge devices. Because of that, fault detection does not need to be performed exclusively on top level applications, on cloud servers, as the rotating machinery monitoring system proposed in thesis demonstrated.

Machine health monitoring systems prevent unexpected breakdowns, but for internal combustion engines, specially for large power generation plants, besides avoiding power generation black-outs, the fuel consumption can be optimized and pollutant emissions minimized, which impacts not only in economic gains but also in environmental benefits.

The condition based monitoring with the Wavelet Convolutional Autoencoder (WCAE) proposed reached very good results for in-cylinder pressure reconstruction. We demonstrate that the autoencoder compression brings the edge computing benefits, without loosing information for the reconstruction. Besides that, some combustion parameter estimations, such as maximum pressure peak and is angle in relation to the cylinders TDC, are very important values to identify if the engine is regulated, for optimum fuel consumption. The work presented show very promising results, even more accurate if comparing the results with the literature.

For rotating machinery health monitoring, the major challenges still open in the literature motivated the development of the models proposed in this thesis. Severe class imbalance and very low tolerance for false negatives were hard restrictions for the models

proposed. The accuracy result reached were not as high as the literature, but in many cases, these restrictions are relaxed in the related works. But, the accuracy of 66% in average, with very low false negative rate, is sufficient for a 64.7% reduction of data transmitted in the sensor network. Besides that, with the detection implemented in edge devices, the worst condition equipment can be flagged and the analysts can deal with them first, reducing the response time for the system.

Besides that, the database gathered and used for the rotating machinery model training in this thesis is much more challenging than the most vibration benchmarks used in the literature. It is composed of measurements on many different kinds of components, such as motors, compressors, pumps, bearings, gearboxes, turbines, etc. All these components are part of real industries around the world, and from many different industry segments, as chemical, foundry, food, beverage, stamping, power generation, etc. And also, a complete list of failures is considered of unbalance, misalignment, wear, rolling element bearing failures such as inner race, outer race, lubricant and electric faults, among others.

The architecture and the models developed together with the wavelet activation and loss function proposed in this thesis must inspire the adoption of edge computing on machine health monitoring systems. The benefits over cloud computing are clear with we assume that condition based monitoring is time sensitive, meaning that the response time to a fault detection should be minimized. Not only time is important, the vibration signal quality is also fundamental. And the deep learning models proposed, that extract relevant features from raw data measurements is the path for building practical vibration monitoring systems, in this Industrial Internet of Things (IIoT) context.

As future works, a state detection for the internal combustion engine monitoring should be implemented, similar to the one proposed for rotating machinery. A binary classification from the vibration features extracted for pressure reconstruction could reduce even more the data transmitted and permit the prioritization to power generation plants fault analysis.

Another future work for implementing the WCAE on embedded devices is the synchronization necessary for data acquisition between vibration and the rotation signals. Synchronization is fundamental for parameter estimation and fuel consumption optimization.

The proposed fourth root loss function was compared to more traditional losses in the context of rotating machinery health monitoring. The results show that this function performed better for false negatives reduction, but to conclude that, a more complete analysis should be executed with different contexts.

Also as future work, a multi-class classification for rotating machinery fault detection can be developed. The identification of the failure type improves the prioritization, and also makes it possible to develop a severity classification together with remaining useful life for the component. All these future works improve the efficiency and benefits for the adoption of edge computing in smart monitoring systems.

Bibliography

[1] D. Abboud, M. Elbadaoui, W.A. Smith, and R.B. Randall. Advanced bearing diagnostics: A comparative study of two powerful approaches. *Mechanical Systems and Signal Processing*, 114:604 – 627, 2019.

[2] Antonios K Alexandridis and Achilleas D Zapranis. Wavelet neural networks: A practical guide. *Neural Networks*, 42:1–27, 2013.

[3] Amaury B. Andre, Eduardo Beltrame, and Jacques Wainer. A combination of support vector machine and k-nearest neighbors for machine fault detection. *Applied Artificial Intelligence*, 27(1):36–49, 2013.

[4] Yuri Sousa Aurelio, Gustavo Matheus de Almeida, Cristiano Leite de Castro, and Antonio Padua Braga. Learning from imbalanced data sets with weighted cross-entropy function. *Neural Processing Letters*, Jan 2019.

[5] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249 – 259, 2018.

[6] Xin-Cheng Cao, Bin-Qiang Chen, Bin Yao, and Wang-Peng He. Combining translation-invariant wavelet frames and convolutional neural network for intelligent tool wear state identification. *Computers in Industry*, 106:71–84, 2019.

[7] Renxiang Chen, Xin Huang, Lixia Yang, Xiangyang Xu, Xia Zhang, and Yong Zhang. Intelligent fault diagnosis method of planetary gearboxes based on convolution neural network and discrete wavelet transform. *Computers in Industry*, 106:48–59, 2019.

[8] ZhiQiang Chen, Chuan Li, and René-Vinicio Sanchez. Gearbox fault identification and classification with convolutional neural networks. *Shock and Vibration*, 2015:1–10, 10 2015.

[9] Giancarlo Chiatti, Ornella Chiavola, Erasmo Recco, Agnese Magno, Ezio Mancaruso, and Bianca M. Vaglieco. Accelerometer measurement for mfb evaluation in multi-cylinder diesel engine. *Energy*, 133(Supplement C):843 – 850, 2017.

[10] Jaesung Chung, Seungsuk Oh, Kyunghan Min, and Myoungho Sunwoo. Real-time combustion parameter estimation algorithm for light-duty diesel engines using in-cylinder pressure measurement. *Applied Thermal Engineering*, 60:33–43, 10 2013.

[11] Paul Daugherty, Prith Banerjee, Walid Negm, and Allan E Alter. Driving unconventional growth through the industrial internet of things. *accenture technology*, 2015.

[12] Levent Eren, Turker Ince, and Serkan Kiranyaz. A generic intelligent bearing fault diagnosis system using compact adaptive 1d cnn classifier. *Journal of Signal Processing Systems*, 91(2):179–189, Feb 2019.

[13] MIMOSA Open Standards for Physical Asset Management. Open system architecture for condition-based maintenance, 2019.

[14] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks for texture classification. *arXiv preprint arXiv:1707.07394*, 2017.

[15] Meng Gan, Cong Wang, and Changan Zhu. Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings. *Mechanical Systems and Signal Processing*, 72-73:92 – 104, 2016.

[16] Qiang Guo, Xuefei Cao, and Qinglong Zou. Enhanced wavelet convolutional neural networks for visual tracking. *Journal of Electronic Imaging*, 27(5):053046, 2018.

[17] Qianjian Guo, Xiaoni Qi, Zheng Wei, Qiang Yin, Peng Sun, Pengjiang Guo, and Jingcheng Liu. Modeling and characteristic analysis of fouling in a wet cooling tower based on wavelet neural networks. *Applied Thermal Engineering*, 152:907–916, 2019.

[18] Xiaojie Guo, Liang Chen, and Changqing Shen. Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis. *Measurement*, 93:490 – 502, 2016.

[19] Shao Haidong, Jiang Hongkai, Zhao Ke, Wei Dongdong, and Li Xingqiu. A novel tracking deep wavelet auto-encoder method for intelligent fault diagnosis of electric locomotive bearings. *Mechanical Systems and Signal Processing*, 110:193 – 209, 2018.

[20] Runzhe Han, Christian Bohn, and Georg Bauer. Recursive engine in-cylinder pressure estimation using kalman filter and structural vibration signal. *IFAC-PapersOnLine*, 51(31):700 – 705, 2018. 5th IFAC Conference on Engine and Powertrain Control, Simulation and Modeling E-COSM 2018.

[21] Duy-Tang Hoang and Hee-Jun Kang. A survey on deep learning based bearing fault diagnosis. *Neurocomputing*, 335:327 – 335, 2019.

[22] Wenhui Hou, Ye Wei, Yi Jin, and Changan Zhu. Deep features based on a dcnn model for classifying imbalanced weld flaw types. *Measurement*, 131:482 – 489, 2019.

[23] Wenyi Huang, Junsheng Cheng, and Yu Yang. Rolling bearing fault diagnosis and performance degradation assessment under variable operation conditions based on nuisance attribute projection. *Mechanical Systems and Signal Processing*, 114:165 – 188, 2019.

[24] Turker Ince, Serkan Kiranyaz, Levent Eren, Murat Askar, and Moncef Gabbouj. Real-time motor fault detection by 1d convolutional neural networks. *IEEE Transactions on Industrial Electronics*, 63(11):7067–7075, 2016.

[25] MM Manjurul Islam and Jong-Myon Kim. Automated bearing fault diagnosis scheme using 2d representation of wavelet packet transform and deep convolutional neural network. *Computers in Industry*, 106:142–153, 2019.

[26] Ali Jafari, Ashwinkumar Ganesan, Chetan Sai Kumar Thalisetty, Varun Sivasubramanian, Tim Oates, and Tinoosh Mohsenin. Sensornet: A scalable and low-power deep convolutional neural network for multimodal data classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, (99):1–14, 2018.

[27] Olivier Janssens, Viktor Slavkovikj, Bram Vervisch, Kurt Stockman, Mia Loccufier, Steven Verstockt, Rik Van de Walle, and Sofie Van Hoecke. Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, 377(Supplement C):331 – 345, 2016.

[28] Shaobo Ji, Xin Lan, Jing Lian, Hao Wang, Meng Li, Yong Cheng, and Wei Yin. Combustion parameter estimation for ice from surface vibration using frequency spectrum analysis. *Measurement*, 128:485 – 494, 2018.

[29] Feng Jia, Yaguo Lei, Liang Guo, Jing Lin, and Saibo Xing. A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines. *Neurocomputing*, 272:619 – 628, 2018.

[30] Feng Jia, Yaguo Lei, Liang Guo, Jing Lin, and Saibo Xing. A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines. *Neurocomputing*, 272:619–628, 2018.

[31] Libin Jia, Jeffrey Naber, and Jason Blough. Frequency response function adaptation for reconstruction of combustion signature in a 9-l diesel engine. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 229, 01 2015.

[32] Yao Jin, Changzheng Shan, Yan Wu, Yimin Xia, Yuntao Zhang, and Lei Zeng. Fault diagnosis of hydraulic seal wear and internal leakage using wavelets and wavelet neural network. *IEEE Transactions on Instrumentation and Measurement*, (99):1–9, 2018.

[33] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3573–3587, Aug 2018.

[34] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219 – 235, 2019.

[35] Yaguo Lei, Jing Lin, Zhengjia He, and Ming J. Zuo. A review on empirical mode decomposition in fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing*, 35(1):108 – 126, 2013.

[36] Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, 108:33 – 47, 2018.

[37] Yang Liu, Lixiang Duan, Zhuang Yuan, Ning Wang, and Jianping Zhao. An intelligent fault diagnosis method for reciprocating compressors based on lmd and sdae. *Sensors*, 19(5):1041, 2019.

[38] M. Ma, C. Sun, and X. Chen. Deep coupling autoencoder for fault diagnosis with multimodal sensory data. *IEEE Transactions on Industrial Informatics*, 14(3):1137–1145, March 2018.

[39] Daniel Moraes, Jacques Wainer, and Anderson Rocha. Low false positive learning with support vector machines. *Journal of Visual Communication and Image Representation*, 38:340 – 350, 2016.

[40] R.B. RANDALL, J. ANTONI, and S. CHOBSAARD. The relationship between spectral correlation and envelope analysis in the diagnostics of bearing faults and other cyclostationary machine signals. *Mechanical Systems and Signal Processing*, 15(5):945 – 962, 2001.

[41] Robert B. Randall and Jérôme Antoni. Rolling element bearing diagnostics—a tutorial. *Mechanical Systems and Signal Processing*, 25(2):485 – 520, 2011.

[42] Paresh GirdharC. Scheffer, editor. *Practical Machinery Vibration Analysis and Predictive Maintenance*. Newnes, Oxford, 2004.

[43] Haidong Shao, Hongkai Jiang, Huiwei Zhao, and Fuan Wang. A novel deep autoencoder feature learning method for rotating machinery fault diagnosis. *Mechanical Systems and Signal Processing*, 95:187–204, 2017.

[44] Haidong Shao, Hongkai Jiang, Huiwei Zhao, and Fuan Wang. A novel deep autoencoder feature learning method for rotating machinery fault diagnosis. *Mechanical Systems and Signal Processing*, 95:187 – 204, 2017.

[45] Muhammad Sohaib, Cheol-Hong Kim, and Jong-Myon Kim. A hybrid feature model and deep-learning-based bearing fault diagnosis. *Sensors*, 17(12), 2017.

[46] Sandeep Sony, Shea Laventure, and Ayan Sadhu. A literature review of next-generation smart sensing technology in structural health monitoring. *Structural Control and Health Monitoring*, 26(3):e2321, 2019. e2321 STC-18-0009.R1.

[47] Dong Wang, Xuejun Zhao, Lin-Lin Kou, Yong Qin, Yang Zhao, and Kwok-Leung Tsui. A simple and fast guideline for generating enhanced/squared envelope spectra from spectral coherence for bearing fault diagnosis. *Mechanical Systems and Signal Processing*, 122:754 – 768, 2019.

[48] Tianyang Wang, Qinkai Han, Fulei Chu, and Zhipeng Feng. Vibration based condition monitoring and fault diagnosis of wind turbine planetary gearbox: A review. *Mechanical Systems and Signal Processing*, 126:662 – 685, 2019.

[49] Zhen-Ya Wang, Chen Lu, and Bo Zhou. Fault diagnosis for rotary machinery with selective ensemble neural networks. *Mechanical Systems and Signal Processing*, 113:112–130, 2018.

[50] Chunzhi Wu, Pengcheng Jiang, Chuang Ding, Fuzhou Feng, and Tang Chen. Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network. *Computers in Industry*, 108:53 – 61, 2019.

[51] Ying Xie, Peng Chen, Fei Li, and Haisong Liu. Electromagnetic forces signature and vibration characteristic for diagnosis broken bars in squirrel cage induction motors. *Mechanical Systems and Signal Processing*, 123:554 – 572, 2019.

[52] Xiaolong Xu, Qingxiang Liu, Yun Luo, Kai Peng, Xuyun Zhang, Shunmei Meng, and Lianyong Qi. A computation offloading method over big data for iot-enabled cloud-edge computing. *Future Generation Computer Systems*, 95:522–533, 2019.

[53] Ruqiang Yan, Robert X. Gao, and Xuefeng Chen. Wavelets for fault diagnosis of rotary machines: A review with applications. *Signal Processing*, 96:1 – 15, 2014. Time-frequency methods for condition based maintenance and modal analysis.

[54] Xiaoan Yan and Minping Jia. A novel optimized svm classification algorithm with multi-domain feature and its application to fault diagnosis of rolling bearing. *Neurocomputing*, 313:47 – 64, 2018.

[55] Rafael Gouriveau Kamal Medjaher Noureddine Zerhouni. *From Prognostics and Health Systems Management to Predictive Maintenance 1: Monitoring and Prognostics, Volume 4*, volume 4. ISTE Ltd, 10 2016.

[56] XiaoLi Zhang, Wei Chen, BaoJian Wang, and XueFeng Chen. Intelligent fault diagnosis of rotating machinery using support vector machine with ant colony algorithm for synchronous feature selection and parameter optimization. *Neurocomputing*, 167:260 – 279, 2015.

[57] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213 – 237, 2019.

Appendix A

WCIC Embedded Implementation

The Wavelet Convolutional Imbalanced Classifier (WCIC) as described in this thesis was implemented in a vibration sensor hardware and tested on a real application.

The experiments were executed on a test rig, used for data measurement, as the one presented in figure A.1.

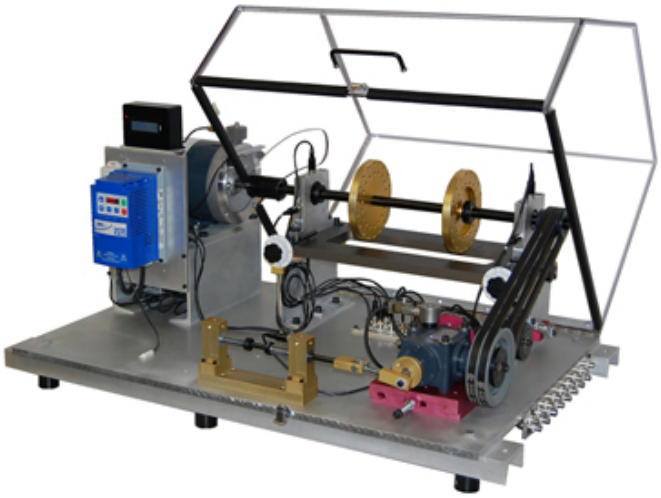

Figure A.1: Test rig for data measurement.

The test rig was measured in five different states, one in normal condition, and four in failure modes: unbalance, rolling element bearing with inner race fault, with outer race fault, and sphere fault. Every condition was measured 400 times, in three rotating speeds: 20, 25 and 30Hz. The hole dataset comprehends 6000 measurements.

The vibration sensor used for the embedded implementation, was developed by SEMEQ, and its technical characteristics are presented in table A.1.

Table A.1: Sensor technical characteristics.

Processor	Arm	Cortex-M0
Memory	Flash	256KB
	Internal RAM	32KB
	External RAM	256Kb (32KB)
Conectivity	Bluetooth	BLE 4.2
Accelerometer	Range	10KHz
	Acquisition Rate	24KS
	Sensitivity	40mVg
	Gain	0 a 48db

No floating point unit is available in the hardware, so all operations must be executed with integer values, or the processing takes a longer time, impacting on the sensors battery life.

Since there is little class imbalance in the dataset, all samples were used for training the model. Table A.2 shows ten randomly selected splits results. The model used was the same Wavelet Convolutional Imbalanced Classifier, proposed in the thesis, with the Fourth Root loss function proposed.

Table A.2: Results for the proposed model in the test rig.

Accuracy	**Recall**	**False Negatives**
0.999	1.0	0
1.0	1.0	0
1.0	1.0	0
0.998	1.0	0
0.989	1.0	0
0.999	1.0	0
1.0	1.0	0
0.998	0.998	1
1.0	1.0	0
1.0	1.0	0

Figure A.2 shows the training history for one of the random splits in table A.2.

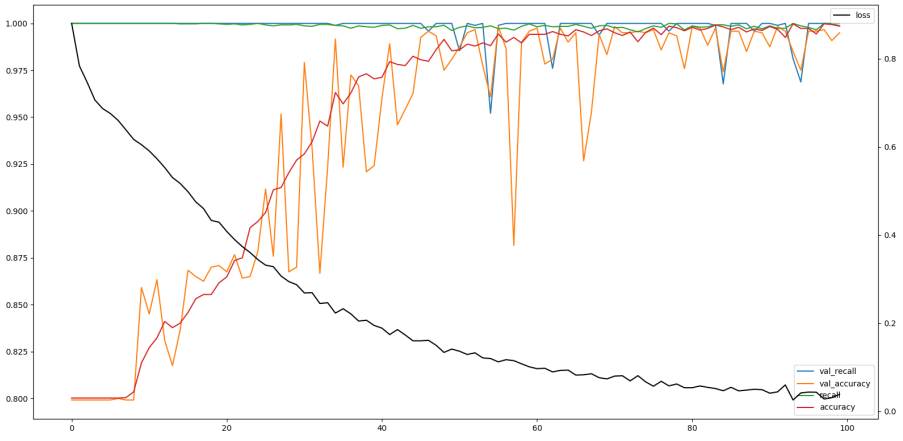

Figure A.2: Training history for the test rig.

For the controlled experiment, accuracy was in average 0.998 with standard deviation of 0.003. Recall was in average 0.999 with standard deviation of 0.001. False negatives was in average 0.1 with standard deviation of 0.3.

To implement the Wavelet Convolutional Neural Network (WCNN) embedded in the vibration sensor, the input vibration signal has 4095 points of length. The integer representation of values proposed in section 3.5 has 16 bits, or 2 bytes. So, after multiplying the input by the scaling factor of 10000, and integer truncation, the full length of the input signal is of 8190 bytes.

The model implemented in the vibration sensor is presented in figure A.3.

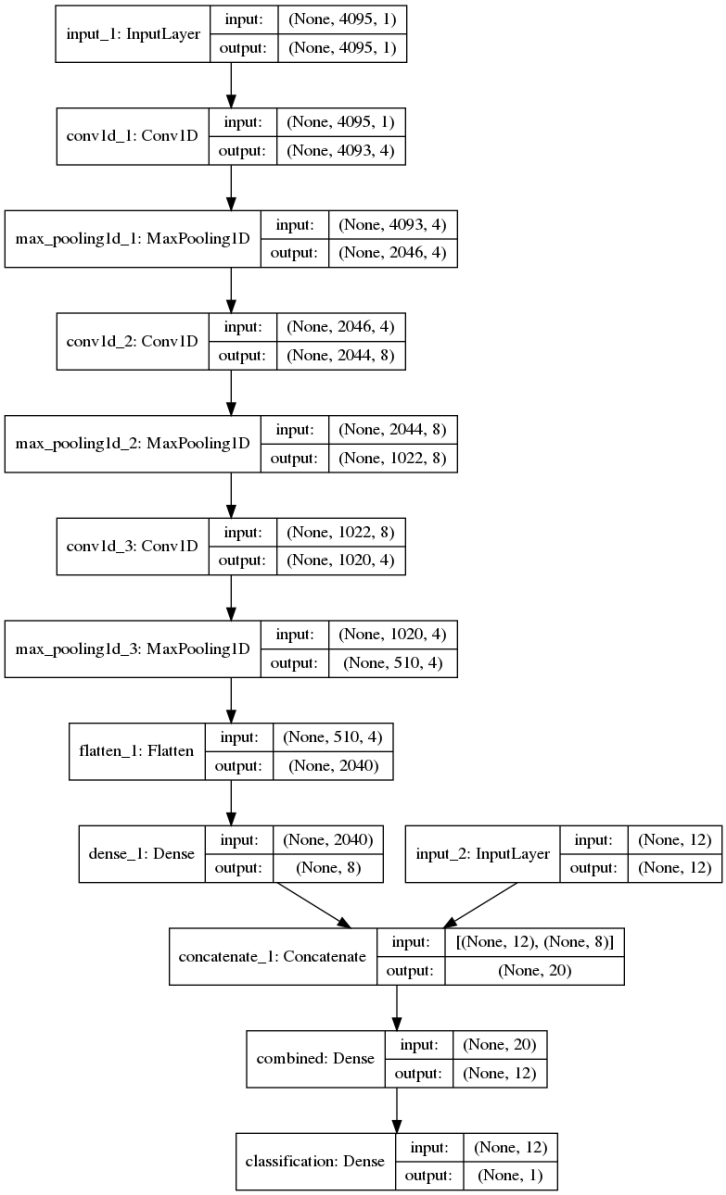

Figure A.3: Model implemented.

From the model dimensions proposed in section 4.1.3, the first convolution layer has 4 filters with 3 weights of length, and a max pooling factor of 2. So the resulting matrix of data has dimensions of $2046 \times 4 \times 2 = 16368$ bytes. The following layer have 8 filters, also with 3 weights of length and a max pooling factor of 2. So the data dimensions

are: $1022 \times 8 \times 2 = 16352$ bytes after the second layer. The third layer has 4 filters with 3 weights of length, and a max pooling factor of 2. So the data dimensions are: $510 \times 4 \times 2 = 4080$ bytes after the third layer. From the third layer, a densely connected neural network is implemented with 8 nodes. So are necessary $8 \times 2040 \times 2 = 32640$ bytes for this layer input.

Analyzing the dimensions, we can infer that the maximum memory necessary to implement the encoder phase embedded in the vibration sensor is about $16368 + 16352 + 4080 + 32640 + 8190 + 2000 = 79630$ bytes. But the total RAM memory available is of 64KB (adding the internal and external RAM). To be able to fit the necessary data into the memory, we reuse allocate arrays. For example, the result of the densely connected layer is stores on the same memory region of allocated for the first and second layers result. With that, the total amount of memory needed is of $79630 - 32640 = 46990$ bytes.

With the model trained, the weights can be exported and implemented in the sensor firmware. The weights are exhibited in the following code.

```
int bias1int[4] = {-8498485,-5787301,-4774846,-3312930};

short wght1int[3][1][4] = {{{4162,236,5644,-773}},
{{-6831,4743,866,4435}},
{{3937,-2338,-4288,-6958}}};
```

Listing A.1: wght1.h

```
int bias2int[8] = {188084,-462958,376401,-787396,-1270379,-990848,-62794,1250377};

short wght2int[3][4][8] = {{{-200,155,-2957,-939,2531,4462,3003,-6159},
{-1151,-3033,1724,2164,-3312,1803,-848,1911},
{3053,1160,-6259,1786,1070,865,-347,445},
{-4694,-5331,201,3701,4541,1630,-2588,-425}},
{{-101,-3274,693,-3081,-1853,1335,-3413,-5053},
{409,-2188,-1628,1600,2133,2517,-2987,1032},
{546,3003,-2338,-591,3691,-1275,4945,1051},
{-3806,-1327,1092,-1676,-2086,4611,160,-2145}},
{{-156,1501,-2593,-810,3502,2452,-4978,-2936},
{-3297,-2813,3563,-802,3132,277,265,-1561},
{-945,-1552,-4911,-45,4015,2619,-3045,-3398},
{-5528,-2856,-2947,-69,4524,4510,1197,-5602}}};
```

Listing A.2: wght2.h

```
int bias3int[4] = {-236314,-1188446,1373396,-1379857};

short wght3int[3][8][4] = {{{-4416,1832,1776,-2423},
{-4057,-1340,-5500,1854},
{-3587,3302,-2399,-2090},
{-4002,-378,-3341,-1782},
{-1439,-4222,568,-5039},
{2836,-4564,4849,-5083},
{-670,-4611,-6159,-1155},
{776,2465,-5098,-3094}},
{{-2556,5444,-2080,-2510},
{-6633,2837,-3530,-2640},
{2843,3836,1365,-3283},
{-3438,-851,-2975,121},
{3090,871,1046,-5424},
{-2433,-4607,187,-933},
{-6456,-266,-2192,-1019},
{-3955,3980,1338,1617}},
{{-4900,-1010,-2881,-36},
{-1889,-577,-1449,2157},
{1468,3170,4271,2388},
{-6380,-2569,-2643,-3167},
{2468,-5258,40,-1854},
{4210,256,-3502,-5353},
{-2162,-592,-5308,-2248},
{-3301,4992,-1992,2174}}};
```

Listing A.3: wght3.h

The weights for the last layer, of the densely connected nodes, are not included in this appendix because of its size, since there are $2040 * 8 = 16320$ different weights.

Comparing the results for all measured data, through the integer implementation of the wavelet convolutional neural network, the absolute difference of the result extracted features can be seen in table A.3.

Table A.3: Comparing integer implementation absolute errors.

	Mean	**Std Deviation**
MAE	0.358	0.065

The classification results after the truncation difference for integer implementation is demonstrated in table A.4.

Table A.4: Comparing integer implementation classifications.

	Float implementation	**Integer implementation**
True Negatives	1196	1197
False Negatives	4	3
True Positives	4800	4799
False Positives	0	1

The results show very little difference of integer implementation and full precision floating point implementation. The rest of the code that executes the integer implementation is presented as follows.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "wght1.h"
#include "wght2.h"
#include "wght3.h"
#include "wght4.h"
#include "data.h"

int FATOR=10000;
short vetor2[2046][4];
short vetor3[1022][8];
short vetor4[510][4];
short vetor5[2040];
int vlx,vly;

#define wvltlength 1000
int wvlt[wvltlength] = {-53532, -55161, -56835, -58556, -60325, -62143, -64012, -65932,

int wavelet_interp(int x) {
        int ind = (((float)x)/(8*FATOR*FATOR))*wvltlength + wvltlength/2;
        if (ind<0) ind=0;
        if (ind>999) ind=999;
        return wvlt[ind];
}

void conv_mpool_int(int rod,int nf,int lf,short wgts[lf][vly][nf],int bias[nf],int mp) {
        int k,i,j,l;
        int tmp[mp];
        int max=0,min=0;
        int soma=0;
        for (k=0;k<nf;k++) {
                for(i=0;i<vlx+1-lf;i++) {
                        soma=bias[k];
                        for(j=0;j<lf;j++) {
                                for(l=0;l<vly;l++) {
                                        if (rod==1) soma+=vetor1[i+j][l]*wgts[j][l][k];
                                        if (rod==2) soma+=vetor2[i+j][l]*wgts[j][l][k];
                                        if (rod==3) soma+=vetor3[i+j][l]*wgts[j][l][k];
                                }
                        }
                        if (soma>max) max=soma;
                        if (soma<min) min=soma;
                        tmp[i%mp]=wavelet_interp(soma);
                        if (i%mp==(mp-1)) {
                                for(j=1;j<mp;j++) {
                                        if (tmp[j]<tmp[j-1]) {
                                                tmp[j]=tmp[j-1];
                                        }
                                }
                                if (rod==1) {
                                        vetor2[(i+2-lf)/mp][k]=(tmp[mp-1]/FATOR);
                                }
                                if (rod==2) vetor3[(i+2-lf)/mp][k]=(tmp[mp-1]/FATOR);
                                if (rod==3) vetor4[(i+2-lf)/mp][k]=(tmp[mp-1]/FATOR);
                        }
                }
        }

        vlx=(vlx+1-lf)/2;
        vly=nf;
}
```

```
int main(int argc, char *argv[]) {
        int i=0,j=0,k=0;

        vlx=4095;
        vly=1;

        conv_mpool_int(1,4,3,wght1int,bias1int,2);
        conv_mpool_int(2,8,3,wght2int,bias2int,2);
        conv_mpool_int(3,4,3,wght3int,bias3int,2);

        k=0;
        for(i=0;i<vlx;i++) {
                for(j=0;j<vly;j++) {
                        vetor5[k]=vetor4[i][j];
                        k++;
                }
        }

        for(j=0;j<12;j++) {
                float soma=0;
                for(i=0;i<2040;i++) {
                        soma += ((float)vetor5[i])*wghtDense[i][j];
                }
                soma = ((float)(soma+biasDense[j]))/(FATOR*FATOR);
                if (soma<0) soma=0;
                printf("%f\n",soma);
        }

        return 0;
}
```

Listing A.4: main