SMART PARALLEL WAVELET TRANSFORMATIONS FOR EDGE AND FOG DETECTION OF BEARING DEFECTS

A Dissertation
Presented to
The Academic Faculty

By

Pierrick Rauby

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Georgia Institute of Technology

Georgia Institute of Technology

December 2021

Copyright © Pierrick Rauby 2021

SMART PARALLEL WAVELET TRANSFORMATIONS FOR EDGE AND FOG DETECTION OF BEARING DEFECTS

Approved by:

Dr. Thomas Kurfess, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Christopher Saldana
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Katherine Fu
Departement of Mechanical Engineering
University of Winsconsin-Madison

Dr. Vincent Paquit
Manufacturing Demonstration Facility
Oak Ridge Nationnal Laboratory

Dr. Ducan McFarlane
Institute for manufacturing
University of Cambridge

Date Approved: November 12, 2021

In a world of change, the learners shall inherit the earth, while the learned shall find themselves perfectly suited for a world that no longer exists.

Eric Hoffer

To my parents, Annie and Alain, for instilling in me the love of learning.

ACKNOWLEDGEMENTS

I would like to thank Professor Thomas Kurfess for his guidance during this work. He consistently helped me with the research and provided advice that allowed me to take a step back when I was going in the wrong direction.

I would also like to thank the other committee members, Dr. Fu, Dr. McFarlane, Dr. Paquit, and especially Dr. Saldana, for his understanding in his supervision of other research projects when the work on my dissertation took most of my time.

My thanks also go to the administrative staff of the Office of International Education and the School of Mechanical Engineering, especially Mrs. Glenda Johnson. Her explanation tremendously helped me with the paperwork for the different degrees I received from GeorgiaTech.

Finally, I must express my gratitude to Claire, her parents, and my family and friends for their support during this pandemic, which made my work in this unusual time easier.

Thank you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Rolling Element Bearings (REB) are critical components of a wide range of rotating machines. Identifying and preventing their faults is critical for safe and efficient equipment operation. A variety of condition monitoring techniques have been developed that gather large amounts of data using acoustic or vibration transducers. Further information about the health of an REB can be extracted via time domain trend analysis, and amplitude modulation technics. The frequency domain-specific peaks corresponding to the defects can also be identified directly from the spectrum.

Such approaches either provide little insight into the type of defect, are sensitive to noise, and require substantial post-processing. Complicating current fault diagnostic approaches are the ever-increasing size of datasets from different types of sensors that yield non-homogeneous databases and more challenging to execute prognostics for large-scale condition-based maintenance. These difficulties are addressable via approaches that leverage recent developments on microprocessors and system on chip (SoC) enabling more processing power at the sensor level, unloading the cloud from non-used or low information density data.

The proposed research addresses these limitations by presenting a new approach for bearing defect detection using a SoC network to perform a wavelet transform calculation. The wavelet transforms enable an improved time- frequency representation and is less sensitive to noise than other classical methods; however, its analysis requires more complex processing techniques that must be executed at the edge (sensor) to limit the need for cloud computing of the results and large-scale data transmission to the cloud. To enable near real-time processing of the data, the BeagleBone AI SoC is employed , the wavelet transforms, and the defect classification are achieved at the edge.

The contributions of this work are as follows: first, the real-time data acquisition driver for the SoC is developed. Second, the machine learning algorithm for improving the wavelet transform and the defect identification is implemented. Third federated learning in a network of SoC is formulated and implemented. Finally, the new approach is benchmarked to current approaches in terms of detection accuracy, and sensitivity to defect and was proven to obtain between 80 and 90 percent accuracy depending on the dataset.

CHAPTER 1
INTRODUCTION AND MOTIVATION

Rotating machines are present in the vast majority of industries and transportation; most of the time, they include Rolling Element Bearings (REB), which constitute a critical part of the machine. A defect in those bearings can, at best, generate vibrations that affect the machine functioning, at worst create a failure of the engine and result in more significant safety and economic issues.

Thus, as economic constraints and technical challenges increase, the maintenance strategies of these machines have evolved. Predictive maintenance is replacing periodic preventive maintenance, which replaced the run-to- failure approach. This evolution in maintenance strategies has been accelerated by Industry 4.0 and the Internet Of Things (IoT). It is now less expensive to mount sensors and microprocessors on a machine to perform condition monitoring.

However, even if it is cheaper and easier to gather data, it also requires more digital discipline to store the data efficiently and organized. Moreover, these data are often time never exploited and analyzed to extract information [1]. The economic loss is even more critical when the data takes a vast space in the Cloud, such as vibration data generally used for bearing defect identification. Current condition monitoring technics developed to identify faults in rolling element bearing rely on vibration to extract status information using time or frequency domain analytics. Besides requiring a large amount of data to be transmitted to the Cloud, those technics provide either little insight into the type and severity of the defect or require essential knowledge of the process parameters (such as rotation speed). These difficulties are addressable via approaches that leverage the recent development on microprocessors and system on chip (SoC) that enable more processing power at the sensor level. The proposed research uses SoC's network for processing wavelet transform to accomplish bearing defect detection. In the rest of this chapter the outline of this dissertation is presented. Thereafter, the different types of rolling element bearing and their maintenance is presented. Finally, a high-level overview of the concept of Industry 4.0, and the machine learning theory are presented.

1.1 Outline

The rest of this work is organized as follows: Chapter 1 will present the different types of rolling element bearings and the current methods used to detect their defects. Then the new developments in System-on-chip and their application to industry 4.0 are detailed and general theory about machine learning is presented. Once the motivation of this work is introduced, Chapter 2 presents the Background and literature review for this work which opens to the identification of the research questions presented in Chapter 3. Chapter 4 introduced the hardware used for deterministic data acquisition. The edge wavelet implementation is described with the processing of the data and the federated learning between multiple devices. The validation and performances of the proposed architecture and comparison with current methods are discussed in chapter 5. Finally, chapter 6 develops the contributions and limitations, and chapter 7 concludes the work.

1.2 Rolling Element Bearing

Rolling element bearing constitutes a critical component of machines; their role is to reduce the friction forces between two mechanical parts that are moving one to another: introducing a rotating element between the two surfaces of the parts enables to transform the friction between the surface into a motion of rotation. The use of such a concept can be traced back to Ancient Egypt around 2500 B.C., during the building of the Gizeh Pyramids. Later, Leonard De Vinci formalized a rolling element bearing (Figure 1.1).

In this section, a brief introduction to rolling element bearing is given, then the different maintenance strategies are presented as well as the state- of-the-art bearing defect detection methods. An introduction to the recent development of the Internet of Things and the platforms used for industrial applications are presented.

1.2.1 Bearing introduction

The bearing market is constantly increasing in size as the manufacturing and transportation sector grow. It is forecasted to reach US$213 billion by 2026, according to an Acumen market report. Bearings are in every single rotating machine, from cars containing up to 60 bearings to airplanes, trains, or manufacturing machines such as lathe or paper machines. As the need for bearings increases, it becomes essential to monitor bearing health as they constitute a critical source of failure: 50% of motors failures are due to bearing failures [3].

There exist multiple types of Bearings. Even if the most common is the rolling element bearing, it is worth mentioning fluid film element bearing and magnetic bearing.

Fluid film element bearings take advantage of the hydrodynamic effects in order to reduce the friction between the rotation parts. The film of lubricant such as oil separates the two parts, the motion of the two parts, one relative to the other, creates a pressure in the film, pressure that prevents the contact between the two parts. The application of such bearings is for high load, and high-speed application

Magnetic bearings use magnetic levitation to avoid friction between parts. They are typically reserved for an application that requires very high speed with low vibration

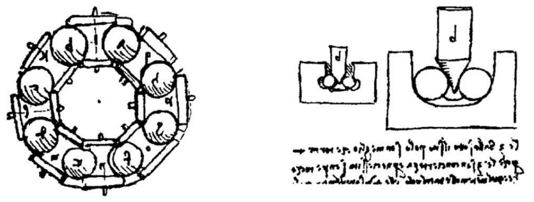


Figure 1.1: First drawing of a Rolling Element Bearing [2]

because of their cost. Contrary to other kinds of bearings, they are usually not passive pieces of equipment and require energy to maintain the levitation.

1.2.2 Types of rolling element bearings

Rolling Elements Bearings (Figure A.1) are typically comprised of 3 parts:

The races or rings: inner and outer races are respectively fixed to the two parts that rotate one to another. The races generally present a groove that keeps the rolling elements in place and transmits the mechanical load between the rotating parts and the rolling elements.

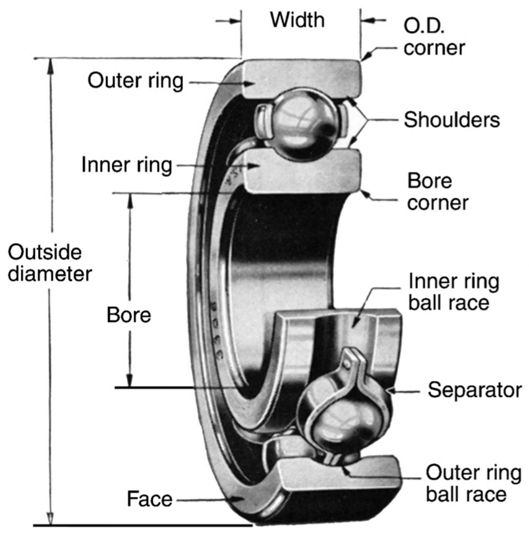
The rolling elements: are located between the races and are transmitting the load between them. They can be of various forms and shapes: balls, spherical rollers, tapered rollers, cylindrical rollers. The type of rolling element is related to the direction of the load applied to it. Table 1.1 from [4] presents the relative performances of rolling element bearings submitted to radial and thrust loads.

The cage or separator: when present, this part holds the rolling elements to reduce the friction between them.

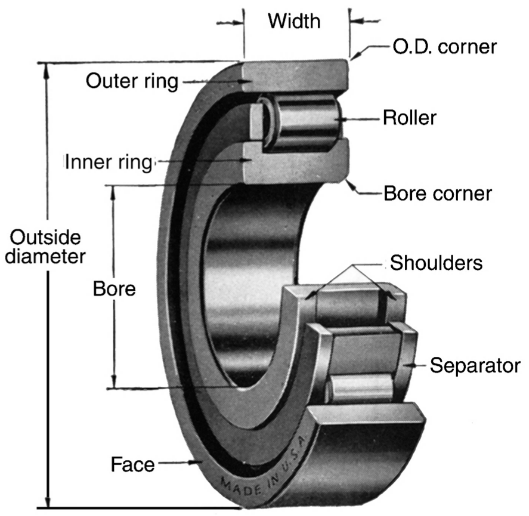
These different components can be made from a wide range of materials, from polymers for small loads applications to stainless steel or complex alloys for higher loads or dimension-sensitive applications. Besides, an essential element that influences the bearing life and performance is its lubrication; indeed, without lubrication, the bearing may deteriorate more rapidly, and defect appears earlier.

1.3 Rolling Element Bearing Maintenance

With time, all bearing element wears and ultimately develop defects. The different defects are generally: inner-race fault, outer-race fault, rolling element fault, or cage fault. Figure



(a) Ball bearing



(b) Straight rollers bearing

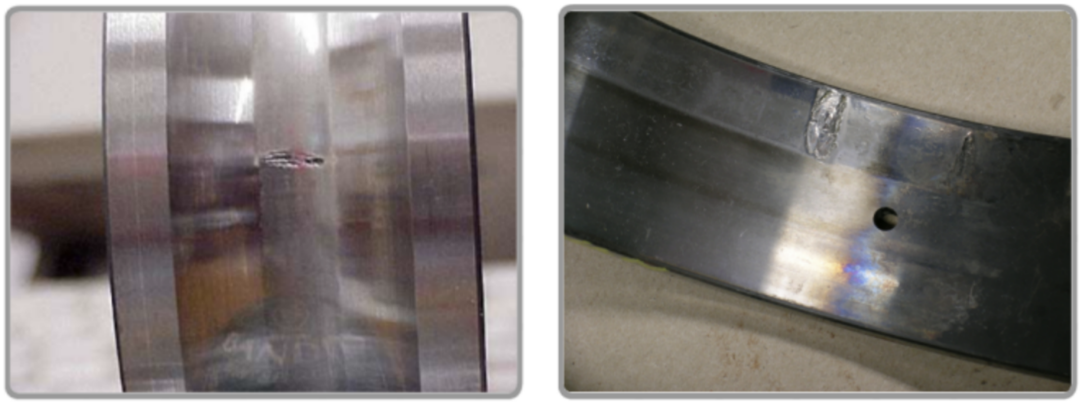
Figure 1.2: Components of rolling elements bearings[4]

Table 1.1: Relative performance for different types of rolling elements bearings for radial and thrust loads[4].

Bearing type	**Radial load**	**Thrust load**
Ball bearing	Good	Fair
Cylindrical roller bearing	Excellent	0
Tapered roller bearing	Excellent	Good
Spherical roller bearing	Good/Excellent	Good
Needle Bearing	Good	0
Thrust bearings	Excellent	0

1.3 presents the most frequent defects: inner-race and outer-race faults. Those faults generate high-frequency vibrations that can be used to monitor the bearing health (see section 2.3). Many parameters influence the bearing's life, such as lubrication, mounting quality (alignment with the axis of rotation), loading intensity, or manufacturing quality.

Figure 1.4 show the difference in the spectral signature between an outer race 1.4a and an inner race defect 1.4b, on the outer race defect the defect frequencies are clearly visible whereas for the inner race defect the defect is harder to identify with a large peak with side bands.



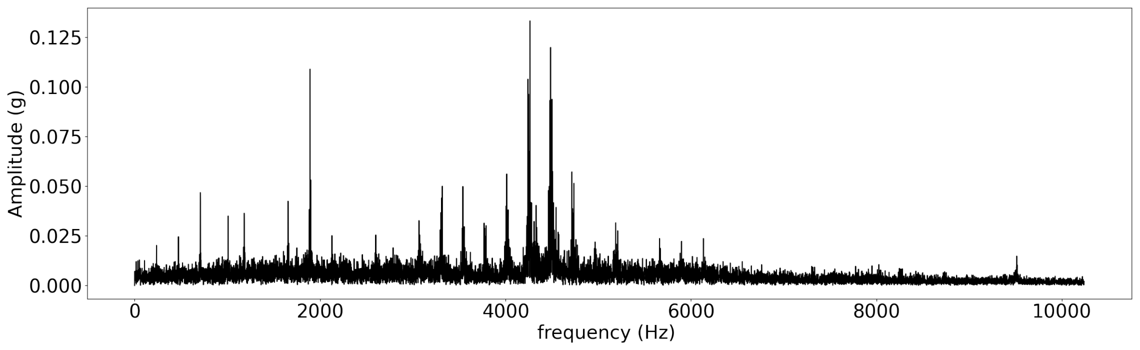
(a) Inner-race fault (b) Outer-race fault

Figure 1.3: Race defects [5].

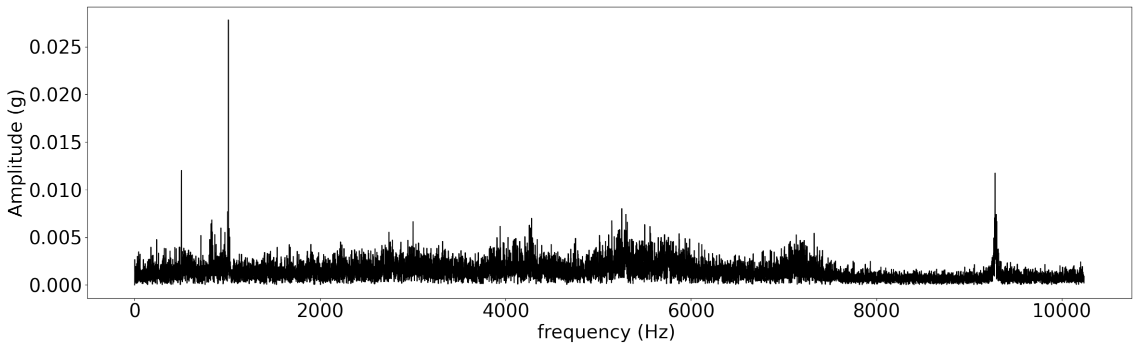
1.3.1 Maintenance strategies

As bearings often constitute critical components of machines, their maintenance has been extensively studied over the years. After the second world war, the maintenance strategies evolved from run-to-failure maintenance, in which operators replace the part when it breaks, to a preventive maintenance strategy, in which the parts are periodically changed before they crack. This new approach limits the risk of failure during the process and the resulting economic loss. However, it also requires sufficient knowledge of the parts' life and can become more expensive than the run-to-failure approach if expensive parts are changed too frequently [6]. Multiples bearing life models have been developed to predict the best interval to change a bearing, such as the SKF Generalized Bearing Life Model (GBML) [7]. More recently, with the development of processors and networks, another type of maintenance is possible, predictive maintenance. It aims to change the part just before the defect becomes critical. Usually, sensors are used to track the defect's development, and action is taken only when needed. It requires a complex understanding of defect evolution. Table 1.2 presents the different maintenance strategies, their limitations, and advantages.

The evolution of the maintenance strategies is closely related to the recent rise of low-cost systems on chip (SoC), communication protocols, and datacenters. They enable



(a)



(b)

Figure 1.4: Spectral signatures of an Outer Race Defect (1.4a) and an Inner Race Defect (1.4b)

Table 1.2: Different maintenance strategies

Maintenance strategy	**Run-to-Failure**	**Preventive Maintenance**	**Predictive Maintenance**
Maintenance interval	Fail & fix	Scheduled	Condition based
Cost effectiveness	Labor intensive	Cost depends on schedule	Cost effective
Technical complexity	Low	Medium	High

real-time prognostics on a wide range of manufacturing processes. The following section presents the influence of the internet of things (IoT) and its influence over maintenance strategies.

1.4 Internet of Things and Industry 4.0

The Internet of things is simply defined by [1] as : "sensors and actuators connected by networks to computing systems". In manufacturing, the application of the internet of things is often referred to as the 4th industrial revolution; it was named this way by Germany when, in 2011, the government invested 400 million euros in order to maintain the country competitivity in these new technologies. This 4th industrial revolution follows, introduction of the steam engine (1st revolution), the beginnings of mass production (2^{nd} revolution) and the automation processes and the introduction of robots (3rd revolution). Currently, machines are being connected to Cyber-Physical Systems (CPS), which will create networks of billions of interconnected objects [8]. Even if some technologies exist to analyze heterogeneous datasets, they are not particularly suitable for CPS [9]. Moreover, recent developments of System on Chip have opened new possibilities for data processing. In the following, the application of these new technologies to manufacturing is presented.

1.4.1 Internet of Things for Manufacturing

Cost efficiency is a major factor of competition in today's industry. The application of the Internet of Things to manufacturing is of great interest for manufacturing as it enables to control of the production process. Many initiatives such as Digital Twins [10], where a digital representation clowns a physical system; geo-localization solutions like the ones developed by Zozio[11] that track parts and production tools with a tracking device; or even to monitor asset health to prevent expensive failure or non-conform parts have been developed. All these applications of the Internet of Things to manufacturing rely on a common architecture that can be decomposed in :

- Data acquisition with some sensor.
- Data transmission from the sensor to a processing unit.
- Data processing to extract information.

The extracted information is then used to decide if action needs to be taken (e.g., change an element of the machine about the break) or improve the overall manufacturing process.

Data acquisition

With the ever-decreasing cost of sensors and Systems on Chip, acquiring large amounts of data is becoming very cost-effective. A low-cost acquisition system is generally composed of:

- A transducer that converts a physical quantity such as noise, vibration, light, temperature; into an analog or digital signal. They can be microphones, accelerometers, proximity probes, photodetector and rely on simple physical effect such as Piezoelectric, light dependent resistance or electromagnetic induction.
- A microcontroller or microprocessor manages the data acquisition and aggregates the data to transmit it. Recent years have seen the variety and power of micro-

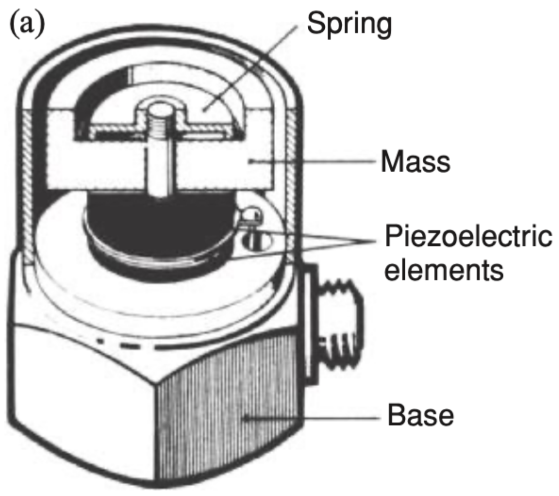


Figure 1.5: A Typical accelerometer design from Brüel & Kjær [12]

processors increase exponentially with SoC such as the BeagleBone or Raspberry board. The recent releases of those boards have seen the emergence of multi-cores Central Processing Units (CPU) and board Graphical Processing Units (GPU) with BeagleBone AI and the Raspberry Pi 4. These enable more processing at the edge see Section1.4.2.

- Finally, a communication port used to transmit the data can rely on technology such as USB, Ethernet, Bluetooth, Wi-Fi, LoRa, or others.

Transmission Protocols

The data transmission needs to follow an organized structure to ensure efficient communication between the sender and the receiver. Over the years, many communication protocols have been developed. Some of them were specially designed for IoT applications such as MQTT [13], which used a publish and subscribe messaging transport: a message broker is used to redirect the message from the publisher to subscribers to the message's topic. Others, like MTConnect, were designed especially for manufacturing data transmission [14], it uses the HyperText Transfer Protocol (HTTP) and XML format file to transfer machine-specific data as presented in Figure refMTConnect example[14].

Those different protocols ensure the efficient transmission of the data between the sender

```
<MTConnectStreams …>
  <Header … />
  <Streams>
    <DeviceStream name="mill-1" uuid="1">
      <ComponentStream component="Device" name="mill-1"
        componentId="d1">
        <Events>
          <Availability dataItemId="avail1" name="avail" sequence="5"
            timestamp="2010-04-06T06:19:35.153141">
            AVAILABLE</Availability>
        </Events>
      </ComponentStream>
    </DeviceStream>
    <DeviceStream name="mill-2" uuid="2">
      <ComponentStream component="Device" name="mill-2"
        componentId="d2">
        <Events>
          <Availability dataItemId="avail2" name="avail" sequence="15"
              timestamp="2010-04-06T06:19:35.153141">
              AVAILABLE</Availability>
        </Events>
      </ComponentStream>
    </DeviceStream>
  </Streams>
</MTConnectStreams>
```

Figure 1.6: An example of MTConnect Data

and the receiver. However, the data needs to be processed in order to extract value from it. Depending on the chosen architecture, the data can be processed at different locations.

Data Processing

This processing can either occur remotely in a data center, which constitutes Cloud Computing or at the sensor's level, referred to as Edge Computing. Both architectures have advantages and limitations.

Cloud Computing on the one hand, generally provides more processing power as the computing resources can be up-scaled to balance the load between multiple cores and processors. However, to process large amounts of data in the Cloud, the network connection needs to provide enough bandwidth to avoid a bottleneck at the sensor level. In the last years major actor have emerged in the Cloud Computing Market, Amazon Web Services (AWS), Azure and Google Cloud Platform (GCP).

AWS, is the dominant cloud infrastructure with 37% of market share. The offer a wide range of Virtual Machine and on demand services such as IoT Hub. The main disadvantage of AWS is the cost of its solution as well as the lack of clarity of its billing system.

Microsoft Azure comes second with 23% of the market share. Azure was the first could player to recognize the trend towards hybrid Cloud and release a Cloud-in-Your-Datacenter offer: Azure Stack. The main advantage of Azure is the integration between the cloud services with the rest of the Microsoft products such as Office. However, the support and documentation are the big shortcoming of Microsoft Azure.

The last major actor Google Cloud Platform, has 9% of the shares. Compared to the 2 previous one GCP offers the least number of virtual machines, but it has the benefit of being extremely user-friendly with tutorials for almost every service. Currently, its being chosen by customers whose business compete with Amazon and where there is little interest in staying inside the Microsoft Environment.

Edge Computing, on the other hand, has limited resources, which cannot be adapted without a physical intervention; but, it does not require a wide network bandwidth as only the information extracted from the data need to be transmitted. Finally, the security of the architecture is also increased by limiting the risk of data leaks.

Finally, some researchers have introduced an in-between cloud and edge computing known as fog computing. The data is processed at some level between the machine and the Cloud. A proposed architecture from Wu et al. [15] is using a local server to store the data and export it to the Cloud; predictive models are then trained at the cloud level and exported back to the local server where inference is made on the data. This use of fog computing takes advantage of lower bandwidth requirements for the data without losing too much computing power compare to an edge implementation.

Once the data is processed, users can retrieve valuable maintenance information that should be used to prevent some machine failures and enable better control of the manufacturing processes [12]. The new development of SoC and networks enable not only to process data sequentially on the onboard CPU but also to take advantage of the other onboard process units such as the Digital Signal Processor (DSP) or Graphical Process Unit (GPU). The following section presents different types of internet of things platforms.

1.4.2 Internet of Things platforms

In order to accomplish data acquisition and, if needed, the data processing, some Internet Of Things platform have been developed. They fall under 2 main categories:

- Microcontrollers (MCU)
- Microprocessors (MPU)

Microcontrollers

They are limited to one single program execution. Indeed, microcontrollers do not have any operating system which does not enable multiple processes to run simultaneously. Even if this limits the applications, it also permits a deterministic program's execution. Moreover, the clock frequency of microcontrollers is generally limited to hundreds of MHz; as the power consumption of a chip changes linearly with its clock frequency, it enables a low power consumption. The deterministic execution and the low power consumption make microcontrollers very suitable for data acquisition. Microcontrollers are generally composed of (Figure 1.8 from [16]):

- In/Out interfaces
- timer
- RAM memory for data storage (volatile)

- ROM memory to store the programs
- Central Process Unit (CPU)
- Analogue to Digital Convert (ADC) is also present on most of the microcontrollers

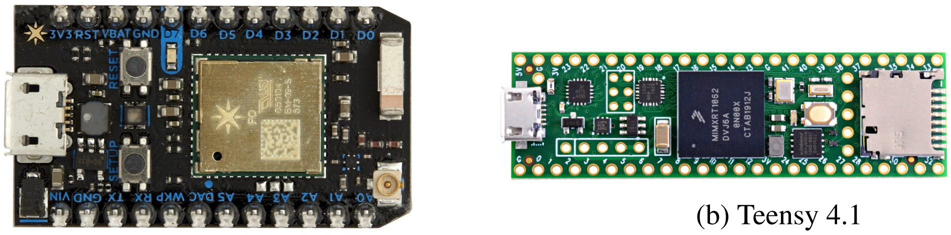
The Arduino Company originated in Italy at the Irrea Interaction Design Institute [17]. It aims at making available a large variety of products for easy prototyping. The company offers multiple ranges of boards from beginning and education boards like the (Arduino Uno, Leonardo, micro) and more advanced features for IoT applications requiring BLE, SIM cards, GSM, or more PINS and processing powers.

Espressif is a Shangai-based company that commercializes the ESP32 family of microcontroller. The four microcontrollers of the family are the C3, C6, S2, and S3, with CPU frequencies at 160MHz or 240MHz [18]. The S series has around 40 pins, while the C series only have about 20 pins. The ESP32-C3, S3, and S6 have Bluetooth 5 and BLE that the S2 does not have. These boards are less user-friendly than the Arduino boards, but their price makes them a good option for advanced users.

Particle is a more recent company started in 2013 in the US [19]. The company provides a more integrated solution focused on IoT application and connectivity; the ecosystem is oriented toward providing a complete solution to clients' problems and is more closed than for other competitors such as Arduino.

Teensy are produced by pjrc; a small company based in Oregon (USA) and mostly run by Paul J Stoffregen and Robin Coon [20]. The last Teensy (Teensy 4.1) is based on an ARM Cortex-M7 at 600MHz, 42 I/O pins with I2C, SPI, and Ethernet capabilities.

Table 1.3 from [16] presents the characteristics of 4 common microcontrollers:



(a) Particle Photon

(b) Teensy 4.1

Figure 1.7: Particle Photon and Teensy 4.1 [20][19].

Table 1.3: Comparison between different microcontrollers

Characteristic	**Arduino Uno**	**Tensy 3.2**	**Particle Photon**	**ESP32**
Processor	ATmega328P	ARM Cortex-M4	ARM Cortex-M3	Tensilica Xtensa
Frequency	16MHz	72MHz	120MHz	240MHz
GPIOs	14	34	18	34
ADCs	6	21	8	18
SPI/I2C	yes	yes	yes	yes
Wi-Fi/Bluetooth	on shield	No	yes/no	yes
RAM	2KB	64KB	128KB	520KB
EEPROM	1KB	2KB	16KB or 64 KB	448KB
Flash Memory	32KB	256KB	1MB	2MB or 4MB
Dimensions(mm)	68.6 by 53.4	35 by 18	36.6 by 20.3	55 by 28mm
Weight(g)	25	15	5	10
Price	35	20	20	15

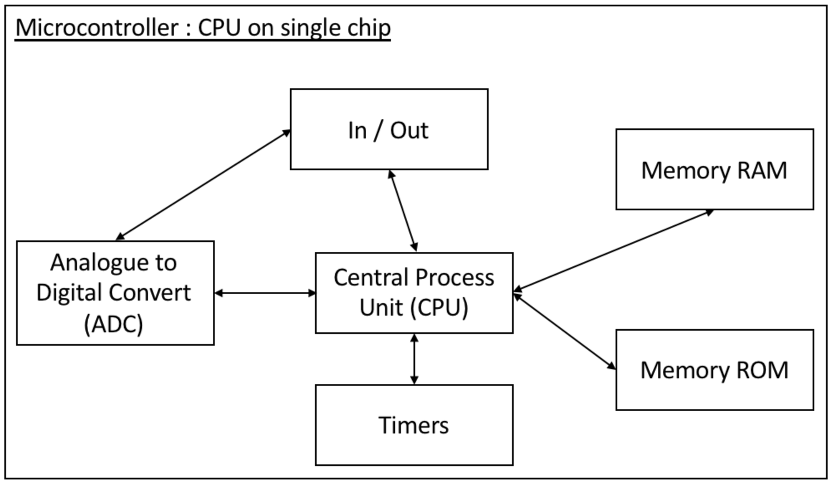

Microcontroller : CPU on single chip
In / Out
Memory RAM
Analogue to Digital Convert (ADC)
Central Process Unit (CPU)
Memory ROM
Timers

Figure 1.8: Architecture of a microcontroller.

Microprocessors

Contrary to microcontroller they are not limited single loop execution, and could be considered more like microcomputer. They usually have more processing power and memory than microcontrollers, as well as more peripheral such as multimedia port. This enables more complexes operations at the edge than with microcontrollers.

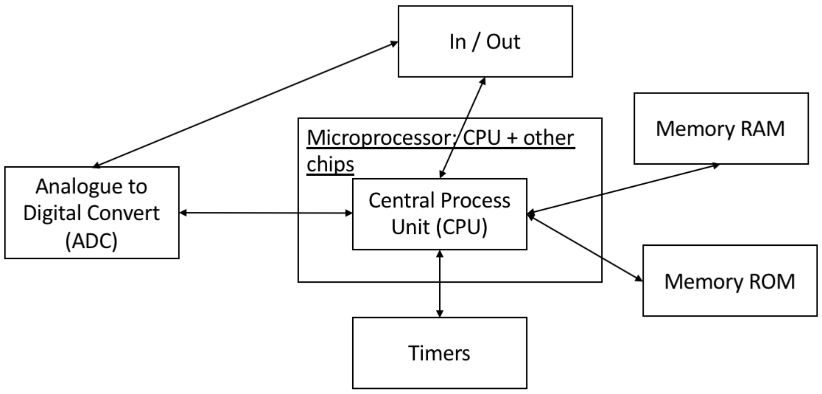
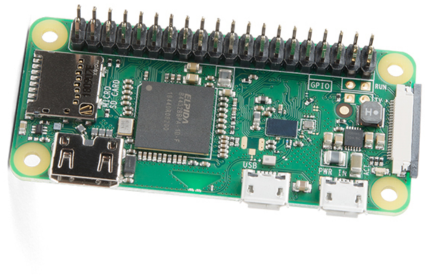
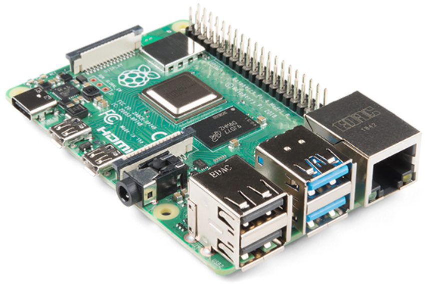

In / Out
Microprocessor: CPU + other chips
Memory RAM
Analogue to Digital Convert (ADC)
Central Process Unit (CPU)
Memory ROM
Timers

Figure 1.9: Architecture of a microprocessor.

Raspberry the Raspberry Pi foundation releases single-board microcomputers. They have eight different models of boards. Some are simple low power systems such as the Raspberry Pi Zero, costing as low as $10. Because the Raspberry Pi Zero has a 1GHz


(a) Raspberry Pi Zero W.


(b) Raspberry Pi 4.

(c) The Nvidia Jetson Nano

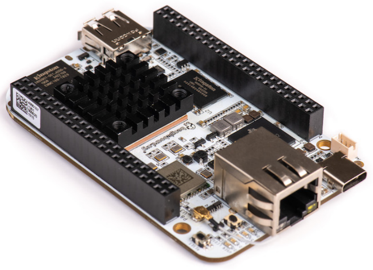

(d) The BeagleBone AI

Figure 1.10: Four different microprocessors from [21][22][23].

CPU, 512MB of random access memory, Bluetooth, Wi-Fi, and 40 header pins, it is well suited for IoT projects with low computing power requirements. However, it is not powerful enough to carry out intensive computation at the edge. Models such as the Raspberry Pi 4 ($35) will be preferred as they present a Quad-Core 1.5GHz processor with up to 8GB of RAM and 4 USB ports. Nevertheless, Raspberry's more advanced boards are more oriented to be low-cost computers with good multimedia capabilities than IoT platformed targeting data acquisition and processing at the edge. Similarly to the Arduino boards, the main advantage of Raspberry's boards is the size of the community and open source project based on these microcomputers.

Nvidia Jetson The Jetson Familly is produced by the Nvidia corporation, the world leader in the production of Graphical Processing Units. The Jetson's are much more expensive than the Raspberry with a price range from \$100 to \$780, but they embed GPUs, bringing parallel computing capabilities to the edge. The Jetson nano (\$99) has a 128 core Nvidia Maxwell GPU, a quad-core CPU at 1.43GHz, and a 2GB memory. The 40 header pins enable the use of multiple GPIOs, I2C, SPI, and UART sensors. However, the absence of an onboard ADC limits the applications to digital sensors. As Nvidia makes the GPU, the CUDA programming platform can be used. It makes parallel programming more accessible than using CUDA's open-source counterpart, OpenCL. It constitutes the main advantage for the Jetson boards over other edge GPU boards.

BeagleBone AI The BeagleBone foundation has released the BeagleBone AI to take advantage of the most recent Texas Instrument chips. The AI is based on the Ti-AM5729 chip and includes a Dual ARM Cortex A15 at 1.5GHz, 2 C66x floating-point digital Signal Processors, 4 Embedded Vision Engines, two dual Cores Programming Real Times Unites, and a PowerVR GPU. There are more In and Out Interfaces on the BeagleBone AI than on any other boards of its category, with two 46 header pins with GPIO, I2C, SPI, UART, and Analog capabilities. The BeagleBone organization is also releasing Open Source designs, so the board can be used in production or as an Evaluation Module where the board is used for prototyping, and a specific PCB is then designed to use in the production phase. The extensive In/Out capabilities of the BeagleBone AI and the different specialized processing units make it a very well-suited board for application in IoT for manufacturing.

Comparison

Above were presented the two main types of IoT Platforms. The microcontrollers do not have an operating system but benefit from a deterministic execution of the instruc-

Table 1.4: Comparison between different microprocessors[21]

Characteristic	**Raspberry Pi Zero**	**Raspberry Pi 4**	**Nvidia Jetson Nano**	**BeagleBone AI**
Processor	Single Core	Quad-Core ARM A72	Quad-Core ARM A57	Dual-Core ARM A15
Frequency	1GHz	1.5GHz	1.43GHz	1.5MHz
GPIOs headers	40	40	52	2x46
ADCs	No	No	No	7 Chan
SPI/I2C	yes	yes	yes	yes
Wi-Fi/Bluetooth	yes/yes	yes/yes	yes/yes	yes/yes
RAM	512MB	2Go	2Go	1GB + 16Go eMMC
Dimensions(mm)	66mmx30mm	85mmx56mm	100mmx80mm	94mmx55mm
Weight(g)	9	50	250	48
Price	10	35	59	120

tions and a generally cheaper cost than the microprocessors, which are more powerful non-deterministic single board computers. The recent developments of that single-board computer have seen the apparition of processors dedicated to intensive computation such as embedded Graphical Processing Units or Digital Signal Processors. Those processors enable expensive computation such as machine learning models to extract valuable information at the edge without requiring data transmission to the cloud. In the next section we present a high-level overview of the machine learning theory and some of its application in prognostics for manufacturing.

1.5 Machine Learning for defect detection

Industry 4.0 transforms the way we are producing parts. Machine Learning, as a subfield of artificial intelligence plays a very important role in this transformation. As machines are increasingly connected to sensors and the cloud, a very important amount of data is generated, it can be used to train machine learning algorithms. Those “learning” techniques are useful, when:

- humans expertise does not exist
- humans are not able to explain their expertise
- prediction problems involve a high level of complexity

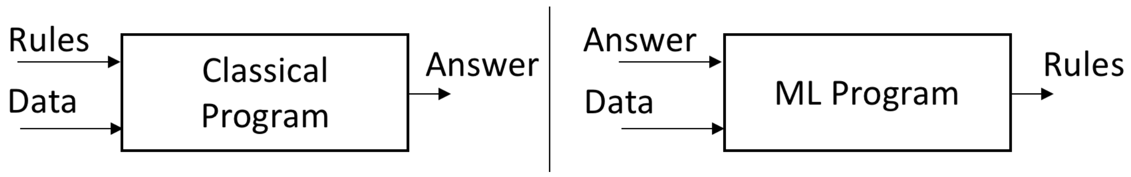


Figure 1.11: Classical programs and Machine Learning programs.

Figure 1.11 presents the difference between classical programs and machine learning problems. In the first ones, data and rules are provided as an entry, and the program gives an answer to the problem. In contrast, for machine learning programs, the entries consist in Data and already known answers; then the program establishes rules over this training set of data. Numerous studies have been conducted on the use of machine learning techniques for manufacturing prognostics.

M. Elangovan et al. [24] have discussed the effect of the Support Vector Machine (SVM) errors functions on the classification of vibration signals for single point cutting tools. The condition of a carbide tipped tool is predicted using a Kernel Support Vector Machine for two different error functions C-SVC and ν-SVC. The efficiency of these functions is then compared to other classifiers such as Decision-Trees, Naïve Bayes and Bayes net. It was found that, either for C or ν errors functions, the RBF Kernel gives higher classification efficiency. Finally, the linear Kernel can be interesting when it comes to have very fast classification. In comparison with other classification algorithms, the Kernel Support Vector Machine (KSVM) with ν-SVC has better efficiency. Then M. Elangovan et al. have shown that KSVM are promising for the prediction of the condition of a single point cutting tool.

C. Drouillet et al. [25] have used the neural network technique to predict tool life by monitoring the spindle power. End milling operations were performed on a steel work, and different MATLAB ™ learning functions were used to train a Neural Networks (NN). This method has demonstrated a good correlation between true and computed Remaining Useful Life (RUL); also it was very fast and could be used for Realtime RUL prediction.

Y. Fu et al. [26] have implemented Convolutional Neural Networks (CNN) for processing images representations of vibration signals. The vibration states have been considered to be a very promising way to real-time monitor machine states. Feeding the algorithm with an image of the signal without any preprocessing avoids possible bias introduced by the feature selection. Finally, the trained CNN showed very good results.

P. O'Donovan et al. [27] have introduced a fog computing industrial cyber-physical system for embedded low-latency machine learning application. Their research highlights that fog computing can be employed for real-time monitoring; this architecture enables a more distributed and scalable network while enhancing the privacy and the security of data.

Different machine learning algorithms have been implemented over the above-mentioned studies. A review of the different available techniques must be conducted in the following. First the difference between supervised and unsupervised machine learning is introduced, then the most well-known supervised ML methods are presented.

1.5.1 Supervised and unsupervised machine learning

As explained above, to be trained, machine learning algorithms usually expect Data and the "answer" of the problem. However, sometimes the output is not known, and this is where the unsupervised machine learning is promising. The goal of these algorithms is to highlight the structure or the distribution of the data, thus it aims to learn a new data's representation. The 2 major techniques of unsupervised machine learning are:

dimensional reduction: a data set of high dimension is reduced to lower dimension while keeping the "important" characteristics. Thus, the redundancies are removed, the storage space and the computational power required to manage the dataset are reduced, finally data visualization and interpretation is improved.

clustering: the general characteristics of the data are understood, then the different object of the data set can be grouped based on those characteristics. Again, the data interpretability is improved.

However, most of the time the answers of the problems for the training sets are known; then it is called supervised learning. The aim is to make predictions rather than to enhance the data interpretability. The predictions can either be in the form of a decision function or of a classifier, that can be binary or multi-class. The mains techniques of supervised machine learning are:

- Decision Trees
- Naïve Bayes classifiers
- Logistic Regression
- Support Vector Machine
- Kernel Support Vector Machine
- Neural Networks

1.5.2 Supervised algorithms

Decision trees

Decision Trees can be used in other fields, but when it comes to machine learning, they are applied to predict the value or the class of an output based on given inputs; to that end these algorithms repetitively divide the working area into subs-sets, which are divided again and again: "A decision tree is a recursive partition of the training set into smaller and smaller subsets" [28]. For data to be used in a Decision Tree model it needs to be discreet and without any ordering (e.g. classify fruit from color, shape, texture, size). Given a split variable j and a splitting point s, two regions (left and right) can be defined with:

$$R_l = x : x_j \leq s \text{ and } R_r = x : x_j > s$$

For regression problems, j and s have to be chosen in order to minimize:

$$\min_{j,s}\left(\sum_{i:x_i\in R_l(j,s)}(y_i-c_l)^2+\sum_{i:x_i\in R_r(j,s)}(y_i-c_r)^2\right)$$

For classification problems, j and s have to be set such that the impurity is minimized:

$$\min_{j,s}\left(\frac{|R_l(j,s)|}{n}\cdot \mathrm{Imp}\left(R_l(j,s)\right)+\frac{|R_r(j,s)|}{n}\cdot \mathrm{Imp}\left(R_r(j,s)\right)\right)$$

The *impurity* $\mathrm{Imp}()$ can be either:

Classification error: the minimum probability that a point is misclassified at the node (j,s) of the Tree:

$$\mathrm{Imp}(R_m)=1-\max_k \hat{p}_{mk}$$

with $\hat{p}_{mk}$ the portion of well-classified points.

Shannon's Entropy: from information theory

$$\mathrm{Imp}\left(R_m\right)=-\sum_k \hat{p}_{mk}\log_2 \hat{p}_{mk}$$

Gini impurity: with still $\hat{p}_{mk}$ the portion of well-classified points.

$$\mathrm{Imp}\left(R_m\right)=\sum_{k=1}^{K}\hat{p}_{mk}\left(1-\hat{p}_{mk}\right)$$

Decision Trees present many advantages; they are easy to understand and to interpret, as they are a mirror to human decision-making; however their predictive accuracy is not very good.

Naïve Bayes classifiers

This classifier uses the posterior probabilities also called *Bayes Theorem* 1.1 to make predictions.

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)} \tag{1.1}$$

For a binary classification problem, the aim is to express the probability distribution in a parametrized form. The probability of a single data point can be written as :

$$p_\theta(x, y) = p_\theta(y, x_1, ..., x_D) \tag{1.2}$$

Thanks to the Bayes Theorem 1.1 and the Naïves Bayes assumption, which states that $p(x_d|y, x_{d'}) = p(x_d|y) \; \forall \; d' \neq d$, the equality 1.2 simplifies:

$$p_\theta(x, y) = p_\theta(y) \prod_D p_\theta(x_d|y) \tag{1.3}$$

Then, depending on data type: binary, continuous... the model of $p(y|x_d)$ can be rewritten using respectively Bernoulli distribution and Gaussian distribution. Finally, the classification is the output is the class that is the more likely to be true.

Regression algorithms

Regression algorithms use the training data to fit curves and find a predictive function that maps the inputs to a continuous output $y = f(x_1, ..., x_n)$, depending on the number of features and the complexity of the relationship, different models can be used: the linear regression adjusts the coefficient b_i on the following equation $y = \sum_i b_i \cdot x_i$ in the case of n features; for more complex problems a polynomial regression can be used $y = \sum_i b_i x^i$. Finally, for some problems the logistic regression can be employed (here with the sigmoid function) $\log\left(\frac{p}{1-p}\right) = b_0 + b_1 \cdot x$

Support Vector Machine

Those algorithms are used to classify linear separable data points; as presented in Figure 1.12 (left). However, different margins can be found for the same data set, and they do not split the dataset equally. Support Vector Machine (SVM) tends to find the best linear boundary between different classes by using a constrained optimization problem, which reads as:

$$\min_{\underline{w},b} \frac{1}{\gamma\left(\underline{w},b\right)} + C \cdot \sum_{n} \xi_n \tag{1.4}$$

with respect to : $y_n(\underline{w} \cdot x_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0$. In formula 1.4, $\gamma\left(\underline{w},b\right)$ is the value of the margin γ which depends on the weight vector $\underline{w}$ and the bias b, ξ_n is the "cost" of having a data point, which is not classified correctly as presented in Figure 1.12 (right). The distance between two points x^+ and x^- at 1 unit from the margin read, as:

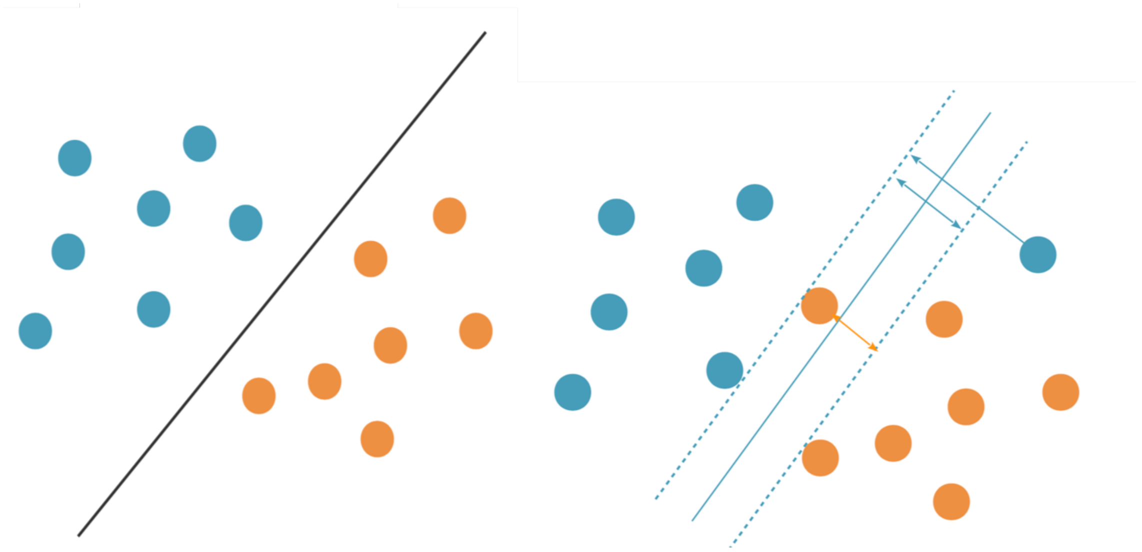


Figure 1.12: Linearly separable points (left), non-linearly separable data point (right). [29]

$$\begin{aligned} d^+ &= \frac{1}{\|\underline{w}\|} \cdot \underline{w} \cdot x^+ + b - 1 \\ d^- &= \frac{1}{\|\underline{w}\|} \cdot \underline{w} \cdot x^- - b + 1 \end{aligned} \tag{1.5}$$

So the margin γ can be expressed this way:

$$\gamma = \|d^+ - d^-\| = \frac{2}{\|\underline{w}\|} \tag{1.6}$$

and the constrained optimization problem is now to minimize the norm of the weight vector $\underline{w}$:

$$\min_{\underline{w},b} \frac{\|\underline{w}\|}{2} + C \cdot \sum_{n} \xi_n \tag{1.7}$$

with respect to: $y_n(\underline{w} \cdot x_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0$. As ξ_n must be positive but also minimum, it can be written that: $\xi_n = 1 - y_n(\underline{w} \cdot \underline{x_n} + b)$ (value of the classification error) if the point is not classified correctly and $\xi_n = 0$ if the point is classified correctly. Introducing $l^{(hin)}$ the hinge loss function as :

$$l^{(hin)}(a, b) = max(0, 1 - a \cdot b)$$

the term $\sum_n \xi_n = \sum_n l^{(hin)}(y_n, \underline{(w)} \cdot \underline{x_n} + b$
and equation 1.7 becomes:

$$\min_{\underline{w},b} \frac{\|\underline{w}\|}{2} + C \cdot \sum_{n} l^{(hin)} \left(y_n, \underline{(w)} \cdot \underline{x_n} + b \right) \tag{1.8}$$

Finding the minimum of the equation above gives information about the position of the optimum boundary. Although, this kind of algorithm is efficient for linearly separable or non-linearly separable data points with only few problematic points, sometimes, a linear boundary cannot be found between the categories (Figure 1.13) In these non-linear spaces,

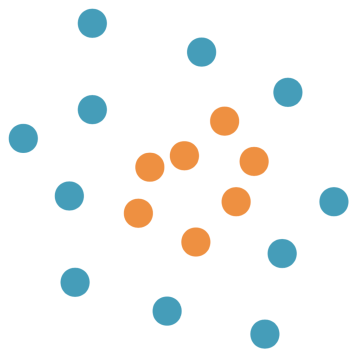


Figure 1.13: Data set where no linear boundary can be found. [29]

the use of a Kernel function is needed.

Kernel Support Vector Machine

Kernel functions can be used with a mapping Φ that projects the data points from the object space to a feature space where linear methods can be used, as in Figure1.14 A

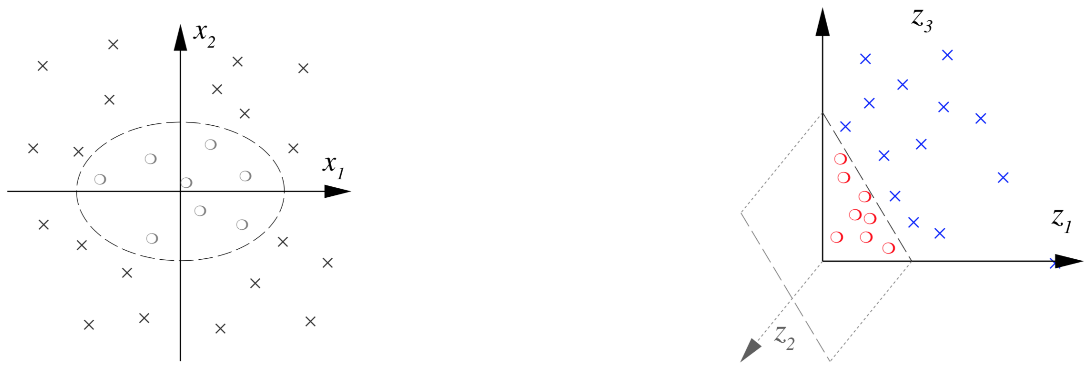


Figure 1.14: Mapping Φ from the data space $\mathcal{X}$ and the feature space $\mathcal{H}$ [30]

function $K(x, x')$ defined on a set $\mathcal{X}$ is called a Kernel function if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ such that for any x, x' in $\mathcal{X}$: $K(x, x') = \langle \Phi(x) \cdot \Phi(x') \rangle$. This enables us to use linear techniques but, more importantly the explicit computation of $\Phi(x)$ can be avoided, and $K(x, x')$ is computed instead. A Kernel Support Vector Machine (KSVM) is useful to classify data points where the data cannot be linearly separated in the data space and more importantly, in most cases Kernel methods reduce the computational power need. Thus, they are suitable for classification problems.

Finally, the most famous algorithms for machine learning are Neural Networks, section 1.5.2 presents different type of Neural Networks: Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

Neural Networks

These algorithms try to replicate the way neurons work. The neuron is modeled with a perceptron, as in Figure 1.15 and its output is given by $f(x) = s\left(w_0 + \sum_{j=1}^{P} w_j \cdot x_j\right) =$

$s\left(\underline{w}^T\underline{x}\right)$ where $s()$ is the threshold function. Other functions such as the sigmoid $\sigma = \frac{1}{1+\exp(-u)}$ can be used.

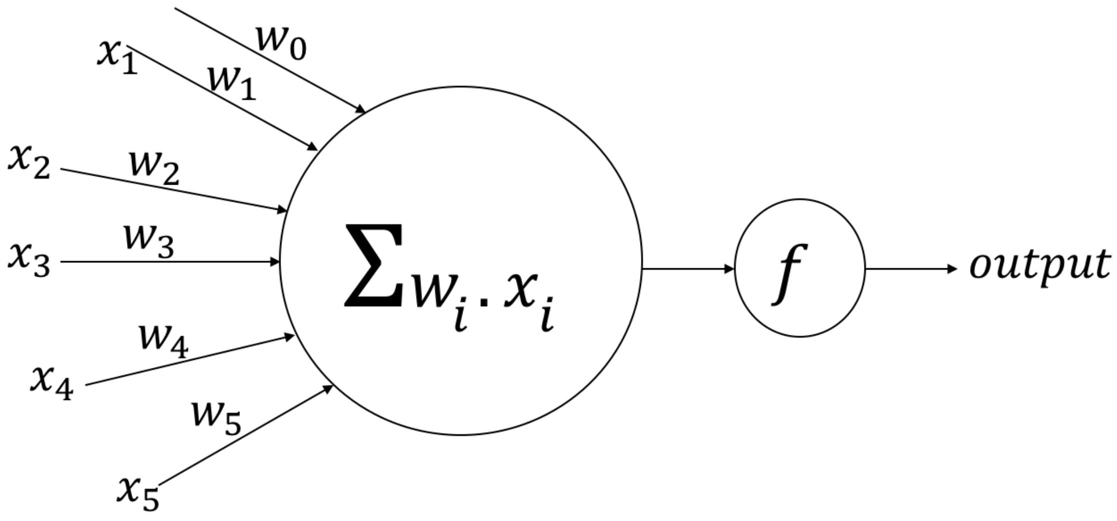


Figure 1.15: Data set where no linear boundary can be found. [31]

For binary classification (using the sigmoid function), the perceptron can be trained by adjusting all components of the weight vector $\underline{w}$ over the data set. For classification problems the cross-entropy error is generally used (η denotes the learning rate):

$$\mathcal{H}\left(f(\underline{x^i}), y^i\right) = -y^i \cdot \log(f(\underline{x^i})) - (1-y^i) \cdot \log(1 - f(\underline{x^i})) \tag{1.9}$$

Then the weight update for every iteration reads as:

$$\begin{aligned} \Delta w_j &= -\eta \frac{\partial \mathcal{H}\left(f(\underline{x^i}), y^i\right)}{\partial w_j} \\ \Delta w_j &= \eta \left(y^i - f(x^i)\right) x_j \end{aligned} \tag{1.10}$$

However, in the case of multiclass classification, the softmax function, equation 1.11, is used to find which class is more probable than the other. If class k is more probable than the other than $\sigma_k(x) \approx 1$ else $\sigma_k(x) \approx 0$.

$$\sigma_k(x) = \frac{\exp(\underline{w^k}^T \cdot \underline{x})}{\sum_{l=1}^{K} \exp(\underline{w^l}^T \cdot \underline{x})} \tag{1.11}$$

Then, the weight update reads as $\Delta w_j^k = \eta \left(y^i - f_k(x^i) \right) x_j^i$. Finally, for each training instance: $w_j^{t+1} = w_j^t + \Delta w_j^t$.

Adding several layers of Perceptrons as presented in Figure 1.16

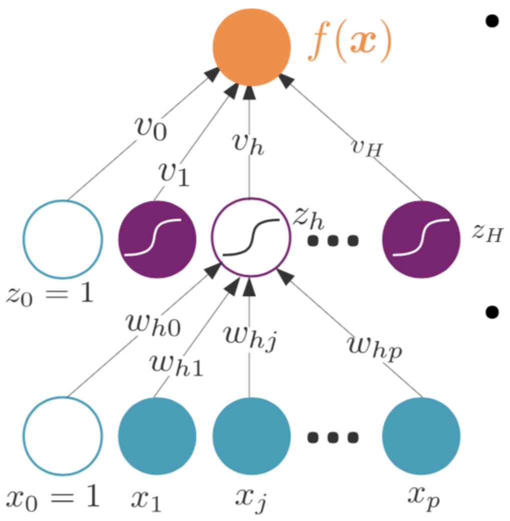


Figure 1.16: Multi-layer Perceptron structure. [32]

It is composed of 3 or more layers of Perceptrons, each layer feeding the following one. This algorithm is efficient for non-linear data classification.

Convolutional Neural Networks, on the contrary add more layers, the first operation transforms the input into feature maps that compose the convolution layer; then after one or multiple convolution maps a rectification layer is applied with functions such as ReLU, sigmoid... At the end, the last layers consist of a common Multi-layer perceptron. Convolutional neural networks are mostly used for image processing, however, Y. Fu et al. [26] have used them for Machining vibration states monitoring based on image representation. The advantage of this technique is that they were able to reduce the bias introduced by feature selection that must be performed for other machine learning methods such that Kernel Support Vector Machine.

Finally, Recursive Neural Networks (RNN) add more connections between the hidden layers of a Convolutional Neural Networks. The nodes are fed information from the previous layer but also information from their own last state. This enables them to learn from the past.

Those different machine learning algorithms can be used to classify images or preprocessed signals from sensors. The choice of the algorithm and its parameters can be made thanks to the programmer's knowledge, and different setups maybe tested to find the most suitable one.

1.6 Motivation

In this chapter the different types of rolling element bearing have been introduced as well as some bearing defect detection methods. The criticality of rolling element bearing in rotating equipments raises the interest to further investigates the different bearing defect identification methods by conducted a more thorough by conducting and in depth literature review to answer the following motivating question:

Motivating Question 1

What is the current State-of-Art of bearing defect detection technics?

In addition, the recent development of System on Chip have brought more computing power at the edge than ever, theses new capabilities may be more adequate than the current Cloud Based Bearing defect detection technics. Consequently, it is pertinent to explore the advantages of the edge approaches compared to the Cloud based ones in order to find

Motivating Question 2

What is the advantage of using edge machine learning over cloud based approaches for bearing defect detection? To what extent the latest development of systems on chip enable edge defect detection?

The next chapter will present the Background for bearing defect detection as well their current limitations.

CHAPTER 2
BACKGROUND AND LITERATURE REVIEW

In this chapter we conduct a literature review of the current bearing defect detection method, we structure the review by starting by the time domain methods, then the frequency domain approaches and finally the time frequency domain with the wavelet transforms. Finally, some studies using the recent concept of federated learning are presented.

2.1 Current bearing defect detection methods

Bearing condition monitoring methods fall under three main categories. First, vibration-based methods in which a mechanical transducer like an accelerometer or a microphone is used to acquire the data; secondly, Thermographic methods in which hot spots due to abnormal friction in the bearing are identified. A limitation is the expensive thermal sensors cost which prevents a large-scale deployment. Lastly, lubricant analysis methods, as the bearing wears out, the lubricant carries information outside the system in the form of metal particles, and chips [12]. The wear particles from the bearing can be identified using analysis techniques such as ferrography or spectrometry. This last method is also classified as an indirect method as the bearing is not directly analyzed. W. Hoffmann has compared ferrography and spectrometry and their application to condition monitoring of an aircraft engine. It was found that the ferrography permits the detection of debris that are too big to be detected by spectrometry on the oil [33]. However, the complexity of the sensors used and the absence of an onboard system for real- time automated monitoring constitute a substantial limitation to lubricant analysis methods[34]. Moreover, in complex machines, the lubricant circuits can go through multiple bearings, making it harder to identify the faulty element [35] and in very straightforward machines, there is simply no access to the lubricants. Finally, oil analysis methods are more efficient in providing information on the wear rate for gears, but less for bearings [36]. Nonetheless, it appears that vibration methods are much cheaper and enable earlier detection of the bearing defect [36]. Thus, they will be the main focus of this work. Currently, vibration analysis techniques can be divided into four different approaches, depending on the analysis's domain.

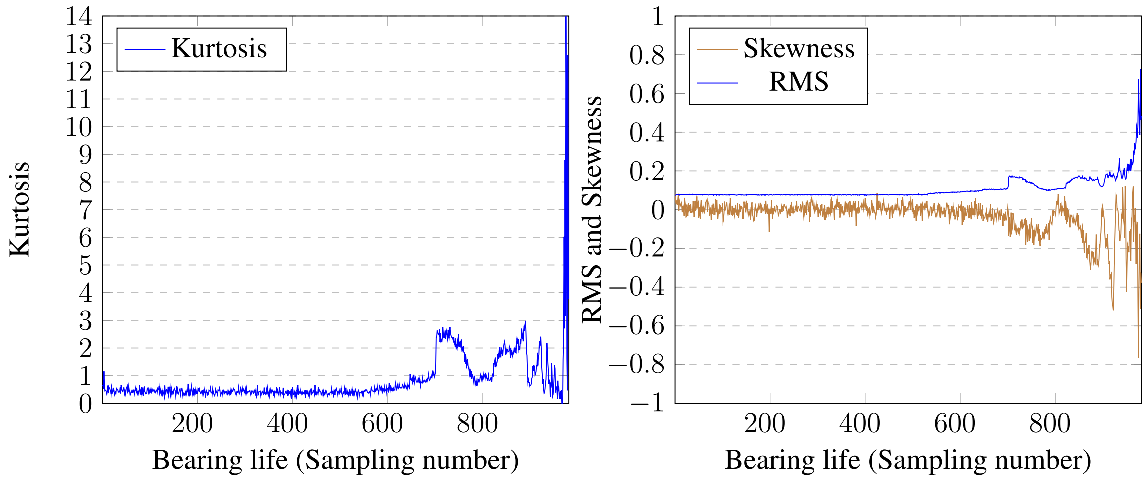
- Time-domain methods where the waveform of the signal is directly used as an input.
- Frequency domain methods in which a Fourier Transform of the signal is processed before identifying defect frequencies.
- Hilbert transform and demodulation technics.
- Time-frequency domain methods, such as the short-time Fourier Transform (SFT) or Discrete (DWT) and Continuous Wavelet Transforms (CWT).

These methods are presented in the following.

2.2 Time domain analysis

These methods directly analyze the waveform of the vibration or acoustic signal generated by the bearing. As the bearing wears out, there is an increase of energy in acoustic emissions (A.E.) and vibration [37]. The most straightforward approach is to use statistical features on the waveform and track the increase of energy in the signal over the bearing's life. Typical features are Root Mean Square (RMS), Skewness (S), and Kurtosis (K), Figure 2.1 presents the evolutions of these three features along the life of a bearing from the NASA Dataset [38]. The Kurtosis evolution of the other test of the Dataset are presented in appendix E. It shows that the Skewness, the Kurtosis, and the Root Means Square signal's values significantly increase at the end of the bearing's life.

$$\mathrm{RMS} = \sqrt{\frac{1}{T_2 - T_1} \cdot \int_{T_1}^{T_2} f(t)^2 \mathrm{d}t},$$
$$\mathrm{S} = \frac{\int_{-\infty}^{+\infty} (x-\mu)^3 p(x) \mathrm{d}x}{\sigma^3},$$
$$\mathrm{K} = \frac{\int_{-\infty}^{+\infty} (x-\mu)^4 p(x) \mathrm{d}x}{\sigma^4}$$



(a) Kurtosis (b) RMS and Skewness

Figure 2.1: Value of Skewness, Kurtosis, and Root Mean Square (RMS) for test 2 bearing 1 of the NASA Bearing Dataset

With:

$x(t)$: the sampling

μ : the mean of $x(t)$

σ : the standard deviation of $x(t)$

Even if these statistical techniques are efficient to monitor a bearing's condition, they require great digital discipline as any issue in data storing, or acquisition may lead to a defect's miss detection. Moreover, some changes in the bearing's mechanical load can also influence the waveform's energy, thus influencing defect detection. Other than those three classical statistical features, some studies have developed custom ones such as Zhang et al. [39], who introduced a dimensionless waveform entropy indicator equation 2.1 derived from the expression of the entropy of random vector.

$$\mathrm{WFE_t} = \frac{1}{M}\sum_{t=0}^{M-1} W_{t-i}\log W_{t-i} \tag{2.1}$$

Where:

WFE_t	: Waveform entropy at time t
W_t	: Waveform at time t
M	: Size of the sliding window

A Long-Short Term Memory Recurrent Neural Network (LSTM-RNN) was then used to classify the bearing condition. However, the approach assumed different degradation states during the training, limiting the trained model's final performance states. More complex time domain-based methods were developed, such as the zero-crossing (Z.C.) method [40]. Using the analogy between signal frequency and the number of crossing between the signal and de x-axis, William and al. detected early defects with an Artificial Neural Network (ANN). Their method was less computationally expensive than a classical FFT method as the computational cost of the FFT $\mathcal{O}\left(n\log(2n)\right)$ was replaced by $\mathcal{O}\left(\frac{Q}{2(N-1)}\right)$ where Q is the number of intervals in the sample.

Apart from applying machine learning techniques, another straightforward tool worth mentioning is to find abnormalities in a time series, using, for example, Shewhart Controls Chart. Shewhart Control Charts help display the evolution of the process over time and the waveform abnormalities' apparition. For the $\bar{X}$, the moving average over m samples is computed. The graph displays the evolution of this moving average. Then, the upper and lower bounds of the domain are defined using $\pm 3\sigma$; if the moving average exceeds these limits, the process likely follows an abnormal evolution. Equation 2.2 details the calculation of one point of the $\bar{x}$ graph.

$$\bar{x} = \frac{\sum_{j=1}^{m} x_i}{m} \tag{2.2}$$

The $\bar{s}$ uses a similar approach with the moving standard deviation of the signal calculated with the equation 2.3.

$$\bar{s} = \frac{\sum_{j=1}^{m} s_i}{m} \tag{2.3}$$

where:

s_i : standard deviation of the i^{th} sample

Those charts can be analyzed in the light of multiple *run-rules*[41]:

Rule 1: a $\pm 3\sigma$ control limit is exceeded

Rule 2: two out of three measurements are in the same $[\pm 3\sigma, \pm 2\sigma]$ warning zone

Rule 3: six consecutive measurements are in increasing or decreasing order.

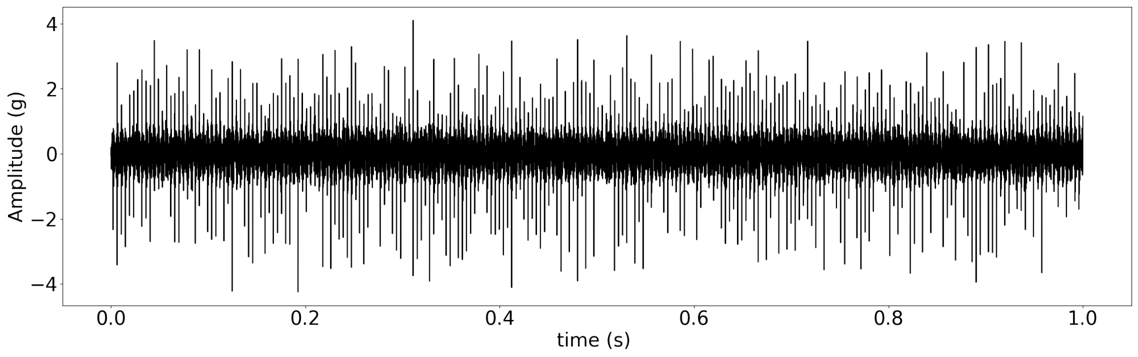
Rule 4: nine consecutive measurements are above or below the mean of the waveform.

These rules are not exclusive and are not backed by a theoretical explanation; one can make other rules that also work like Page[42], who ruled that an abnormality could also be *two consecutive measurements in opposite warning zones*. Finally, a general look at the waveform can also provide valuable information about the observed signal's general type. However, for bearing defect identification, the analysis of the signal in the frequency domain gives valuable information about the defect's type and severity. For instance,

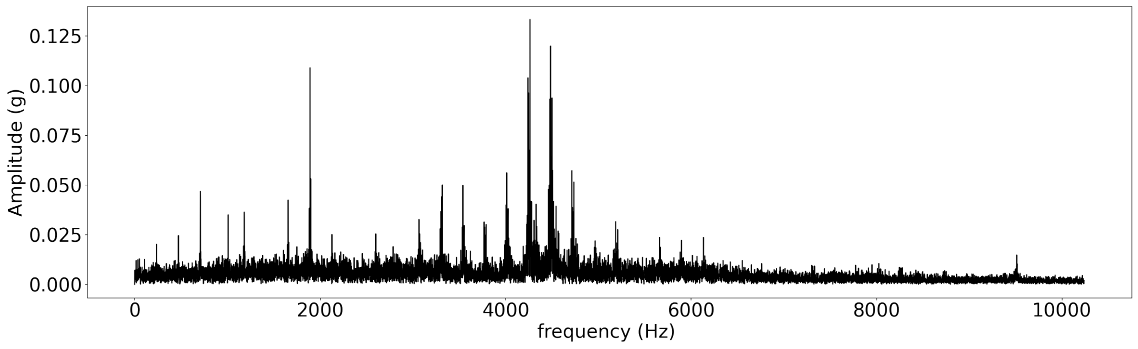
Time-domain methods are usually more straightforward to implement than other methods. However, they do not provide insight into the type of defect growing in the bearing, and they also do not provide information about the severity of the defect. In order to identify the defect, other methods relying on the Frequency domain have been developed.

2.3 Frequency domain analysis

These methods use the Fourier transform to analyze the spectrum of the signal.



(a)



(b)

Figure 2.2: Waveform (2.2a) and Spectrum (2.2b) of a bearing with an Outer Race defect.

2.3.1 Fourier Transforms of a Signal

The Fourier Transform of a T-periodic continuous signal aims to decompose the signal in a basis of sinusoidal functions. It is presented in equation 2.4.

$$g(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} b_k \sin(k\omega_0 t) \tag{2.4}$$

Where:

ω_0 : is the fundamental angular frequency $\omega_0 = \frac{2\pi}{T}$

a_k : obtained by correlation with $g(t)$, $a_k = \frac{T}{2}\int_{\frac{-T}{2}}^{\frac{T}{2}} g(t)\cos(k\omega_0 t)\mathrm{dt}$

b_k : also obtained by correlation with $g(t)$, $b_k = \frac{T}{2}\int_{\frac{-T}{2}}^{\frac{T}{2}} g(t)\sin(k\omega_0 t)\mathrm{dt}$

However, the signals are usually not continuous and not T-periodic. Thus, the Fourier transform is usually used in another form, the Discrete Fourier Transform.

Discrete Fourier Transform of a non-periodic signal and Fast Fourier Transforms: Even if the signal is not periodic, the fact that the acquisition is finite makes it implicitly periodic. As the signal is also discrete, the expression of the Fourier transform in equation 2.4 becomes for a sample $g(n)$ with $n \in [0, n-1]$:

$$G(N) = \frac{1}{N} + \sum_{n=0}^{N-1} g(n)\exp(\frac{-j2\pi kn}{N}) \tag{2.5}$$

Coley and Tukey [43] have developed an algorithm to compute the Discrete Fourier Transform in $\mathcal{O}(N\log N)$ instead of $\mathcal{O}(N^2)$. Known as Fast Fourier Transform, it requires a sampling length of 2^n and is used most of the time instead of straight Discrete Fourier Transform. This transform is used to obtain the spectrum and to detect peaks that correspond to the defects.

2.3.2 Identification of defect frequencies

As the bearing rotates, the presence of a defect will generate impulses in the waveform that will be present in the spectrum. The theoretical frequencies of these pulses are presented in equation 2.6 to 2.9, their derivation is presented in Appendix A:

Ballpass frequency, outer race (BPFO):

$$\mathrm{BPFO} = \frac{nf_r}{2}\left(1 - \frac{d}{D}\cos\Phi\right) \tag{2.6}$$

Ballpass frequency, inner race (BPFI):

$$\mathrm{BPFI} = \frac{nf_r}{2}\left(1 + \frac{d}{D}\cos\Phi\right) \tag{2.7}$$

Fundamental train frequency (FTF):

$$\mathrm{FTF} = \frac{f_r}{2}\left(1 - \frac{d}{D}\cos\Phi\right) \tag{2.8}$$

Ball spin frequency:

$$\mathrm{BSF} = \frac{D}{2d}\left(1 - \left(\frac{d}{D}\cos\Phi\right)^2\right) \tag{2.9}$$

where:

f_r : The rotational speed of the shaft

n : The number of rolling elements

D : The pitch diameter of the bearing

d : The ball diameter

Φ : The contact angle of the rolling element

However, because bearings generally have a cage, all the elements are forced to rotate at the same speed, which will cause a slip. The presence of this slip will cause the low harmonics of the defect to be masked by other components in the spectrum, making the peak detection difficult without filtering [12].

Finally, the defect frequencies' calculations are based on many parameters that can change or be unknown, such as the rotational speed. For example, in paper machines, a section usually contains multiple bearings and is driven by a single belt. Only the speed of the driving wheel is known. Consequently, it becomes complicated to obtain the defect frequencies of the other bearings. Prevost's Master Thesis [44] tries to address this problem by recursively identifying different combinations of peaks and harmonics in the spectrum. This approach has given some results for BPFO but was less successful on other defects, such as BPFI.

As most condition monitoring techniques rely on identifying impulsiveness in the signal, direct peak identification techniques are generally avoided. Techniques that are less sensitive to noise and slip are preferred.

2.4 Cepstrum Analysis

Introduced by Borgert et al. [45]. The cepstrum is obtained by taking the spectrum of a spectrum; this is also equivalent to the inverse Fourier transform of the spectrum's logarithm.

$$C(\tau) = \mathcal{J}^{-1}\left[\log(X(f))\right] \tag{2.10}$$

where: $X(f)$ is the Fourier Transform of $x(t)$, $X(f) = \mathcal{J}\left[x(t)\right] = A(f)\exp j\Phi(f)$ Generally, the power spectrum is used instead of the spectrum, this lets the result to be real and to obtain the *real cepstrum*:

$$C_{xx}(\tau) = \mathcal{J}^{-1}\left[2\ln(A(f))\right] \tag{2.11}$$

The cepstrum analysis is frequently used to detect a defect in gearbox [46], however harmonics and sideband form bearing defect can be missed for advanced defects. Consequently, for bearings, enveloped analysis is often time preferred [46],[47].

2.5 Envelope analysis

The techniques' idea is to use demodulation techniques such as Hilbert transform to obtain information about the signal by amplifying it using the structure's resonance. Indeed, as the vibration signal travels from its source to the sensor, it excites the structure. Thus, the defect harmonic close to the structure characteristic frequencies are amplified [46].

$$Y(f) = H(f) * X(f) \tag{2.12}$$

$X(f)$: the defect spectrum (excitation of the structure)

$H(f)$: the transfer spectrum representing the structure's mechanical properties

$Y(f)$: the response spectrum arriving at the sensor

First, the best frequency range is selected in order to perform demodulation on it. The selection can be made by analyzing the signal-to-noise ratio on the power spectrum for different frequencies. However, Spectral Kurtosis (S.K.) has provided similar results without requiring any reference signal [12]. Indeed, as the S.K. increases with the signal's impulsiveness, it will increase when the signal-to-noise ratio is the most important. That is to say, where the region needs to be selected for demodulation [48]. In his Doctoral Thesis, Sawalhi [49] developed an envelope analysis method that can be semi-automated; the different steps are:

- Order tracking to reduce speed fluctuation.
- Remove the discreet frequencies by using Linear prediction.
- Identify frequency range for demodulation by taking the maximum Spectral Kurtosis.
- Demodulate the signal using the selected band-pass filter.

- User observation for peak identification.

Sawalhi and Randall successfully applied this approach to detect a BPFI on a helicopter gearbox from a test rig at the Defense Science Technology Organization (DTSO) [12][49].

However, this method, even if it requires little user intervention during the first part (no need to determine the demodulation bands manually) or historical data, still requires a user to make the final call on the demodulated signal spectrum to identify the peak. Thus, it is hard to automate this method fully.

Instead of using the time domain or the frequency domain, some approaches take advantage of both domains.

2.6 Time-frequency analysis

The most common Time-Frequency techniques are:

- Short Time Fourier Transform, also known as the Gábor Transform
- Wavelet Transforms

First, an introduction to Time-Frequency domain is given, then, the two above-mentioned techniques are presented.

2.6.1 Time-Frequency domain

Time-domain methods and Frequency domain methods have been discussed previously. The first one enables a good time location of phenomena; however, it provides very little information about the frequencies. The second one provides excellent information about the frequency domain, but all information about the time domain is lost because the Fourier Transform requires an integration over the entire time interval. This duality is often called the Heisenberg uncertainty principle. Any increase in resolution in one domain is a decrease of resolution in the other domain, an analogy with the uncertainty principle between the momentum and the position of a particle in Quantum physics. In order to overcome this, Gábor Dénes introduced the Gábor Transform.

2.6.2 Short Time Fourier Transform (Gábor Transform)

To obtain time resolution in the frequency domain, Gábor [50] introduced a window $w(t-\tau)$ in the Fourier Transform (Equation 2.13) so that the result is localized in time.

$$S(f,\tau) = \int_{-\infty}^{\infty} x(t)w(t-\tau)\exp^{-j2\pi ft}\,\mathrm{dt} \tag{2.13}$$

However, the introduction of this window also reduces the minimum frequency that the transform can represent. Any part of the signal with a period exceeding the time window will be considered constant over time and then lost. It constitutes a significant limitation of the Short-Time Fourier Transform.

2.6.3 Wavelet Transforms

Another approach using wavelets' family to decompose the signal can overcome the short-time Fourier Transform frequency resolution limitation. The advantage of Wavelet compared to other time-frequency transforms is that they are localized in both time and frequency domains. As opposed to the Fourier transform, Wavelet transforms can more easily represent abrupt changes in the signal, such as an impact from a rolling element on a bearing defect. A wavelet is a rapidly decreasing oscillations with 0 means; they only exist for a given time to be localized. Many functions satisfy the 0 means and localization properties; the choice of such a function will depend on the application. Contrary to the Fourier transform, in which the basis of decomposition is a sinusoidal function, the Wavelet Transform uses multiple wavelets resulting from the translation and dilatation of a mother wavelet: $\Psi_{a,b}(t)$. By tuning the parameters a and b, the wavelets can be moved in the time domain, and its frequency resolution can be adapted; Figure 2.3 from [51]

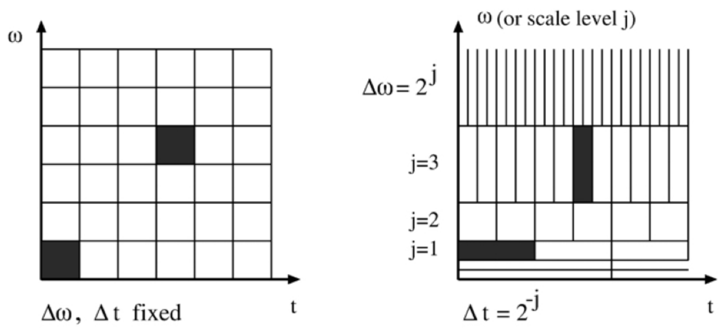


Figure 2.3: Time frequency resolution of the Short Time Fourier Transform (left) and the Wavelet Transform (right).

presents the difference between the Short-Time Fourier Transform, and the wavelet transforms in terms of time-frequency resolution. Each *box* represents one window of analysis: contrary to the Short Time Fourier Transform, the Wavelet Transform frequency resolution can be adapted.

There are two types of wavelets, Continuous wavelets, and Discrete wavelets. The main difference between the two is how the coefficients a and b are changed. The Discrete Wavelet Transform only takes integer values contrary to the continuous form where it can take non-integer values; this is often time preferred as the computational cost is dramatically reduced.

Continuous Wavelets give a higher resolution in time and frequency, which captures the oscillations of the signal with more accuracy than for Discrete Wavelet Transforms. The discrete Wavelet Transform usually uses powers of 2 coefficients for shifting and scaling; this enables an analogy with changes by octaves in the signal's frequency. Mallat [52] developed a fast pyramid algorithm comparable to a filter bank that can be use for fast processing of wavelet transforms. The filter bank is composed of high-pass and low-pass filters (as presented in Figure 2.4).

The high-pass sub-band (H in Figure 2.4) captures the high pass sub-band of the signal (i.e., the details) and the low-pass sub-band (G in Figure 2.4), the low frequencies, i.e., the signal's trend. At each stage of the filter bank, the high pass filter extracts the details for

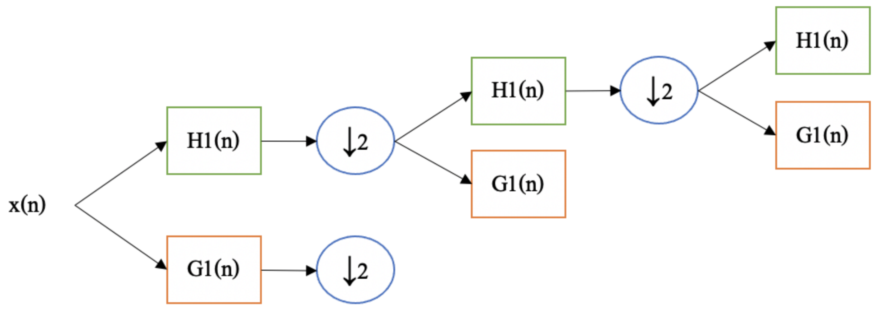


Figure 2.4: Schematic of a filter bank (H, low-pass and G, Hight-pass)

this level, and the low pass filter gets the trends that will be fed into the next level (another set of high-pass and low pass filters) of decomposition with different filters parameters.

Wavelet Transforms: As mentioned previously, a wavelet derives from a mother wavelet. This function only needs to satisfy one condition called admissibility condition 2.14.

$$\int_{-\infty}^{\infty} \Psi_{a,b}(t)\mathrm{dt} = 0 \tag{2.14}$$

As long as this condition is satisfied, we can define a wavelet transform 2.15, where a and b are the dilations, and the translation factor is used to tweak the mother wavelet to generate the rest of the basis.

$$W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\Psi^{*}\left(\frac{t-b}{a}\right)\mathrm{dt} \tag{2.15}$$

Shifting of a Wavelet: To shift a wavelet $\Phi(t)$, we can offset the time by a constant factor b to obtain $\Phi(t-b)$ This will cause a translation of the Wavelet in the time domain.

Scaling of a Wavelet: The scaling of a Wavelet $\Psi(t)$ can be expressed by adding a factor $\frac{1}{a}$ in the Wavelet definition in order to compress the time:

- $\Psi(\frac{t}{a})$ with $s > 1$ will reduce the frequency of the wavelet causing a dilation of its representation.

- $\Psi(\frac{t}{a})$ with $s < 1$ will increase the frequency of the wavelet causing a shrinking of its representation.

A shrunk Wavelet will be more localized and will consequently capture rapid change in the signal, whereas a dilated wavelet will help express the signal's slow evolution.

Common wavelet families Many mother wavelets can be defined, and the selection of the wavelet is based on the application. The first wavelet introduced were the Haar wavelets 2.5. Other famous wavelets such as the Daubechies or Morlet, are worth mentioning. The former constitute an orthogonal basis and are very useful for analysis/synthesis application (e.g.: image processing) and the later are not orthogonal but enable some useful overlap for visual applications [12].

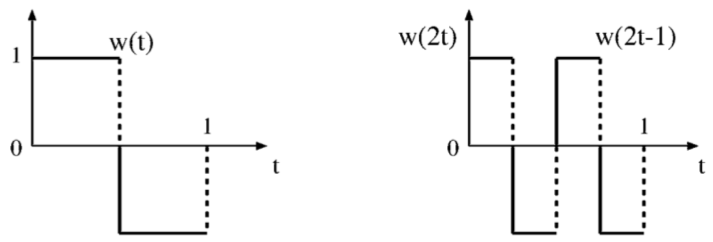


Figure 2.5: The Haar wavelet

From Wavelet to filter banks: an introduction As afford mentioned, a wavelet family can be used to decompose a function $f(t)$ in the form:

$$f(t) = \sum_{j,k} b_{j,k} w_{j,k}(t) \tag{2.16}$$

This decomposition is easier when the wavelet families are orthogonal; that is, they satisfy 2.17

$$\langle w_{i,j}, w_{k,l} \rangle = \int_{-\infty}^{\infty} w_{i,j}(t) w_{k,l}(t) \mathrm{dt} = \delta^{i,k} \delta^{j,l} \tag{2.17}$$

The members of the family are described by two different indices as they depend on the time location of the wavelet: Shifting, first index. As well as, the shrinking (Scaling) of the wavelet, represented by the second index.

For instance, $w_{0,k}(t)$ will be the k^{th} shift of the wavelet $w_{0,0}(t) = w(t)$, the mother wavelet; and $w_{i,0}(t)$ will start at 0 but have a different scale than $w(t)$.

An easy way to represent a wavelet from the mother wavelet is to write as follows:

$$w_{i,j}(t) = w(2^i \cdot t - j) \tag{2.18}$$

The equation 2.18 shows that the increase of i will $2^i \cdot t$ for a given t. Consequently, for each increment of $\mathrm{t} \to \mathrm{t} + \mathrm{dt}$ we would obtain $2^i(\mathrm{t}+\mathrm{dt})$ thus an increase of i compresses the wavelet. Similarly, an increase in j would translate the argument $(2^i \cdot t - j)$ in the expression of equation 2.18, which achieves the shifting of the wavelet.

A key idea behind wavelet decomposition in relation with filter bank is *multiresolution*: If we consider an orchestra playing (or a bearing vibrating), there will be high and low frequency components played at different times in the song. The idea is to approximate the signal as a sum of:

$$\text{Signal} = \text{Signal levels} \oplus \text{Signal details} \tag{2.19}$$

This can also be seen as a sum of:

$$\text{Signal} = \text{Local averages} \oplus \text{Local differences} \tag{2.20}$$

For instance, considering the impulse signal $x = (....0\ 0\ 1\ 0\ 0....)$ and the two simple high-pass and low-pass filters:

$$H_p :\ y(n) = \frac{1}{2}\left[x(n) - x(n+1)\right] \tag{2.21a}$$

$$L_p :\ y(n) = \frac{1}{2}\left[x(n) + x(n+1)\right] \tag{2.21b}$$

Note: The H_p and L_p filter uses indices $x(n)$ and $x(n-1)$ because at discrete time n, the value of $x(n+1)$ is unknown as it has not happened yet.

Applying our filter to the above considered impulse signal provides the impulse response of the filters. For the Low-pass filter:

$$Y_{Lp} = (...\ 0\ 0\ \frac{1}{2}\ \frac{1}{2}\ 0...) \tag{2.22}$$

And for the Hight-pass filter:

$$Y_{Hp} = (...0\ 0\ \frac{1}{2}\ \frac{-1}{2}\ 0...) \tag{2.23}$$

The two previous response highlight the fact that we can reconstruct the initial signal using the response of the $\mathrm{H_p}$ and $\mathrm{L_p}$ filers:

$$x = y_{Hp} + y_{Lp} \tag{2.24}$$

y_{Lp} giving the moving average of the signal and y_{Hp} giving the moving difference, i.e. the details. In order to improve the two filter it is usually to normalize their response by adding a factor $\sqrt{2}$, this will change the Finite Impulse Response (FIR) of the filter to be:

$$Y_{Lp} = (...\ 0\ 0\ \frac{1}{\sqrt{2}}\ \frac{1}{\sqrt{2}}\ 0...) \tag{2.25a}$$

$$Y_{Hp} = (...\ 0\ 0\ \frac{1}{\sqrt{2}}\ \frac{-1}{\sqrt{2}}\ 0...) \tag{2.25b}$$

In the following, the coefficients of the Lp and Hp filter will be denoted as $\mathrm{c(n)}$ and $\mathrm{d(n)}$

respectively, such that:

$$\mathrm{c}(0) = \mathrm{c}(1) = \frac{1}{\sqrt{2}} \tag{2.26a}$$

$$\mathrm{d}(0) = -\mathrm{d}(1) = \frac{1}{\sqrt{2}} \tag{2.26b}$$

Now in the continuous time we have generic expression for the scalling function that include the $\mathrm{H_p}$ filter:

$$\Phi(t) = \sqrt(2) \sum_{k=0}^{N} c(k)\Phi(2t-k) \tag{2.27}$$

If we express this using the $\mathrm{c(n)}$ we obtain:

$$\Phi(t) = \sqrt(2)\left(\frac{1}{\sqrt{2}}\Phi(2t) + \frac{1}{\sqrt{2}}\Phi(2t-1)\right) \tag{2.28}$$

Equation 2.28 can be further simplified to obtain the dilatation equation, for our is simple case :

$$\Phi(t) = \Phi(2t) + Phi(2t-1) \tag{2.29}$$

Figure 2.6 from [53], presents the solution to the dilation equation for the considered filters.

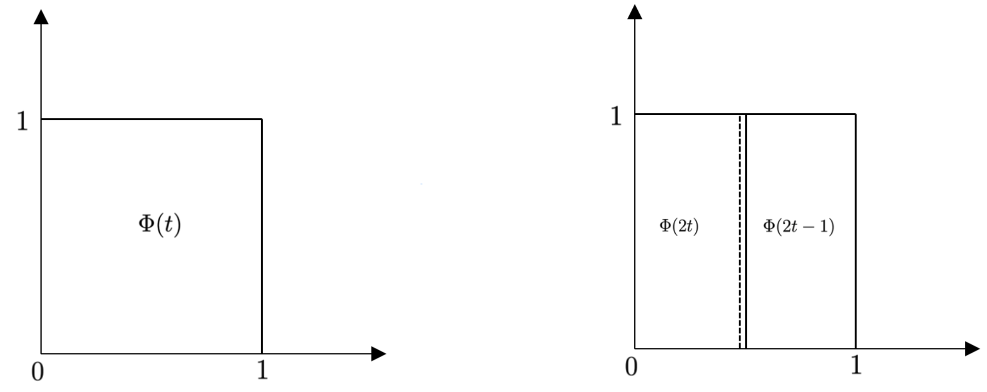


Figure 2.6: Solution to 2.29

Similarly, the generic wavelet equation is given by:

$$w(t) = 2\sum_{k=0}^{N} d(k)\Phi(2t-k) \tag{2.30}$$

and, injecting the $\mathrm{d(k)}$ it simplifies into:

$$w(t) = \Phi(2t) - \Phi(2t-1) \tag{2.31}$$

The two simple $\mathrm{H_p}$ and $\mathrm{L_p}$ filters are actually very specific as they helped construct the Haar Wavelet presented in Figure 2.5.

The idea of decomposing a signal into its difference and its average can be applied more than once so that we can decompose the average again to obtain more details, by applying the Low-pass and High-pass filter in the manner described in Figure 2.4, we find a more refined description of our signal.

2.7 Wavelet Transform for bearing defect detection

Wavelet analysis is among the best time-frequency techniques. Due to their use in image and signal processing, wavelet transform algorithm and their implementation were heavily studied and are very efficient[54]. They are also very appropriate for signals dominated by impulse responses at different frequencies [49], Sawalhi introduced Morlet wavelets to estimate the most important spectral Kurtosis to select a demodulation frequency band (see section 2.5). Garzon et al. [55] followed the same process by using the Kurtosis and the RMS to select the frequency band.

In using Discrete Wavelet transform, Kumar et al. [56] identified the Symlet 5th order to detect minimal outrace defect. However, they have only applied this technique to visually identified defects in the signal, which does not provide a solution for automated classification of the bearing condition. Some work has been carried out by Holm-Hansen

[57] to derive a customized filter bank adapted to bearing defects. The identified wavelet was compared with the Daubechies 4th for different loads on the bearing shaft. Kankar et al. [58] have used wavelet transform to identify features fed into three different machine learning algorithms: Support Vector Machine, learning vector quantization, and self-organizing maps. The two first ones being supervised learning techniques, and the last one being unsupervised. In order to select a wavelet transform, they have compared the Shanon Entropy of the wavelet coefficient and chose the Complex Morlet family.

Kumar et al. [56] have used Continuous Wavelet transform associated with Deep Convolutional Neural Networks to achieve a 100% accuracy on multiple types of bearing defects. However, this study only considered artificially created defects. Paya et al. [59] used the Daubechies 4 and an Artificial Neural Network to detect bearings and gears defects in various conditions. They achieved 96% average classification rates. As in [56] the default was only simulated. Jose et al. [60] used the Daubechies 5th in 3 types of Neural Networks: A Multi-Layer Perceptron, a Radial Basis Function Neural Networks, and a Probabilistic Neural Network. The 3 Networks were evaluated on different faults and claimed to give state-of-the-art results with 70% accuracy.

For filtering purposes, wavelet techniques were also used on the result of an Ensemble Empirical Mode Decomposition by applying to the maximum kurtosis Intrinsic Mode Function a Tunable Q-factor Discrete wavelet was able to denoise the signal and to diagnose an outer-race defect earlier than with classical methods[61]. Wang et al. [62] applied two discrete wavelet transforms in parallel with different filters; introduced this dual-tree approach [63] and clarified by [64]. The technique successfully identified a defect in both the inner and outer race of a bearing and is said to be suitable for online parallel implementation. Zang et al. [65] used a similar approach with the introduction of a Q-Factor Gabor Wavelet Transform applied to the signal to identify the presence of the defect frequencies; their methodology is presented in figure 2.7[65].

Work has been done on adaptive natural Morlet Wavelet 2.32 to detect gearbox de-

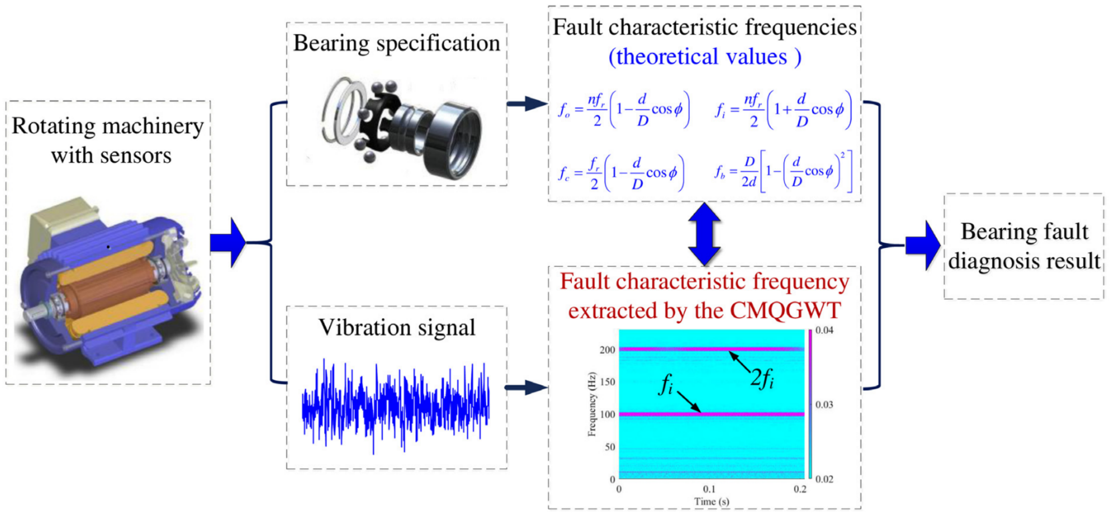

Rotating machinery with sensors
Bearing specification
Vibration signal
Fault characteristic frequencies (theoretical values)
$f_o = \frac{nf_r}{2}\left(1-\frac{d}{D}\cos\phi\right)$
$f_i = \frac{nf_r}{2}\left(1+\frac{d}{D}\cos\phi\right)$
$f_c = \frac{f_r}{2}\left(1-\frac{d}{D}\cos\phi\right)$
$f_b = \frac{D}{2d}\left[1-\left(\frac{d}{D}\cos\phi\right)^2\right]$
Fault characteristic frequency extracted by the CMQGWT
f_i
$2f_i$
Frequency (Hz)
Time (s)
Bearing fault diagnosis result

Figure 2.7: The Continuous Multiple Q-factor Gabor Wavelet Transform approach

fects. The Morlet Wavelet is adapted by tweaking the parameter β; the wavelet transform is applied for different values of β, the one with the maximum Kurtosis is then selected. This method detected early cracks in a gearbox's teeth but has not yet been applied to bearings.

$$\Psi_{a,b}(t) = \exp\left[-\frac{\beta^2(t-b)^2}{2a^2}\right]\cos\left[\frac{\pi(t-b)}{a}\right] \tag{2.32}$$

Some have also applied wavelet combined with machine learning techniques rather than a filter to detect theoretical defect frequencies. Zhang et al. [66] trained a convolutional neural network on images generate by Short-Time Fourier Transform of bearings defect compared their method with classical deep learning method and obtained better results; moreover, the training time was reduced as their approach enables the use of a small image. Wavelets were also used in image processing of bearing defect by Islam et al. [67] who processed the acoustic emission of a defective bearing with wavelet packets to training a convolutional neural network. For this, they made use of an NVIDIA GeForce 580 GPU that would be very expensive to deploy at the edge. He et al. [68] also used the wavelet to represent the signal in the time-frequency domain; the bearing defect frequency was directly extracted from this representation. Haidong et al. [69] used

Gaussian wavelet in an autoencoder to avoid the feature selection and reduce the induced bias that this step introduces in the model. Compared to an auto-encoder and a support vector machine model trained with feature selection, the wavelet auto-encoder enabled better detection of electric locomotive bearings. Gelman et al. [70] also used wavelet as a new representation of bearing's condition and used wavelet bi-coherence feature to characterize if the bearing is defective. This approach suffers from the arbitrary choice of the parameters in the features. Liu et al. used the Mexican hat wavelet as the kernel function of a support vector machine algorithm [71].

Wavelets were introduced at the edge by Borova et al. [72] to compress measurements of CO_2 and Temperature. The compressed data were then sent to a front-end device where they were to be decompressed. The study focused on multiple families of wavelets such as Haar, Meyer, Simlet, and bi-orthogonal.

To summarize, the use of wavelets for condition monitoring could be classified in 3 categories that serve:

- To identify the maximum spectral Kurtosis for envelope analysis [49, 12].
- As filter to denoise the signal and then apply classical frequency domain techniques [62, 61, 65].
- To generate new representation of the data (images, 3 dimensions spaces...)[70, 67, 71], which is the same idea as [66, 12].

However, in these two first cases, the results are denoised spectra, so human intervention is required to evaluate the bearing's status, making it challenging to automatize.

Moreover, when wavelets are used to generate a new image representation and to apply machine learning, they are generally processed in the Cloud, which results in a high volume of data sent transmitted. Recently a new form of Machine Learning called Federated Learning was introduced by Google. By using the idling processing power this approach try to reduce the amount of data transmitter to the cloud, it is presented more into detail in the next section.

2.8 Federated Learning

The increased performance of Edge devices has enabled the execution of machine learning models along with their training at the edge. For instance, a fleet of edge devices using the same machine learning model, can retrain the model with the data from those devices. Federated learning refers to the process of locally retrain the model on the edge devices and merging the new model parameters before propagating into the entire fleet of devices.

2.8.1 Concept behind federated learning

The main idea of federated learning is to use the idling processing power at the edge. From an edge device perspective, it can create its own local dataset. For example, a next word prediction local training set can be created on a smartphone using the user's actual next word to validate if the prediction was correct. The smartphone can then use this local dataset to retrain its model and obtain more optimized parameters. From the perspective of the global performance of the entire fleet, the centralized server requests a representative subset of the device to retrain their models when they are idling and send back the new parameters. The set of new parameters can be merged using federated averaging (Cf.: section 2.8.3) and sent back to all the edge devices to improve the global performance of the fleet. The process can be repeated with another subject of devices.

2.8.2 Advantages of Federated Learning

Google introduced federated learning to solve data privacy concerns according to a White House's report on minimization of data collection [73]. Indeed, retraining a federated learning model does not require sharing the data with a central server. For smartphone

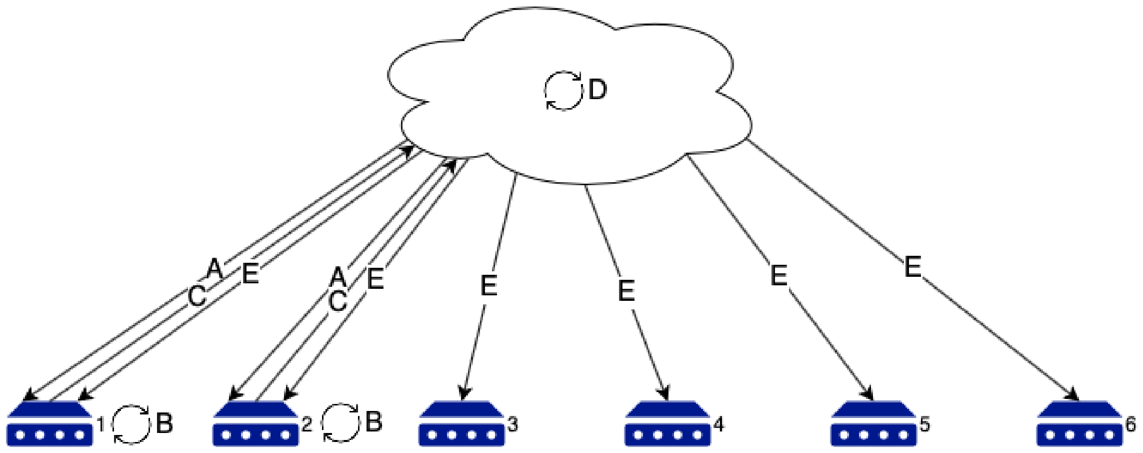


Figure 2.8: The Federated Learning Process.

users, it becomes essential that the data stays on their devices. Even the centralized server controls the training, it never accesses the data which significantly reduces the risk of data breach as the device will never share the data.

Moreover, smartphone chips (and other microprocessors' chips) have become extremely powerful and, contrary to centralized servers, they are available when the owner is not using his phone (e.g., at night when charging and connected to WI-FI). Leveraging this computing power permits reducing the load on a centralized server and presents significant advantages from an ecological perspective. Centralized servers produce much heat that lessens the server's performance and necessitates a cooling system.

Some projects have put the server's room heat to used like water heating for energy production or other ways to recover the heat energy like the one discussed in [74]. Alternatively, the cooling of the server could be accomplished by submerging the servers in the Coastal area near a big city like Microsoft's Natick project [75].

Nevertheless, federated learning to reduce the load on the server is a simple and efficient solution that can also be combined with the ideas mentioned earlier to reduce the environmental impact of data center and Industry 4.0 solutions.

2.8.3 Theory

A typical machine learning problem can be summarized into the minimization of the error f. To do so, we associate a cost, the "loss" to the model performance. Consider a dataset $\mathcal{D} \triangleq \{(\underline{x}_1, y_1)...(\underline{x}_N, y_N)\}$, where $\underline{x}_i$ is a vector of features and y_i is the corresponding label. We can denote the loss for the sample $i \in [\![1, N]\!]$ as $l(\underline{x}_i; y_i; \underline{w})$ a function that depends on the samples and $\underline{w} \in \mathbb{R}^d$ the current model parameters.

In centralized learning, the objective is to minimize the overall cost for all the samples of the dataset with respect to the dataset parameters, that is:

$$\min_{w \in \mathbb{R}^d} \left(\frac{1}{N} \sum_{k=1}^{N} l(\underline{x}_k; y_k; \underline{w}) = \min_{w \in \mathbb{R}^d} \right) \tag{2.33}$$

In federated learning, the approach is different. The central server does not have access to the dataset of each client.

Each client, has its own non-iid dataset which will be denoted $\left(\underline{x}_i^k, y_i^k\right)$ for client k with $k \in [\![1, K]\!]$. The optimization problem can then be rewritten by considering the loss for each client as in [76]:

$$F_k(w) = \frac{1}{n_k} \sum_{i=1}^{n_k} l(\underline{x}_i; y_i; \underline{w}) \tag{2.34}$$

where n_k is the size of the dataset of the k^{th} client. Then, the loss across clients is defined as:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{2.35}$$

a weighted average of the loss for each client.

In [76], the federated learning algorithm is presented, this algorithm is at the basis of federated learning as it enable a good global learning while maintaining low communication cost.

As opposed to centralized training, where high communication speed is achieved between the different processing units; federated learning communication cost are very important as the communication is often achieved over lower bandwidth networks.

The federated learning algorithm from [76] is presented in algorithm 1.

Algorithm 1 FedereatedAveraging from [76]

```
Server executes:
  initialize $w_0$
  for each round t=1,2,... do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ ((random set of $m$ clients))
    for each client $k \in \mathcal{S}_t$ do                ▷ In parallel across clients
      $w_{t+1}^k \leftarrow \mathrm{ClientUpdates}(k, w_t)$
    end for
    $w_{t+1}^k \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for

Clients Updates (k,w):                                       ▷ Run on client k
  $\mathcal{B} \leftarrow$ (Split $\mathcal{P}_k$ into batches of size B)
  for each local epoch $i$ from 1 to E do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \nu \nabla l(w; b)$
    end for
  end for
  return to server
```

2.8.4 Application of Federated Learning

In the applications of federated learning, two categories and two different setup of federated learning should be introduced:

Federated learning setups

The federated learning setup refers to the scale at which the “federation” occurs. Google describes it either as cross-silo or cross-devices federated learning as presented in 2.8.1.

In the former, the data is kept local to the devices, for example, a fleet of smartphones in which each device has its own dataset illustrates such an application. This idea as been used in [77] to defect an out-of-Vocabulary Words. In cross-silo, the data needs to be kept inside an organization, but it can be shared across different devices. For example, manufacturing plants in which there is no need to share the data across the facility and a local cloud can be used to retrain the model based on each facility's dataset. As a practical example, [78] where multiple hospitals joined an effort to develop a model for breast density classification without sharing their data. Other sectors such as the Agriculture and food sector also have an interest in cross-silo federated learning. Durrant et al. [79] used federated learning as a way to leverage the limited amount of data collected per silo while maintaining data privacy.

2.9 Federated Learning in Manufacturing

The field of applications for federated learning to manufacturing is vast, especially in the 4th industrial revolution and the Internet Of Things development. More than ever, machines are connected to the internet and generate large amounts of data.

Federated Learning for Manufacturing enables more data privacy, and uses of lower data bandwidth which are the main challenges for the Internet of Things for Manufacturing [80].

Nandury et al. [81] have applied cross-silo federated learning for automated speech recognition (ASR). They have introduced a new Federated Learning Averaging algorithm (FedAvg-DS) that gives better results in ASR and its application to a personal assistant such as Amazon Alexa.

Kanagavelu et al. [82] have developed a platform for horizontal, cross-silo federated learning integrated into the Industrial Internet of Things for manufacturing. They compared the computation times for two settings: a Two-phase Multip-Party computation and a Peer-to-Peer setting. Hiessl et al. [83] have identified that contrary to other fields of application, federated learning for manufacturing is limited by the large variety of assets. As opposed to the example of a fleet of smartphones, there is a large variety of machine types and industrial operations in manufacturing. Youyang et al. [84] have proposed a higher integration of federated learning with blockchain to increase the framework's resistance to poisoning attacks in an IIoT setup. Ge et al. [85] have investigated the use of vertical federated learning versus centralized learning on a Bosh dataset. In particular, they have tested two types of machine learning algorithms: random forest and support vector machine. It was shown that Vertical Federated Learning could achieve an accuracy similar to centralized learning. Savazzi et al. [86] have demonstrated that Federated Learning for automation enables low communication times in a domain of application where Ultra-Low Latency is required. The addition of federated learning in this context is shown to be promising for application in collaborative automation. Dhada et al. have shown in [87] the use of federated learning for Remaining Useful Life (RUL) prediction on a turbofan fleet. The use of Federated Learning in this context enables overcoming the highly distributed nature of the data. The federated trained Recurrent Neural Network (RNN) showed "pictorially" [87] that it predicts a similar trend to the actual values. However, the Centralize learning model gives much better results in the study. Zhang et al.[88] have tested Deep Neural Network in a horizontal federated learning layout. Two different organizations of data have been tested, an IID and a non-IID. In the case of the non-IID data, the Federated Learning method has a significant gap with the centralized techniques.

The previous studies emphasize that Federated Learning applied to manufacture primarily in asset condition monitoring shows promising results. However, no studies obtained satisfying results for federated learning for bearing defect detection.

The following chapter will present a summary of the limitations of the current approaches to developing research questions.

CHAPTER 3
PROBLEM FORMULATION

In this chapter we present the limitation of the current bearing defect detection approaches, then the research questions and research hypothesis are introduced.

3.1 Summary and Limitation of the current approaches

3.1.1 Time domain limitations

As mentioned above, time domain analyses are applied to track the trend of the vibration signal from a statistical viewpoint, which does not provide a good insight about the type of defect nor its severity.

Moreover, apart from the semi-automated method proposed by Sawalhi in his PhD Thesis[49], time domain method relies on a user decision-making. Consequently, it is difficult to imagine deploying them on a large scale.

3.1.2 Frequency domain limitations

Defect frequency identification methods are among the most developed; they rely on multiple denoising techniques that aim to increase the impulsiveness of the signal to identify theoretical defect frequencies. However, this requires knowledge of the rotation speed of the bearing which is unknown for some system. In paper machines for instance, multiple rollers are driven by a belt, but only the driving roller is equipped with a tachometer; thus, any slip between the belt and the wheels will affect the actual defect frequency. The difference between the theoretical and actual defect frequency will make any automation attempt very difficult [44].

Other frequency methods like Cepstrum analysis or envelope analysis also rely on peak identification, which requires an action from an operator to make the classification.

3.1.3 Time-frequency possible improvements

These methods try to acquire information in both time and frequency domain. The Wavelet transform is often time preferred to the Short Time Fourier Transform as it is suffering less from the Heisenberg uncertainty principle. Their applications have mostly been for denoising or as a tool in envelope analysis. Some studies have shown promising results by combination of wavelets and machine learning techniques, which was carried out in the cloud.

As mentioned earlier, even though this is interesting from an automation viewpoint, it raises a new issue: the transmission of this data to the cloud is not efficient as it is likely to be stored and never processed.

3.2 Research questions

Regarding the above discussion, 2 major gaps can be identified:

1. Current bearing defect detection methods are not adapted to large scale automated implementation as they require user intervention.

2. Techniques using a Cloud based machine learning approach result in a high volume of data transmitted, and low information density data stored in the Cloud.

Thus, the proposed work aims to fill two gaps by answering the following Research Question (RQ):

Research Question

How can a new approach for automated bearing defect detection be decoupled from Cloud computing?

This main question could be divided into three main steps:

Research Question 1

How can bearing defect detection be automated without previous knowledge of their configuration?

To answer this question we can formulate the following hypothesis.

Hypothesis 1

Wavelet transforms, associated with machine learning, enable automated bearing defect detection with same performance as classical methods without previous knowledge of the bearing configuration.

Hypothesis 2

The wavelet transform step can be merged into the learning approach in order to significantly improve the process. This can be achieved not only at the device level by merging the algorithms, but also at the network level by implementing a federated learning approach between multiple devices.

Then, in light of the cloud limitation expressed above, we can formulate the following question:

Research Question 2

How can this approach be decoupled from cloud computing while maintaining adequate sampling rate and sensitivity for predictive maintenance applications?

The above question can be addressed with the following hypothesis:

Hypothesis 3

By leveraging the recent developments on microprocessor and System on Chip, a parallel implementation of the above-mentioned techniques can be developed at the edge.

Hypothesis 4

The parallel wavelet implementation at the edge enables an early detection of the bearing defect that is suitable for predictive maintenance planning.

Finally, the proposed method should be benchmarked against current techniques.

Research Question 3

What is the trade-off between cost and performance (speed, sampling rate and sensitivity) comparing the edge implementation and the cloud implementation?

Hypothesis 5

The edge implementation will have comparable or better results than cloud computing services at a lower cost.

In this chapter, the previous existing techniques for bearing defect detection and their limitation were presented. A research question was identified and hypothesis made to answer this question. The following chapter is going to present the approach that will be followed to validate or reject them.

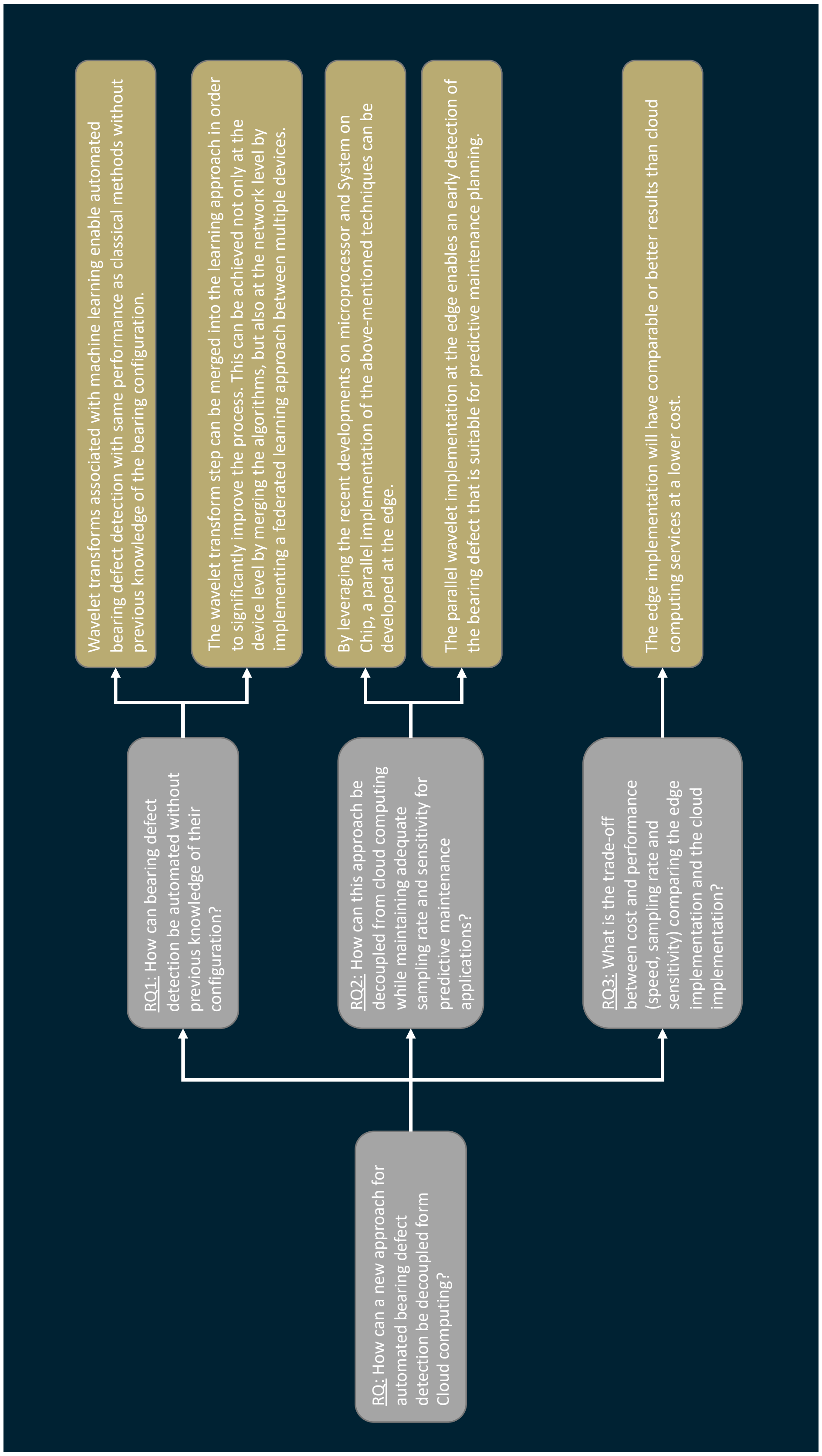

RQ: How can a new approach for automated bearing defect detection be decoupled form Cloud computing?
RQ1: How can bearing defect detection be automated without previous knowledge of their configuration?
RQ2: How can this approach be decoupled from cloud computing while maintaining adequate sampling rate and sensitivity for predictive maintenance applications?
RQ3: What is the trade-off between cost and performance (speed, sampling rate and sensitivity) comparing the edge implementation and the cloud implementation?
Wavelet transforms associated with machine learning enable automated bearing defect detection with same performance as classical methods without previous knowledge of the bearing configuration.
The wavelet transform step can be merged into the learning approach in order to significantly improve the process. This can be achieved not only at the device level by merging the algorithms, but also at the network level by implementing a federated learning approach between multiple devices.
By leveraging the recent developments on microprocessor and System on Chip, a parallel implementation of the above-mentioned techniques can be developed at the edge.
The parallel wavelet implementation at the edge enables an early detection of the bearing defect that is suitable for predictive maintenance planning.
The edge implementation will have comparable or better results than cloud computing services at a lower cost.

Figure 3.1: Research Questions

CHAPTER 4
FEDERATED EDGE IDENTIFICATION OF BEARING DEFECTS

In this chapter, the implementation of the Edge computing architecture is presented. First, the System on Chip selected for this work is presented. Then we detail the development of the data acquisition followed by the Wavelet Transform processing. Finally, we explain the Edge inference of bearing status and federated learning in System On Chip's (SoC) network.

4.1 BeagleBone AI presentation

The BeagleBone AI is the latest board release by the BeagleBone Bone foundation. As opposed to the popular Raspberry Pi [89], the BeagleBoard Foundation release Open Source designs [90]. The board is based on the Texas Instrument Sitara AM5729 chip, which provides enough computing power to places it between the small SoC and powerful industrial computers.

4.1.1 Texas Instrument Sitara AM5729 chip presentation

Texas Instruments developed the AM572x family to offer powerful processing for embedded applications. These ARM-based chips features, among others, a dual Cortex A15 RISC processor and two Digital Signal Processors (DSP). As presented in figure 4.1, the ARM architecture's main advantage is to enable access to the different port from all the processing units.

Moreover, it permits the separation of computing tasks from the operating system execution; this is a clear advantage over other boards like microcontrollers, which are limited to a single instruction loop.

Texas Instruments integrated two Programmable Real-Time Unit Subsystems, and

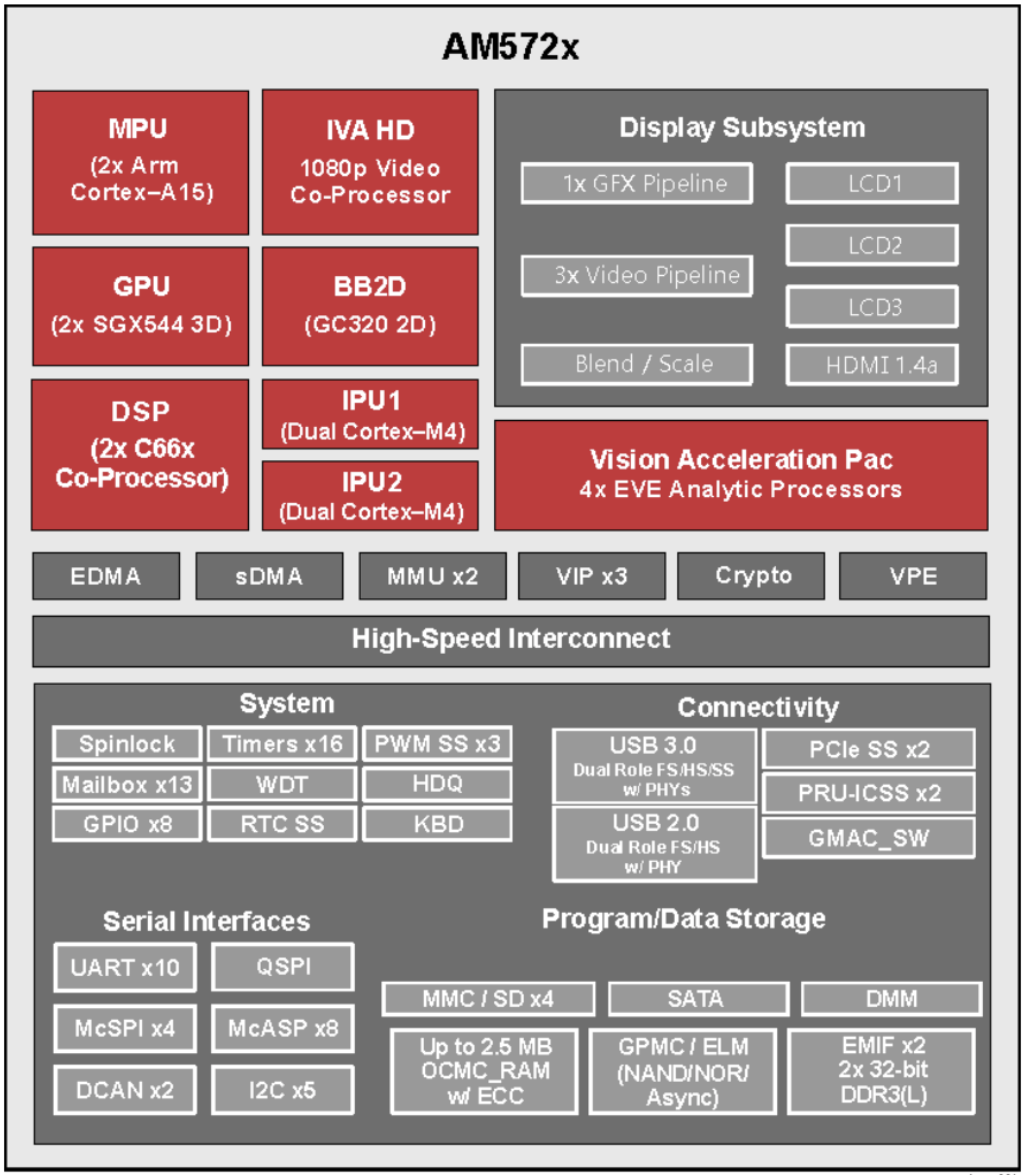


Figure 4.1: AM5729 block diagram from [91].

Industrial Communication Subsystem (PRU-ICSS), which can be is used for deterministic data acquisition separately from the other control functions.

The AM572x family is composed of three different chips: AM5726, AM5728, and AM5728. The last one featuring two Embedded Vision Engines (EVE) that are not present in the two others. Finally, all the AM572X chips have a GPU expect the AM5726. To facilitate the use of this new onboard processing power, Texas Instrument released the Texas Instrument Deep Learning (TIDL) library to show some test cases of online inferences using the DSP and the EVE.

The Texas Instrument Chips are integrated into different development platforms, the BeagleBone AI is the cheapest at $125 [92]. In the next section we will present the BeagleBone AI development platform.

4.1.2 BeagleBone System on Chip

The BeagleBoard Foundation releases open-source SoC based on the Texas Instruments chips. Their flagship board, the BeagleBone Black, sells for $82 [93]. In 2019, the foundation released the BeagleBone AI. As mentioned earlier, it targets more computing-intensive tasks. To expand the BeagleBone AI application, the board features a Touch Screen controller with a 12bits 3.3V Analog to Digital Converter that can be accessed from the PRU-ICSS via the I2C Bus. The BeagleBone AI has the same form-factor and headers as the BeagleBone Black. The board's low cost and robust capabilities make it a competent development platform for IoT applications. In the next section, we present the proposed solution's architecture at the SoC level.

Proposed Architecture at SoC level.

In order to implement bearing defect identification at the edge, it is proposed:

- To use the PRU-ICSS for deterministic data acquisition. Using the I2C Bus, the PRU-ICSS can access the ADC and retrieve measurements from an analog accelerometer.
- The ARM Cortex will be used for communication with other boards in the network and will process the Wavelet Transform of the vibration data acquired with the PRU

- The DSP and EVE engine will execute the Machine Learning model compiled with the Texas Instrument Deep Learning library.

First test on the BeagleBone AI

Before selecting the BeagleBone AI as the board for this project, the different examples proposed by the BeagleBoard foundation were tested to verify that this new board was working as expected. The test examples present the use of the DSP and the TIDL library for machine vision using a webcam plugged to the USB port of the BeagleBone AI. [94]. It will permit the check that the DSP is accessible from the ARM Cortex A15 and that the heat sink placed on the chip diffuses enough energy to prevent the board from overheating.

Use of TIDL examples: A Logitech webcam was plugged into the BeagleBone AI, and the board was powered. After a warm-up period, the example codes were compiled and executed. During the experiment, CPU, GPU, and DSP temperatures were recorded every two minutes. After 5 minutes of inference, the board stops due to overheating the CPU, exceeding the maximum temperature accepted by the board (85C) before automatic shutdown. As the board was just released, no commercial solution to this overheating problem was available. The temporary solution was to cool down the board with a fan plugged into the VDD 3.3V pin. The same test was carried out, and the solution was found to be effective; soon after the foundation released a fan cape (Figure 4.2) for $6.99 which is a much cleaner solution to the problem.

In the following part we develop in more detail the PRU-ICSS subsystem before presenting the use of the BeagleBone AI Process-Realtime-Units for deterministic data acquisition.

Figure 4.2: The Fan Cape from [90].

4.2 PRU-ICSS subsystem

4.2.1 Presentation of the PRU-ICSS

The PRU-ICSS subsystems of the ti-am5xx chips are one of the main advantages on the BeagleBone boards over other similar boards like the Raspberry-Pi. Indeed, it gives the ability of executing some deterministic programs on the microprocessor. On the BeagleBone Bone AI there are 2 dual-core PRU-ICSS. They are 32bit microcontroller that are highly integrated with the rest of the system and can easily access the different port of the BeagleBone. The main limitation of the use of the PRU with some IO ports comes from some interference with the ARM-Cortex already using this ports (C.f. section 4.3.3).

The PRU is clocked at 200MHz which makes it among the fastest microcontroller in the market. For instance, they rank third at 16.66MHz, behind the iCEBreaker v1.0e (at 60MHz) and the Teensy 4.1 (at 23.03MHz) on a software ring oscillator test from the MIT center for Bits and Atom [95] (see Table 4.1).

The next section will present the PRU memory used to store the data sample before transferring them to the ARM. Thereafter, the communication between ARM and the

Table 4.1: Results from ring oscillator test from [95]

Frequency	Processor	Description	Date
60.00 MHz	iCE40UP5K	iCEBreaker V1.0e, 120 MHz, Verilog	September 2021
23.08 MHz	IMXRT1062	Teensy 4.1, 600 MHz, Fast, GPIO	August 2021
16.66 MHz	PocketBeagle	C, PRU	March 2019
4.616 MHz	ATSAME54	C, SRAM (aligned)	January 2021
4.00 MHz	ATxmega8E5	C, VPORT	October 2015

PRU-ICSS subsystems will be introduced, alongside with a how to run a code on the subsystem.

4.2.2 PRU-ICSS memory

Each PRU-ICSS core has 8kiB of memory that can be accessed in a minimal number of clock cycles as the data only needs to go through the PRU-ICSS interconnect (Figure 4.3). Additionally, each microcontroller can access the 8 KiB memory of the other core and a 32 KiB random access memory shared between the two cores in the same clock cycle.

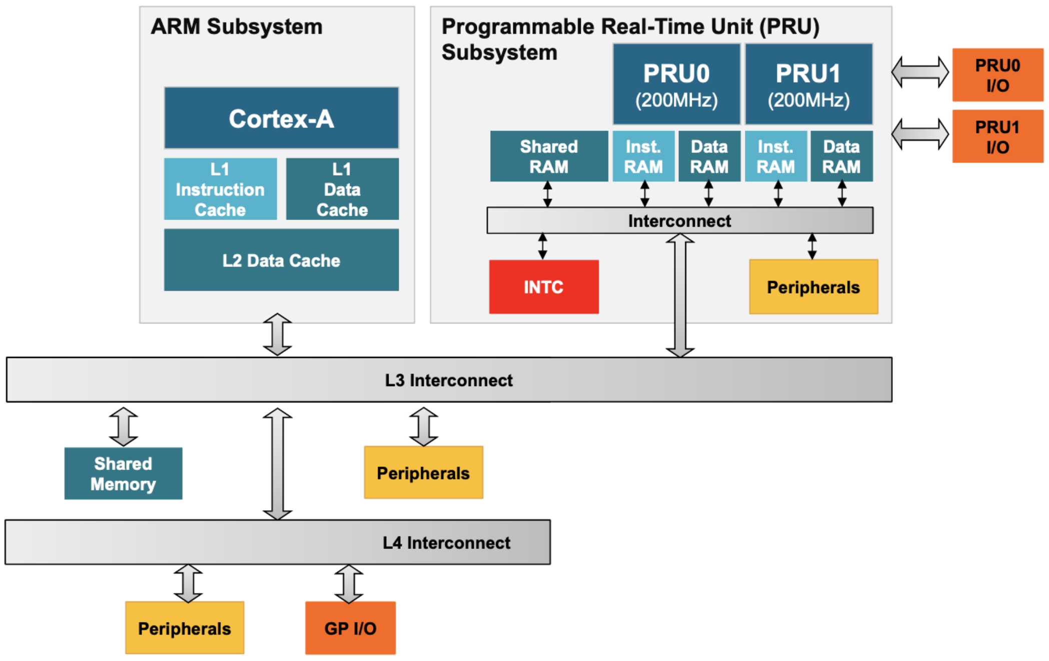


Figure 4.3: The PRU-ICSS subsystem and the Interconnect [96].

Assuming that each data point is stored as a float (or int), it will take 4 bytes; that is, for 8 KiB of memory, $\frac{8 \cdot 2^{10}}{4} = 2048$ data points can be stored. This storage could be doubled by using the memory of the other core and increased up to 8192 data points by using the 32 KiB shared memory. It should be noted that the two 8 KiB memories are continuous; the 8 KiB of one core is directly addressed before the 8 KiB of the other core, as presented in Table 4.2 from [97]. However, to access the 32 KiB of shared memory, there is an offset of $0x0001$.

Table 4.2: PRU-ICSS local memory map from [97].

Start Address	**PRUSS_PRU0**	**PRUSS_PRU1**
0x0000_0000	Data 8KiB RAM0	Data 8KiB RAM1
0x0000_2000	Data 8KiB RAM1	Data 8KiB RAM0
0x0000_4000	Reserved	Reserved
0x0001_0000	Data 32 KiB RAM2	Data 32 KiB RAM2

The most convenient solution is to have a default case where the data is placed in 32KiB the shared memory of the PRU-ICSS and if the combination of the sampling rate and the acquisition duration requires more than 8192 data points to use the two 8KiB of data RAM of each PRU-ICSS. This solution is limited to $2 \cdot 2048 + 8192 = 12288$ data points. The most extreme case where even more data points are required will not be considered here, but a workaround would use the AM5729 memory as storage. However, this solution would be limited because the data will have to go through the L3 main interconnect with access speed almost ten times slower than the PRU-ICSS memories.

As mentioned earlier, the BeagleBone AI has an onboard ADC connected to I2C1 (Figure 4.6). However, the ADC's direct sampling from the Linux host is not suitable for vibration analysis. Indeed, Linux is not a real-time operating system and the sampling rates could not be guaranteed. A Linux Kernel Module (LKM) has been introduced to overcome this limitation. The Industrial In/Out driver enables the sample of the I/O ports of a Linux-based SoC deterministically. However, it requires interrupting all other tasks handled by the kernel during the sampling operations. In the context of SoC's network, this represents a severe limitation to using the IIO driver in SoC's network as the BeagleBone would appear offline while sampling their ADC.

Because the PRU-ICSS are microcontrollers separated from the Linux host and their execution is deterministic; they are suitable for data acquisition. The next part presents the communication between the Cortex A15 running Linux and the PRU-ICSS.

4.2.3 Interfacing the ARM and the PRU-ICSS

The AM5729 structure does not give direct access to the PRU-ICSS microcontrollers. Instead, they are managed via the Linux ARM host, which, consequently, is responsible for [98]:

- Loading the PRU-ICSS firmware into the instruction memory of the PRU-ICSS cores
- Starting and Stopping the PRU-ICSS
- Establishing communication with the PRU-ICSS cores
- Since the PRU-ICSS is fully integrated into the chip, the ARM Linux also needs to manage its ports and interfaces with the different buses and interconnects.

To transfer a message from the Linux host to the PRU, the Linux host starts by placing the message in the VRING (allocated DDR memory used to transfer data by RPMsg), Figure 4.4 from [98]. Once the message is placed, the Linux host can notify the PRU-ICSS by posting the index of the buffer into the mailbox. This step is referred to as kicking the PRU-ICSS. The targeted PRU-ICSS will receive an interrupt and start to read the data from the VRING buffer, labels it as free, and notify the Linux host that the transaction is completed. The procedure is very similar to transmit a message from the PRU to the host ARM; the PRU writes in a VRING kick the ARM, the ARM can read the data from the buffer, frees it, and kicks back the PRU.

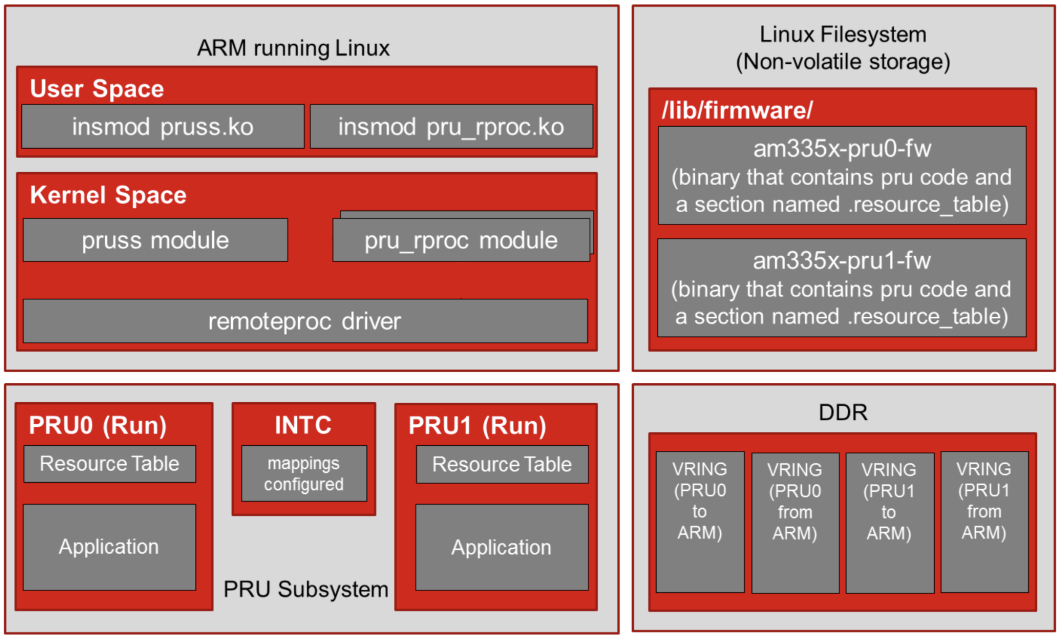


Figure 4.4: The VRINGS and the mailboxes used by remoteproc and RPMsg. [98].

4.2.4 Communication between the Host ARM and the PRU-ICSS subsystem

There exists two main ways to communicate between the PRU-ICSS cores and the ARM Cortex. The UIO driver used in libraries like Libpruio was used until 2018, but since Texas Instrument as well as the BeagleBone foundation are switching to the use of the Remote-Processor Messaging (RPMsg) driver. This new solution presents the advantage to be also compatible with the other processing cores of the AM5729 chip like the DSP and EVE. Finally, the two drivers are mutually exclusive and cannot be used simultaneously, and because this work aims to use both the PRU-ICSS subsystem and the DSP and EVE cores, the RPMsg driver will be used for this work.

The RPMsg driver exposes a file in the Debian file system under */sys/class/remoteproc/remoteprocN/state* where *N* is the index of the remoteprocessor. For the AM5729, there are 8 remote-processors presented in Table 4.3.

The Remote Processor driver is also used to start the code execution on the PRU. However, RPMsg transaction are limited in size to 492 bytes. Thus, it will not be possible to transfer the PRU data using the method. To overcome the above-mentioned limitation

Table 4.3: Remote Processor of the AM5729

Remote Processor index	**Device Name**
Remoteproc0	IPU 1
Remoteproc1	IPU 2
Remoteproc2	DSP 1
Remoteproc3	DSP 2
Remoteproc4	PRU1_0
Remoteproc5	PRU1_1
Remoteproc6	PRU2_0
Remoteproc7	PRU2_1

the data will be placed in the PRU memory.

4.2.5 Running a code on the PRU

There are again two ways to run some codes on the BeagleBone AI PRUs, Texas Instrument Software Development Kit (SDK) or directly compile and run the codes from a Linux Debian distribution running on the BeagleBone. This second solution is preferred to the cross compilation with the Texas Instrument Code Composer Studio software as it the way the BeagleBone community is the most likely to adopt the results of this work.

Consequently, to run some PRU C code, it needs to be compiled into a PRU firmware, this is achieved using the PRU C compiler directly on the BeagleBone AI. The BeagleBone foundation has released a precious Makefile that one can use to compile on the BeagleBone. This Makefile was adopted for the particular needs of this work. Once the code is compiled it is placed under the */lib/firmware/am57xx-pruX_Y-fw/* where *X* and *Y* are the PRU core. The PRU can then be started by echoing $start$ in the corresponding remoteproc state file. Then the remoteproc driver will load the binary from the firmware folder to the PRU instruction memory and prepares the resources that the PRU will need during the execution of the code.

The linked repository presents basic examples of the use of PRU:

- LED blink: to blink and user LED at a given frequency from the PRU
- RPMsg Echo: a message sent from the ARM is send back to the PRU

In the following part, we present the implementation of a deterministic data acquisition process on the BeagleBone AI using the PRU-ICSS and the onboard Analog to digital converter.

4.3 Deterministic data acquisition implementation

The PRU-ICSS subsystem of the BeagleBone AI will be connected to an I2C accelerometer, however, there was no I2C driver at the time of this work.

Thus, a driver needs to be written for the PRU-ICSS to communicate with the I2C controller. This driver can be written in ASSEMBLY using the PRU ASSEMBLY instruction set [99] or in C99 using the Texas Instrument PRU C Compiler. The following section will present the development of the driver mentioned above.

4.3.1 Development of an I2C driver

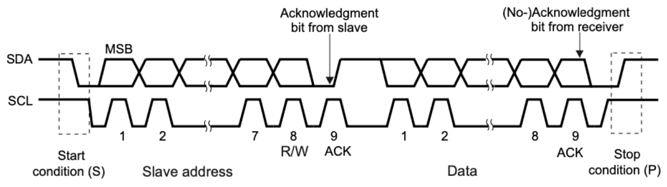
Here, we present the driver's implementation that enables the communication between the PRU-ICSS and the Multi-master High Speed I2C Controllers of the Ti -Am5729 chip.

An I2C transaction

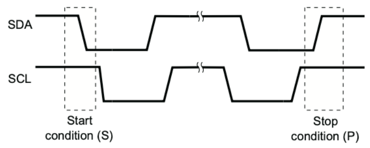
I2C stands for Inter-Integrated Circuit, it is a communication protocol developed by Phillips in 1982. I2C allows multiple devices to communicate using two lines:

SCL line: provides clock pulses for the devices to have synchronized communication.

SDA line: to transmit the data, including the devices addresses, start and stop conditions of the transfer.



(a) I2C Transaction



(b) Start and Stop condition

Figure 4.5: I2C transaction schematics[97].

In its original version, devices on the bus use an 8 bits address (7 address bits and 1 Read/Write bit) allowing up to 112 devices to be connected on the same bus. However, some devices currently support a 10 bit address.

A I2C transaction is carried out between a master and a slave device. The Master initiates the communication by sending a start condition on the bus, followed by the 7 bits address of the slave and the read/write bit. Once the targeted slave device has sent the acknowledgment (ACK) bit, the master send another start condition and then either sends or receives data. Finally, after the last ACK the master sends a stop condition on the bus. In order to configure an I2C sensor on the bus, the I2C controller of the BeagleBone AI needs to perform the following steps:

- Send the address of the device with a write bit.
- Send the address of the sensor's internal register with a read or write bit depending if it wants to read the value of the register or send data to it.
- Either wait to receive the data or write the data in the bus.

Presentation of the BeagleBone AI HSI2C module

The AM5729 chip has five I2C controllers compliant to the Phillips I2C standards; they support a standard mode at 100kHz and a fast mode at 400kHz; some the controllers also support a High speed mode at 3.4MHz.

The controller's memory registers can be accessed via the L4 Interconnect of the chip, and the Power Reset and Clock Management (PRCM) Module provides the clock for the controller. The data can be transmitted to the host using interrupt or Direct Access Memory (DMA) transfers. In the host interrupt, the HSI2C Controller notifies the Local Host (LH) that new Data is accessible in the data register by raising interrupt flags. Once the flags are raised, the host can get the register's data and place it in memory. On the contrary, Direct Access Memory (DMA) does not require the host to transfer it. The HSI2C and the DMA controllers can transfer the data directly to memory. DMA's advantage over interrupt is that it frees the host from the data transfer task. However, in the proposed architecture, the PRU-ICSS is in charge of the data acquisition and can handle the data transfer when interrupts are raised. Consequently, the data will be read from the HSI2C data registers using the PRU and interrupts.

The different steps are directly carried out by the controller and the PRU only needs to configure its registers. To simplify the interaction between the PRU and the controller, a I2C driver was implemented.

Implementation of the I2C driver

As presented in Figure 4.6, The BeagleBone AI on-board ADC is connected to the chip on I2C1 and the PRU can control the I2C controller. However, this ADC is also required by the Linux host and cannot be used as a dedicated ADC for data acquisition. Consequently, an external I2C accelerometer was used with I2C4 in order to carry out the data acquisition.

A PRU-I2C library was implemented in C for the BeagleBone AI. It exposes basis function to setup the I2C controller using the PRU. The codes are available on the Github linked. The library contains a user space and a PRU code. The user space is used to communicate with the PRU using RPMsg and to tell the PRU to start the I2C controller acquisition. The repository contains two folders a main with the codes to be executed on

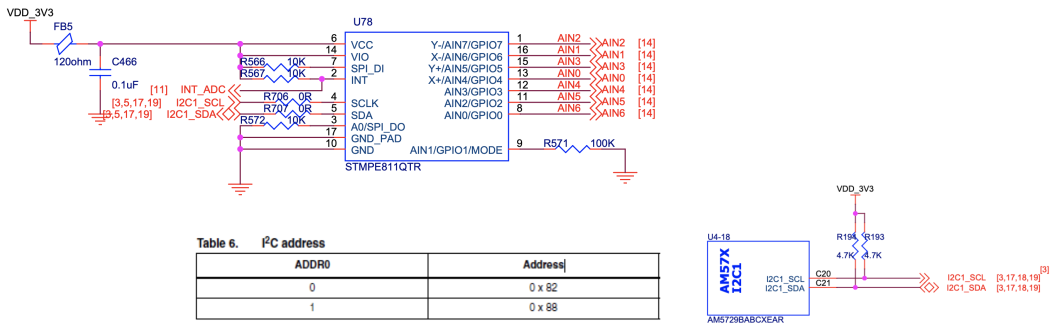


Table 6. I²C address

ADDR0	Address
0	0 x 82
1	0 x 88

(a) ADC (b) I2C controller

Figure 4.6: I2C Bus wiring[100].

the user space side (ARM Linux) and PRU directory. Under the PRU directory is located the drivers codes that exposes the following function:

pru_i2c_driver_init: initializes the I2C controller according to the Texas Instrument documentation. The CM_L4 _PER_I2C1_CLKCTRL gate is close to provide clocks to the I2C controller. Then the prescaler is initiated to obtain a 100kbps using Table 24-6 and 24-7 of the Technical Reference Manual SPRUHZ6L. Finally, the controller is enabled and the address of the Slave device on the bus is provided.

pru_i2c_driver_transmit_byte: transmits a bite of data to a slave device by first specifying the slave's register to write the data to.

pru_i2c_driver_transmit_bytes: transmits multiple bites of data to a slave device by first specifying the slave's register to write the data to. This function can also be used to send the configuration to a device by sending an array of bytes. The first byte being the data for the specified register and then an alternation between the internal address of the register in the slave and the data to send to this register.

pru_i2c_driver_receive_byte: receives a byte of data to a slave device by first specifying the slave's register to receive the data from.

pru_i2c_driver_receive_bytes: receives multiple bytes of data to a slave device by first specifying the slave's register to receive the data from. After sending the address of the first register to receive data from; the I2C bus will wait for the slave device to send the specified number of bytes. Most of I2C devices will directly auto-increment the address of the register to send the data from. Consequently, to read multiple consecutive register, one can call this function on the first register's address and specify the number of register to read after.

pru_i2c_driver_software_reset: performs a software reset according to section 24.1.4.3 of the Technical reference manual

In addition, the driver exposes some helper functions to poll the I2C controller internal registers.

pru_i2c_poll_I2C_IRQSTATUS_RAW_XDR: Pool the I2C_IRQSTATUS_RAW register on bit XDR and return 1 if transmit draining is active.

pru_i2c_poll_I2C_IRQSTATUS_RAW_RDR: Pool the I2C_IRQSTATUS_RAW register on bit RDR and return 1 if a draining feature is active.

pru_i2c_poll_I2C_IRQSTATUS_RAW_BB: Pool the I2C_IRQSTATUS_RAW register on bit BB and return 1 if the bus is busy.

pru_i2c_poll_I2C_IRQSTATUS_RAW_XRDY: Pool the I2C_IRQSTATUS_RAW register on bit XDRY and return 1 if the controller is ready to transmit.

pru_i2c_poll_I2C_IRQSTATUS_RAW_RRDY: Pool the I2C_IRQSTATUS_RAW register on bit RDRY and return 1 if the received data is ready to be read.

pru_i2c_poll_I2C_IRQSTATUS_RAW_ARDY: Pool the I2C_IRQSTATUS_RAW register on bit ADRY and return 1 if the access to the register is ready.

pru_i2c_poll_I2C_IRQSTATUS_RAW_NACK: Pool the I2C_IRQSTATUS_RAW register on bit NACK and return 1 if the not NACK (No Acknowledge) as been detected.

pru_i2c_poll_I2C_IRQSTATUS_RAW_AL : Pool the I2C_IRQSTATUS_RAW register on bit AL and return 1 if an arbitration loss between two master on the same I2C line as been detected.

pru_i2c_poll_I2C_IRQSTATUS_RAW_BF : Pool the I2C_IRQSTATUS_RAW register on bit BF and return 1 if the bus is free.

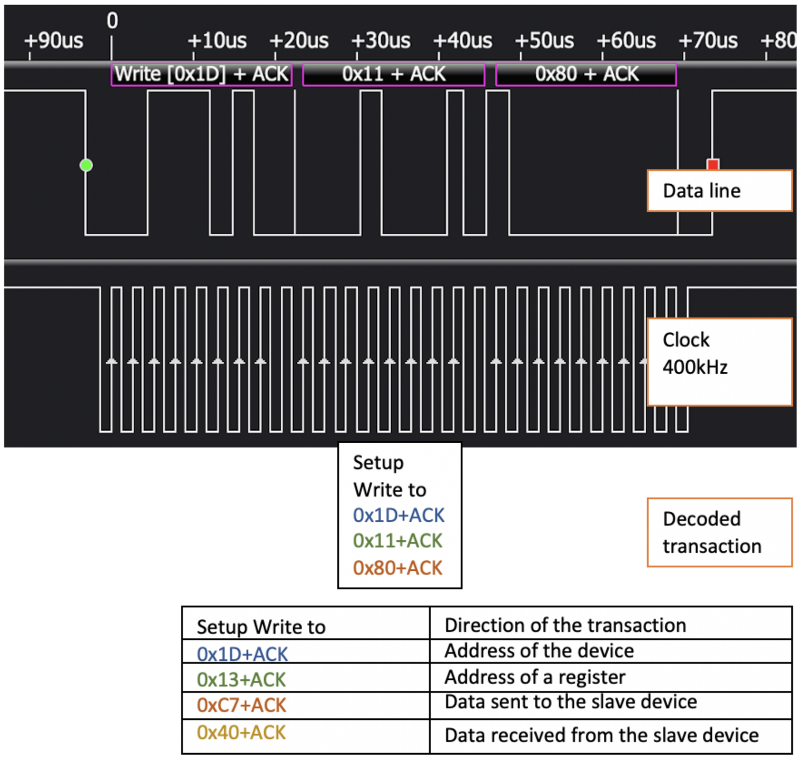
Using these functions and the main program, one can easily configure the I2C controller, read or place data in the I2C_DATA register which is the FIFO end point for the DATA transferred through the I2C bus.

4.3.2 Driver's validation

The I2C transaction were validated using an LA1010 I2C Bus analyzer from Kingst Logic Analyzer. An I2C sensor was connected to the I2C4 Bus of the BeagleBone. The read and write register functions were tested, and the transaction are presented on figure 4.7 and 4.8. The bus was configured to communicate at 400kHz (Fast Mode) and that is also verified on Figure 4.7 with a measured clock frequency of 398kHz. In the following we present the validation of the major function of the I2C driver:

pru_i2c_driver_transmit_byte: Figure 4.7 presents a simple byte transmission between the BeagleBone AI and a sensor's register. First the I2C controller of the BeagleBone AI specifies the address of the sensor on the I2C bus. After, Acknowledgment of the selected sensor the controller send the address of the register it wants to write to, and finally it can send the data byte for the sensor to write to these registers. The transaction it finalized by the sensors acknowledging the reception of the data and the bus is freed.

pru_i2c_driver_receive_byte: Figure 4.8 presents the reception of single byte between the BeagleBone AI and a sensor's register. This transaction starts by the controller sending the address of the sensor's register it wants to received data from. Then the



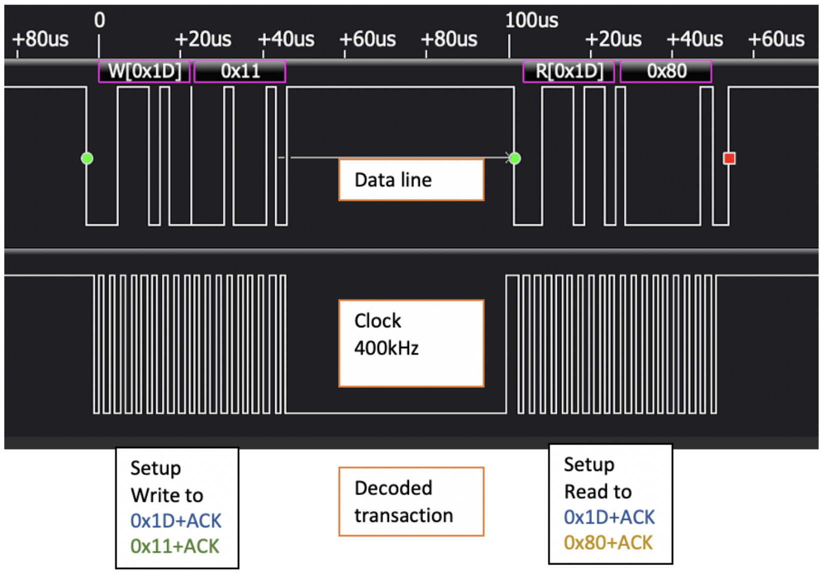
Setup Write to	Direction of the transaction
0x1D+ACK	Address of the device
0x13+ACK	Address of a register
0xC7+ACK	Data sent to the slave device
0x40+ACK	Data received from the slave device

Figure 4.7: Single Byte transmission.

sensors waits to receive a read instruction from the controller. Once this instruction is received and acknowledged, the sensor can send the data from its register.

pru_i2c_driver_transmit_byte and pru_i2c_driver_receive_byte: Figure 4.9 presents a simple byte transmission between the BeagleBone AI and a sensor's register followed by a simple read from the sensor's register. This is presented to validate that the software reset perform between the transmission and the reception is working correctly. Additionally, a minimum of 6000 PRU clock cycle was required to wait for the reset to be performed correctly. Below this delay, the I2C controller would not send the correct data in the bus.

pru_i2c_driver_transmit_bytes and pru_i2c_driver_receive_bytes: Figure 4.10 presents the transmission of multiple bytes followed, by the reception of multiple bytes. This function is mainly used to send a configuration to a device and to validate that this configuration was written correctly in the devices internal



Setup Write to	Direction of the transaction
0x1D+ACK	Address of the device
0x13+ACK	Address of a register
0xC7+ACK	Data sent to the slave device
0x40+ACK	Data received from the slave device

Figure 4.8: Single Byte reception.

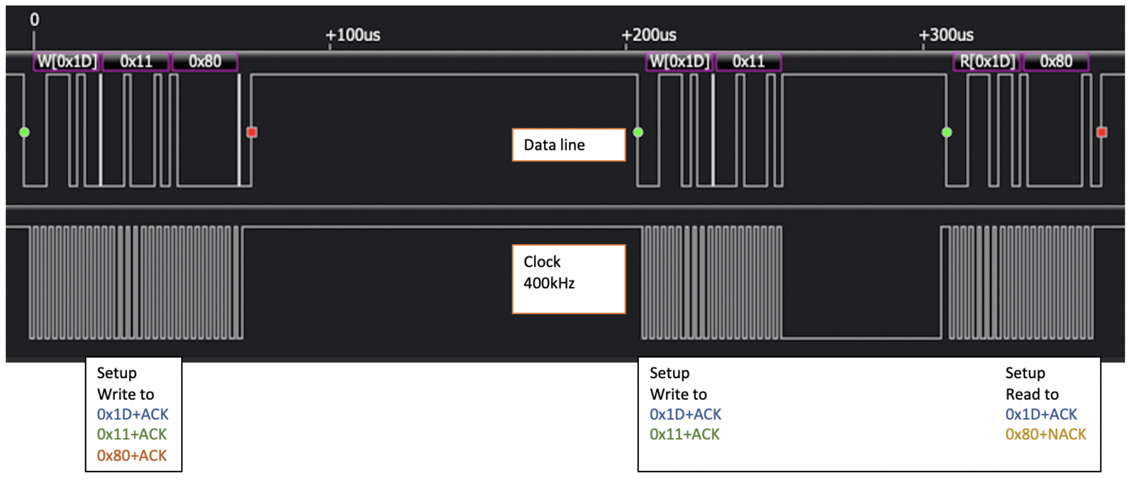
registers. It is also used during data acquisition to read multiple consecutive data registers taking advantage of the auto-increment feature of the I2C slaves. This presents the advantage of saving some communication time by not having to send the address of all the register that one wants to read.

4.3.3 Deterministic Data Acquisition

Initial plan

The Data acquisition schema is based on the considerations of section 4.2.2. As, RPMsg is limited to 492 bytes per transaction, it will not be used for data transfer and the PRU will store the data in the 32KiB shared memory of the two microcontrollers.

In order to acquire data, the user space (Linux ARM) notifies via RPMsg the PRU that it needs new data. The PRU configures the I2C controller for the I2C data transmission.



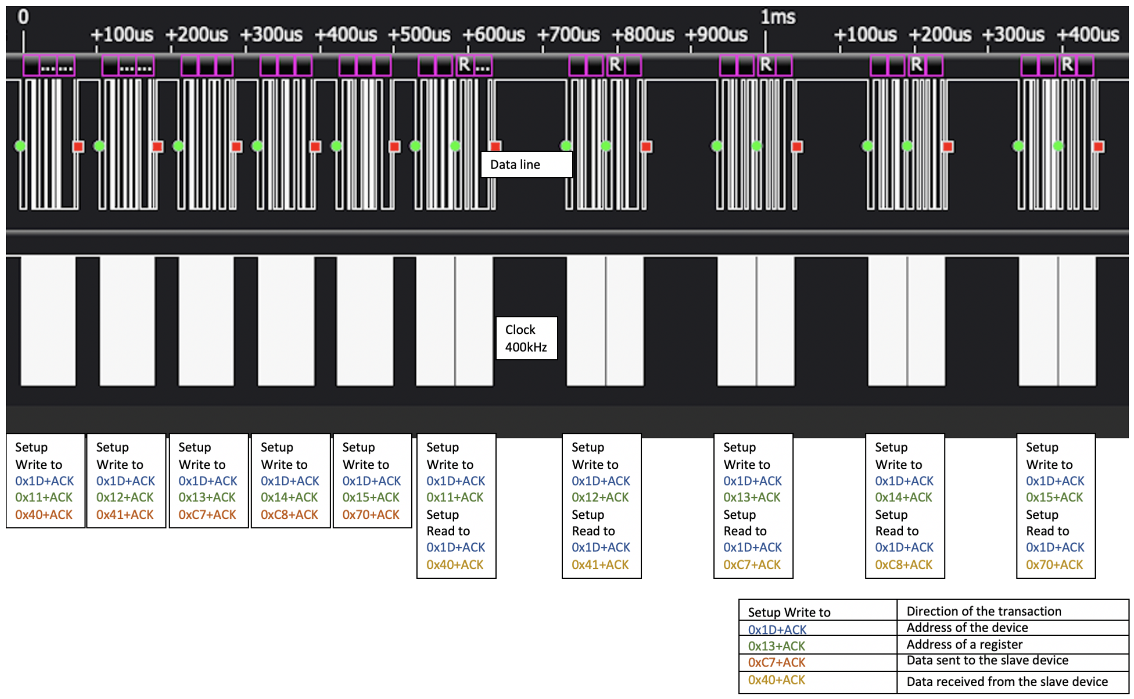
Setup Write to	Direction of the transaction
0x1D+ACK	Address of the device
0x13+ACK	Address of a register
0xC7+ACK	Data sent to the slave device
0x40+ACK	Data received from the slave device

Figure 4.9: Single transmission followed by single receive.

The I2C controller obtains the data from the ADC and places it in its I2C_DATA register of the I2C bus. Then, PRU reads the I2C_DATA registers and places the data to the shared memory address provided by the ARM. Once the data is ready in the shared memory, the PRU notifies the ARM that data can be accessed from the user space. The Linux ARM can then transfer the data by reading the PRU shared memory via the */dev/mem/*. The data is then copied from the shared memory to the RAM of the Cortex A15 through the L3 main interconnect.

Limitations

The above-mentioned architecture has a major limitation that was not foreseen at the beginning of the work. This limitation is that, to ensure that the reading of the ADC are correct, we have to be sure that the I2C1 Controller is always available for the PRU to use. Any interference from one of the other core of the BeagleBone AI will hinder the deterministic data acquisition. However, the onboard ADC is used by one of the Linux Kernel Driver of the BeagleBone AI essential tasks such as temperature reading.


0
+100us +200us +300us +400us +500us +600us +700us +800us +900us
1ms
+100us +200us +300us +400us
Data line
Clock 400kHz
Setup Write to 0x1D+ACK 0x11+ACK 0x40+ACK
Setup Write to 0x1D+ACK 0x12+ACK 0x41+ACK
Setup Write to 0x1D+ACK 0x13+ACK 0xC7+ACK
Setup Write to 0x1D+ACK 0x14+ACK 0xC8+ACK
Setup Write to 0x1D+ACK 0x15+ACK 0x70+ACK
Setup Write to 0x1D+ACK 0x11+ACK Setup Read to 0x1D+ACK 0x40+ACK
Setup Write to 0x1D+ACK 0x12+ACK Setup Read to 0x1D+ACK 0x41+ACK
Setup Write to 0x1D+ACK 0x13+ACK Setup Read to 0x1D+ACK 0xC7+ACK
Setup Write to 0x1D+ACK 0x14+ACK Setup Read to 0x1D+ACK 0xC8+ACK
Setup Write to 0x1D+ACK 0x15+ACK Setup Read to 0x1D+ACK 0x70+ACK

Setup Write to	Direction of the transaction
0x1D+ACK	Address of the device
0x13+ACK	Address of a register
0xC7+ACK	Data sent to the slave device
0x40+ACK	Data received from the slave device

Figure 4.10: Multiple bytes' transmission followed by multiple read transmission.

Multiple attempts were made to unload the driver from the Kernel, but this was not successful. Consequently, some revisions were made to the architecture to by-pass the onboard ADC.

Final Implementation for Deterministic Data Acquisition

As aforementioned, the ADC of the BeagleBone AI is required by the Linux Kernel running on the ARM Cortex. Consequently, the deterministic data acquisition with this ADC cannot be guaranteed. This solution also provides the advantage of a better resistance to potential electromagnetic perturbations during the data acquisition. Indeed, using a digital signal instead of a voltage on the wire connecting the sensor to the BeagleBone AI makes the system more robust.

To overcome this limitation, instead of using an analog sensor, a digital sensor with I2C capabilities was used.

The considered sensors are presented in table 4.4.

Table 4.4: Selection of I2C accelerometers.

Name	Price	Accel Range	Max Sampling Rate	Resolution
ADXL345	\$18.95	±2g/4g/8g/16g	3.2kHz	10bits
ADXL313	\$15.95	±0.5g/1g/2g/4g	3.2kHz	13bits
KX134	\$19.95	±8g/16g/32g/64g	25.6kHz	16bits
KX132	\$13.95	±2g/4g/8g/16g	25.6kHz	16bits
LSM6DSO	\$11.95	±2g/4g/8g/16g	6.6kHz	16bits

Based on the cost and characteristics of the sensors of table 4.4.

The *KX134* (Figure 4.11 from [101]) was selected for this work. It enables a high precision of 16 bits with a high maximum out put data rate of 25.6kHz while also being low cost.

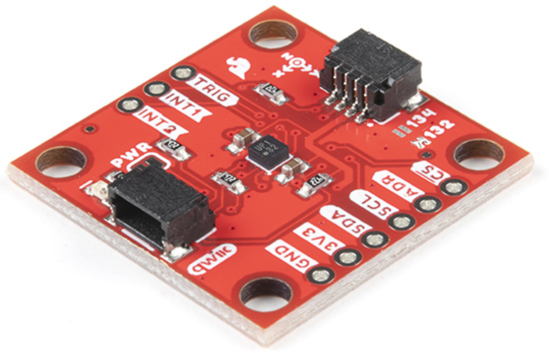


Figure 4.11: KX132 I2C accelerometer selected for this work.

This accelerometer will be connected via I2C to the BeagleBone AI I2C4 bus. The driver is then used to send the configuration information to the sensors.

Configuration of the KX134: The maximum output data rate of the KX132 is of 25.6kHz, however, the data needs to be transferred through the I2C bus at least at the same speed. Otherwise, even if the data points are sampled at a given data rate, the PRU will not be able to receive the data from the sensor register's values before the next sample.

Additionally, the data is sampled in 16 bits precision which decreases the maximum output data-rate even more, as for each data sample, two consecutive 8 bits data registers needs to be read.

Finally, other information needs to be communicated through the bus, such as the value of the *DATA_READY KX132_INS2* register which indicates if a new sample is placed in the data registers. This register needs to be pulled at least twice for each sample: this ensures that the first time the data is not ready and the second time the data is ready, and the PRU can request the data from the two data registers. The clock of the I2C bus being set at 400kHz, the maximum output data rate (ODR), to ensure the strict validity of each sample was found to be 1.6kHz. The other configuration parameters of the sensors are High performance, High resolution mode, with a ±4g range and a pulling method for the data transmission.

Data acquisition sequence: Algorithm 2 presents the sequence to acquire 1024 data points at 1.6kHz

Algorithm 2 Data Acquisition from the PRU perspective

```
PRU request KX132_WHO_AM_I register's value
if KX132_WHO_AM_I != WHO_AM_I then
    return error sensors not detected
end if
PRU: 0x00 → KX132_CNTL1                          ▷ stop KX132 for reconfiguration
PRU: 0x0b → KX132_ODCNTL                                      ▷ Set ODR to 1.6kHz
PRU: 0xe8 → KX132_CNTL1   ▷ Set High perf., high-res., 4g range, pulling data and
wake-up sensor
for each sample N=0,...,1024 do
    while KX132_INS2!= 0x10 do                   ▷ Wait for new data to be ready
        PRU: read KX132_INS2
    end while                                                    ▷ Data is ready
    PRU: data[N] ← KX132_XOUT_L,KX132_XOUT_H            ▷ Read 2 consecutive
registers for low and high byte
end for
```

Expected limitation of the proposed architecture

As mentioned above, the main limitation of this architecture does not come from the sensor maximum output data rate (25.6kHz), but from the speed of the I2C bus limited by the clock at 400kHz. It was found experimentally that: 1.6kHz is the maximum sampling rate at which the KX132_INS2 is pulled at least once (in reality the register is pulled up to 4 times before data is ready) before the data is ready, ensuring that no data point is missed. At a sampling rate of 3.2kHz the *DATA_READY* register is, almost every time, pulled more than once, indicating that this sampling rate could be achieved too. However, the most conservative choice is made and the data-rate is limited at 1.6kHz.

Actions to increase data-rate if needed

First, the 3.2kHz data-rate can be used by dropping the 16 bits precision to only read 8 bits, which should be enough to ensure that every reading is validated. Second, both the KX132 sensors and the I2C4 controller of the BeagleBone AI are rated for high speed I2C communication. Even if the driver configures the clock at 400kHz, it can be easily adapted to enable a 3.2MHz clock. These two recommendations will increase the maximum possible Output Data rate to be very close to the maximum achievable by the sensor (25.6kHz).

4.4 Data processing

Once the data is placed in the Cortex RAM, the data processing can start. As discussed in introduced, in hypothesis 1, in section 3.2, this work will use time-frequency analysis of the vibration signal to obtain information about the bearing defect. The proposed work was suggesting to use the GPU of the AM5729 to process the wavelet transform.

However, the implementation revealed that the GPU of the chip is not OpenCL compatible, and it is only OpenGL compatible. OpenGL being a graphical framework, it is not adapted to express operations such as Wavelet transforms. Thus, the implementation of the wavelet transform would be much more complex than necessary if done on the GPU. The EVE and DSP cores of the AM5729 are OpenCL compatible. However, they will be used by the Machine Learning Algorithm converted via TIDL. So the choice is made to process the wavelet transform using some library on the ARM CPU.

A Python main code runs on the ARM; when it receives the message via RPMsg that the vibration data is written in memory, the CPU fetches this data and executes a discrete wavelet transform. Using a discrete transform is motivated by the difference in processing time between Continuous and Discrete Wavelet Transforms. The Python library PyWavelet was used to estimate the complexity of the Algorithm for both Continuous and Discrete Transforms. On a Macbook Pro 2019 with an Intel Core I7 at 2.8GHz, the processing of a signal of 2048 data points takes around 7 seconds for a Continuous Wavelet Transform with 150 scales and only 10 milliseconds for a discrete wavelet with 7 scales which motivates the choice to use Discrete Wavelet Transform over Continuous Wavelet Transforms to process the vibration data at the edge with the much smaller CPU of the AM5729. Then, the result of the wavelet transform will be transferred to the DSP for inference with the TIDL Library.

4.5 Edge inference of bearing status

4.5.1 Introduction to Texas Instrument Deep Learning Library

On the AM5729, Texas Instrument released a Deep Learning Library to enable the use of the Deep Learning Models on the EVE engines and the DSP cores of their edge devices. This library helps to convert pretrained models into edge executable models. It supports model trained in on the following frameworks: Caffe, Caffe-Jacinto, or Tensorflow. Once the model is trained on an external device (PC or Cloud) it has to be converted into a binary format using the Texas Instrument import tool provided in the SDK. The obtained binary can then be loaded in the edge devices and run on the DSP or EVE cores from the

ARM. Figure 4.12 from [102] presents the flow to train, convert and execute a model with the TIDL Library on an Edge device.

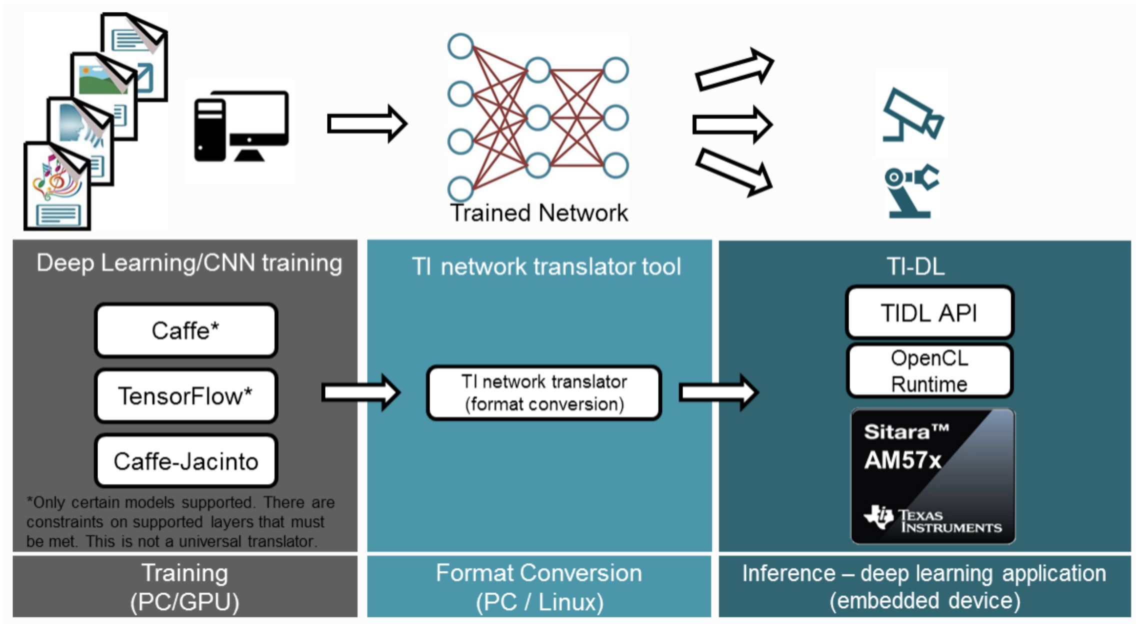

Trained Network
Deep Learning/CNN training
Caffe*
TensorFlow*
Caffe-Jacinto
*Only certain models supported. There are constraints on supported layers that must be met. This is not a universal translator.
TI network translator tool
TI network translator (format conversion)
TI-DL
TIDL API
OpenCL Runtime
Sitara™ AM57x
Texas Instruments
Training (PC/GPU)
Format Conversion (PC / Linux)
Inference – deep learning application (embedded device)

Figure 4.12: Process to train and execute a model with TIDL[102].

4.5.2 Limitation of the library

Even though this library enables this execution of powerful models on the edge device, it comes with multiple limitation.

First, the import tools needs to run on a Windows machine or on a Linux Debian 18.x which is not the OS supported by the BeagleBone and this prevents the retraining of the models at the edge.

Second, the convert tool only support some Deep Learning models, and it is not possible to use personalized architectures as stated in [103].

Third, Texas Instrument provides some pretrained model on some classical dataset such as MobileNetV2 trained on the MNIST dataset. Those models import correctly with the import tool. However, the import fails on the same models once they are retrained in TensorFlow. This makes the library impractical for applications different from what the models have been trained for.

Finally, there is a significant lack of documentation for the ti-am5729 as most of the focus from Texas Instrument seemed to have shifted to a next version chip, the TDA2Px.

Extensive work has been done to try using the TIDL library and multiple tickets have been raised on Texas Instrument support forum. However, the numerous afford-mentioned limitations, the fact that the models cannot be retrained at the edge, as well as the lack of support for ti-am5729 makes it very hard to use the library on the BeagleBone AI.

Consequently, the library will not be used in this work and TensorFlow models will be used in the format .tflite for the proposed implementation.

Finally, the next version of the BeagleBone AI which should be released in 2022 will be based on the TDA2Px chip and let user take advantage of the new Deep Learning Framework from Texas Instrument Edge AI [104], which is supposed to support .tflite model without the limitation of the current library.

4.6 Implementation on TensorFlow

4.6.1 Introduction to TensorFlow

TensorFlow [105] is an open source deep learning framework supported by Google. It is with PyTorch (from Facebook), it is one of major the solution for deep learning. The main advantage of TensorFlow over the other frameworks is its ease of use with many easily accessible tutorials [106] that can be run on Google Colab notebooks. TensorFlow also lets user export the models in the “lite” format .tflite that possible to use on edge devices such as the BeagleBone AI. Finally, as mentioned in 2.8.2, the concept of Federated Learning has been introduced by Google which give an edge to TensorFlow for Federated Learning implementations.

The following will present the implementation of a Centralized Learning solution for bearing defect detection that will be compared with the Federated Learning solution presented in section 4.7

4.6.2 Artificial Neural Network

In section 4.5.2, the use of TensorFlow model on the BeagleBone AI was preferred. On the current version of the BeagleBone AI, these models will have to be executed on the CPU as they will not be compatible with the TIDL library. However, it should be possible to execute them in the next version on the board running TDA2Px with EdgeAI. The fact that the model will be run on the CPU and not on the specialized processor limits the models to simple Artificial Neural Networks (ANN). Five different ANN have been selected for the centralized learning, and they will be used in the Federated Learning context too in order to have a fair comparison between the two different settings. Table 4.5 to 4.10 present those architectures. It should be noted that only minimal work has been done on the optimization of the hyperparameters of those models as the possibilities are endless. The different model were selected because they gave good results on at least one of the dataset, but future work could look at the optimization of these models. Finally, as a wavelet transforms is carried out on the waveform before classification with the ANN, we will not add convolution layer in the architecture.

Table 4.5: Model 0 Layers, Shapes and parameters

Layer (activation)	**Output Shape**	**Param #**
Dense_1 (relu)	(None, 512)	20992
Dropout_1 (0.3)	(None, 512)	0
Dense_2 (sigmoid)	(None, 64)	32832
Dropout_2 (0.3)	(None, 64)	0
Dense_3 (sigmoid)	(None, 32)	2080
Dropout_3 (0.3)	(None,32)	0
Dense_4 (sigmoid)	(None, 16)	528
Dense_5	(None, 1)	17

4.7 Federated Learning

TensorFlow presents an Application Program Interface (API) for federated learning:

Table 4.6: Model 1 Layers, Shapes and parameters

Layer (activation)	Output Shape	Param #
Dense_1 (relu)	(None, 128)	5248
Dropout_1 (0.3)	(None, 128)	0
Dense_2 (sigmoid)	(None, 64)	8256
Dropout_2 (0.3)	(None, 64)	0
Dense_3 (sigmoid)	(None, 32)	2080
Dropout_3 (0.3)	(None, 32)	0
Dense_4 (sigmoid)	(None, 16)	528
Dense_5	(None, 1)	17

Table 4.7: Model 2 Layers, Shapes and parameters

Layer (activation)	Output Shape	Param #
Dense_1 (relu)	(None, 256)	10496
Dropout_1 (0.3)	(None, 256)	0
Dense_2 (sigmoid)	(None, 64)	16448
Dropout_2 (0.3)	(None, 64)	0
Dense_3 (sigmoid)	(None, 32)	2080
Dropout_3 (0.3)	(None, 32)	0
Dense_4 (sigmoid)	(None, 16)	528
Dense_5	(None, 1)	17

Table 4.8: Model 3 Layers, Shapes and parameters

Layer (activation)	Output Shape	Param #
Dense_1 (relu)	(None, 128)	5248
Dense_2 (sigmoid)	(None, 64)	8256
Dense_3 (sigmoid)	(None, 32)	2080
Dense_4 (sigmoid)	(None, 16)	528
Dense_5	(None, 1)	17

Table 4.9: Model 4 Layers, Shapes and parameters

Layer (activation)	Output Shape	Param #
Dense_1 (relu)	(None, 256)	10496
Dense_2 (sigmoid)	(None, 64)	16448
Dense_3 (sigmoid)	(None, 32)	2080
Dense_4 (sigmoid)	(None, 16)	528
Dense_5	(None, 1)	17

Tensor Flow Federated TFF. It enables the simulation of large fleets of devices for federated learning with the training and evaluation on edge devices.

Table 4.10: Model 5 Layers, Shapes and parameters

Layer (activation)	Output Shape	Param #
Dense_1 (relu)	(None, 128)	5248
Dropout_1 (0.3)	(None, 128)	0
Dense_2 (sigmoid)	(None, 64)	8256
Dropout_2 (0.3)	(None, 64)	0
Dense_3 (sigmoid)	(None, 32)	2080
Dropout_3 (0.3)	(None, 32)	0
Dense_4 (sigmoid)	(None, 32)	1056
Dense_5	(None, 1)	17

This permit the validation of the federated learning approach for bearing defect detection on the proposed dataset. The API requires the use of a federated dataset, which differs from a centralized dataset as it is split between different clients. Moreover, contrary to a centralized setting, the federated learning dataset needs to be non-IID as there is no reason for the dataset to be iid across the different clients. However, this API is still in development, so it will not be used in this work.

Consequently, a custom federated learning API was developed. It acts as a wrapper around the TensorFlow framework, and creates a federated learning server to manage clients. The server can require a given set of clients to retrain their own model and to send back the weight, the server aggregates the weight with respect to the Federated Averaging algorithm presented in 1. The API uses an object-oriented paradigm, the federated_server exposes the following methods:

initiate_federated_learning_process creates the required number of clients for the federated learning process and build the TensorFlow model in the server as a model cannot receive weight before being built.

propagate_weights_to_all_clients takes the weights of the server's model and send them to all the clients of the network.

federated_averaging_algo is called to fetch the weights from the retrained clients and average them according to 1. The server local weight can then be updated and the propagate_weights_to_all_clients function is called

do_one_round takes advantages of the previous functions to do one iteration of the federated learning process. The server requests the retraining of the model of a number of randomly selected clients. Once clients have retrained their models the federated_averaging_algo methods is used which results in an updated version of the server's weights. After that, the server's weights are propagated to all the clients (even the one which were not retrained)

The clients have the following methods:

retrain_model , when requested by the server, this function retrains the client model with its current dataset.

update_model_weights is used to set the clients weights to the new weights received by the server

get_balance_train_data: obtain a balanced dataset for training where the number of defective sample is the same as the number of healthy sample.

get_balance_test_data: same as above but for a test set.

In order to use the library, first the federated learning server needs to be created and initialized with the desired number of clients. Then the server can set up round of federated learning on as many clients as required and propagates the new weights obtained via the federated averaging algorithm.

4.8 Reviewed architecture

The initial proposed architecture had to be adapted, considering that:

1. The Analog to Digital Converter of the BeagleBone AI is used by the Linux Host
2. The GPU is not OpenCL compatible, but only OpenGL compatible. It was not possible to implement the wavelet transform on the GPU.
3. Texas Instrument Deep Learning is limited and there is a lack of documentation and support for this library on the AM5729.
4. The next version of the BeagleBone AI is supposed to directly support TensorFlow models.
5. It takes longer to initiate the specialized cores than to carry out the discrete wavelet transforms directly from the CPU.

The following modifications were introduced:

- An external accelerometer is used to acquire the data and to transmit them to the PRU via the I2C4
- The Discrete Wavelet transforms as well as the inference of the machine learning models will be carried out on the CPU

In this chapter, we have presented the implementation of the data acquisition and the edge processing of the wavelet transforms. Even if some major limitation were found, the final goal of having a system that sample deterministic vibration data while performing wavelet transforms and inference of machine learning models was achieved.

In the next chapter, we will test this data acquisition on the Machine Fault Simulator for an Outer Race defect. Different types of wavelet transforms will be tested, performance metrics will be introduced, and the results will be compared with peak finding method. The use of this federated learning algorithm is presented in chapter 5 where the performance of the proposed architecture are compared in both a centralized context and a federated learning context.

CHAPTER 5

VALIDATION OF THE PROPOSED ARCHITECTURE

In this chapter, we will test the deterministic data acquisition architecture proposed in chapter 4. We start by presenting the data acquisition on the Machine Fault Simulator. Then, we introduce another data set available online. Thereafter, we explain the metrics used to compare different bearing defect detection methods. Finally, we compare a peak finding method and a time series method to the results obtained with the proposed architecture in the context of a centralized learning and federated learning.

5.1 Machine Fault Simulator data acquisition

In this section, we present the acquisition procedure for the Machine Fault Simulator used at GeorgiaTech.

5.1.1 Presentation of the Machine Fault Simulator

The Machine Fault Simulator is a vibration test bench distributed by Spectral Quest Inc. [107]. It is power by a three phases, 1HP, asynchronous motor that can drive the main shaft up to 6000rpm (100Hz). The motor rotational speed is regulated with tachometer that displays the speed of the shaft on a mounted LCD.

Figure 5.1 presents the Machine Fault Simulator. On the left of the picture are located the tachometer and the motor that rotate the shaft through a metallic coupling. Two bearings (black) are located on the mounting on the left and

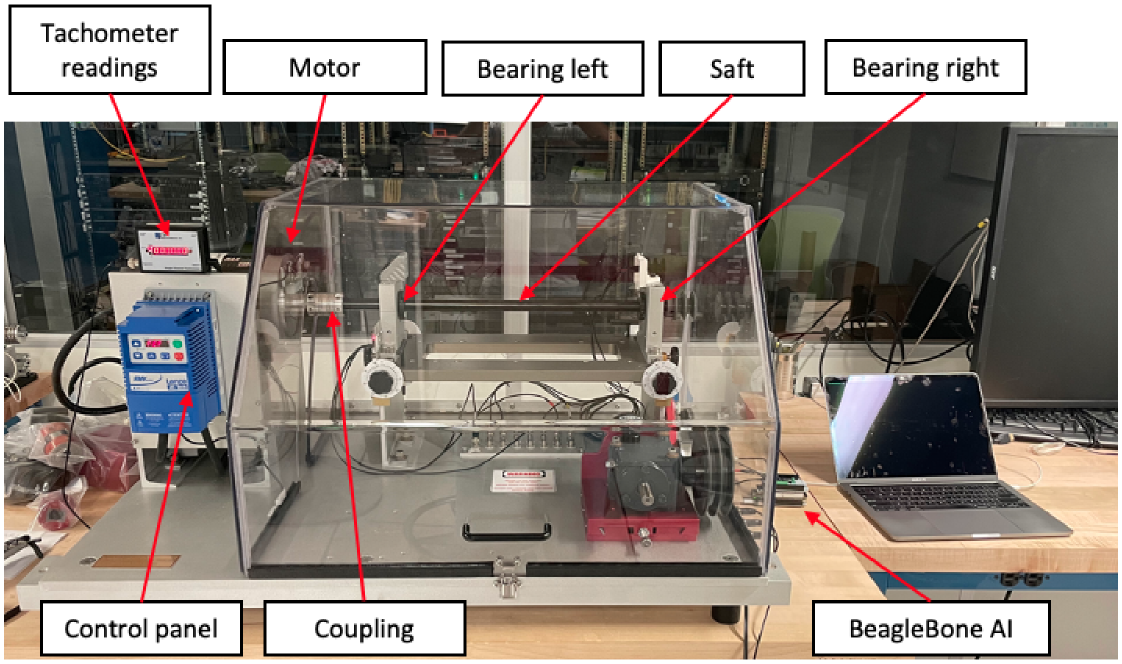


Figure 5.1: Front view of the Machine Fault Simulator

the right of the picture. On the shaft some equipments can be added such as an imbalance loader, where some weights can be added to imbalance the rotating shaft. Instead of using an imbalance loader, one can use a simple 5Kg loader to add some weights to the shaft. With the Machine Fault Simulator are provided multiple sets of bearings in different condition: healthy, outer race defects, inner race defects, and a combination of defects. Unfortunately, those bearings are in close casing, so no visual inspection of these defects can be provided. Figure 5.2 presents the shaft, the healthy and defective bearing, the 5Kg bearing loader and other element of the acquisition setup.

5.1.2 First data acquisition on the Machine Fault Simulator

To validate that the signal acquisition works correctly, an imbalance signal was recorded. An imbalance defect on rotating equipment occurs when the center of rotation is not aligned with the center of mass; this is characterized

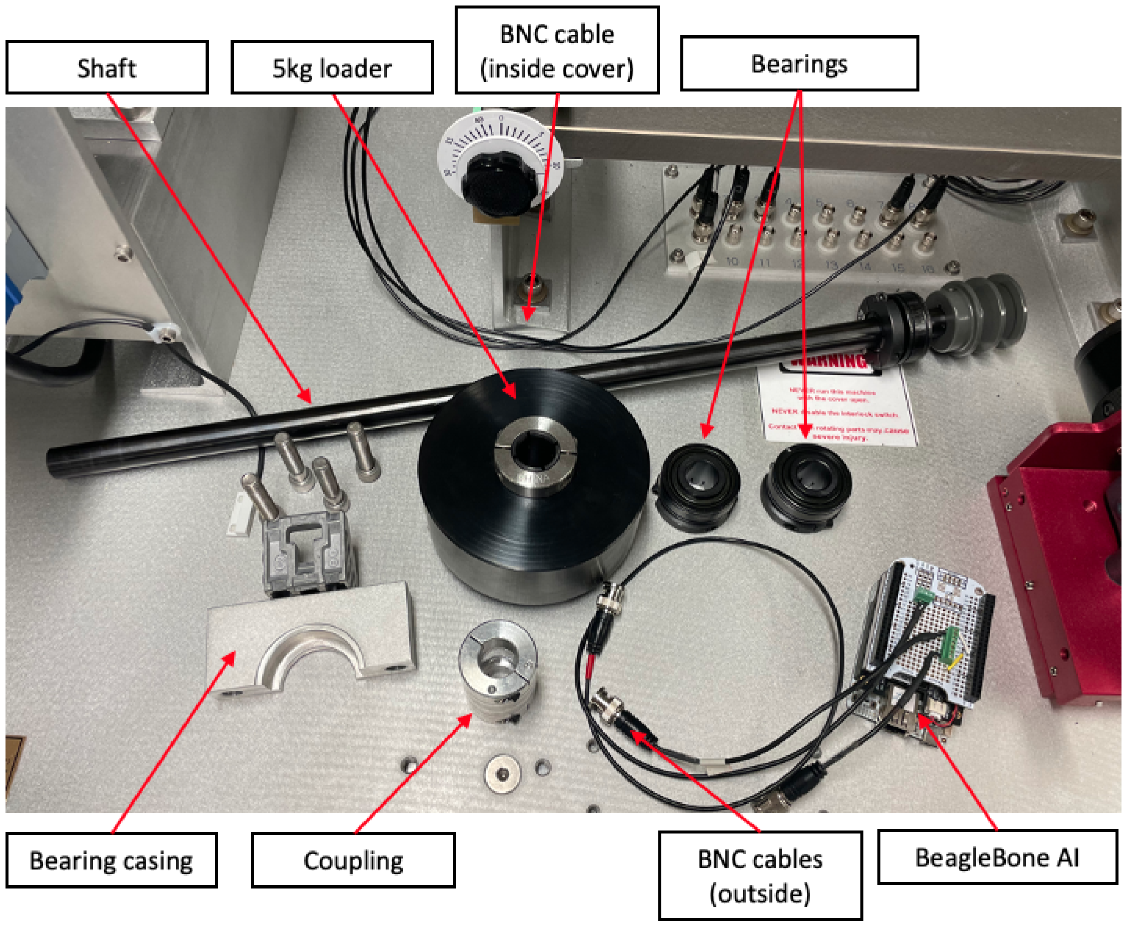


Figure 5.2: Shaft, bearing and bearing loader

by a large peak in the spectrum at the rotational speed of the equipment. Therefore, by unbalancing the shaft of the MFS, we can validate that the data acquisition architecture of chapter 4 works correctly by checking that the largest peak in the spectrum appears at the same frequency as the reading from the tachometer. Some weights were added to the imbalance loader. The Machine Fault Simulator was started, and the rotational speed was set to 15Hz. Additionally, the LA1010 analyzer was connected to the I2C bus to monitor the first transactions between the sensors and the BeagleBone AI.

Perturbation in the I2C Bus

Figure 5.3 shows the first attempt to obtain data. It is clear that there is an error in the I2C transaction. After investigation, there were two reasons for this error.

First, the Machine Fault Simulator Asynchronous Motor was generating electromagnetic fields, which were creating interference along the long unshielded wire going from the KX132 accelerometer to the BeagleBone AI.

Secondly, the Machine Fault Simulator was also polluting the ground of the laboratory and, because the BeagleBone AI only has a digital ground, that was connected to the same ground as the machine fault simulator in the laboratory.

Consequently, the clock line was randomly fluctuating between 0 and 1 which prevents any meaning full communication to go through the BUS. So the transaction in the I2C bus were not successful on first attempt.

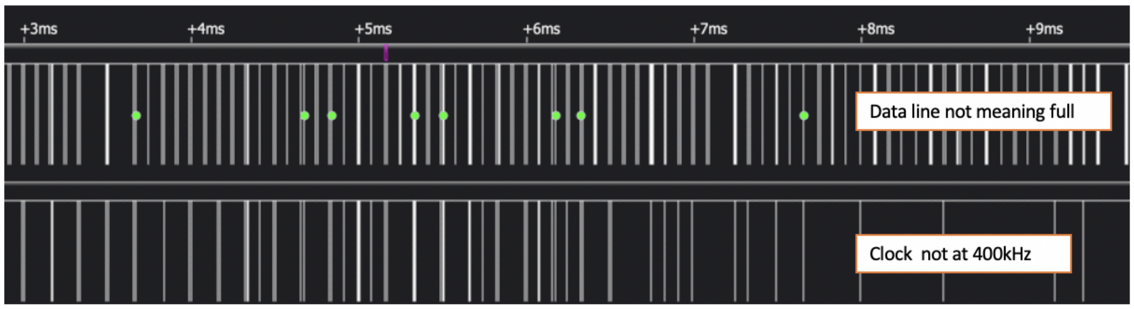


Figure 5.3: Perturbation in the I2C line from the MFS

Improvement of the data acquisition hardware

After identification of the cause of the bug, the following action were taken:

- To reduce the electromagnetic interference in the bus, the wires going from the accelerometer and the BeagleBone AI were shielded and connected to the digital ground of the BeagleBone AI.

- To solve the issues of the polluted ground, the BeagleBone AI was powered with a battery.

Figure 5.4, presents the final acquisition setup, where the BeagleBone is powered with an external battery (MacBook Pro) and with the BNC cables connecting the BeagleBone to the sensor. It should be noted that the sensor axis is not aligned with the radial direction of the bearing. However, experimental measurement showed that it had little impact on the vibration acquisition. Additionally, this mounting point was more repeatable than the other available mounting points in the machine while limiting the self vibration of the sensors mounting plate. Consequently, while not ideal, this solution is preferred as it both, does not hinder the vibration acquisition too much and also reflects the fact that, in industrial application, the perfect placement of the accelerometer is not always achievable.

It should be noted that, in industrial application, shielding the cable is very frequent. Additionally, industrial machines are often powered on a different electrical installation than other equipment, so it would not be required to have the BeagleBone AI run on battery as the ground would not be polluted. Thus, the two solutions proposed do not restrict the use of the system for industrial application.

Validation of data acquisition on the Machine Fault Simulator

The above-mentioned solution solved the issue of data acquisition on the Machine Fault Simulator. Indeed, Figure 5.5 show the spectrum of the vibration signal acquisition at a rotational speed of 15Hz with an imbalanced shaft. The

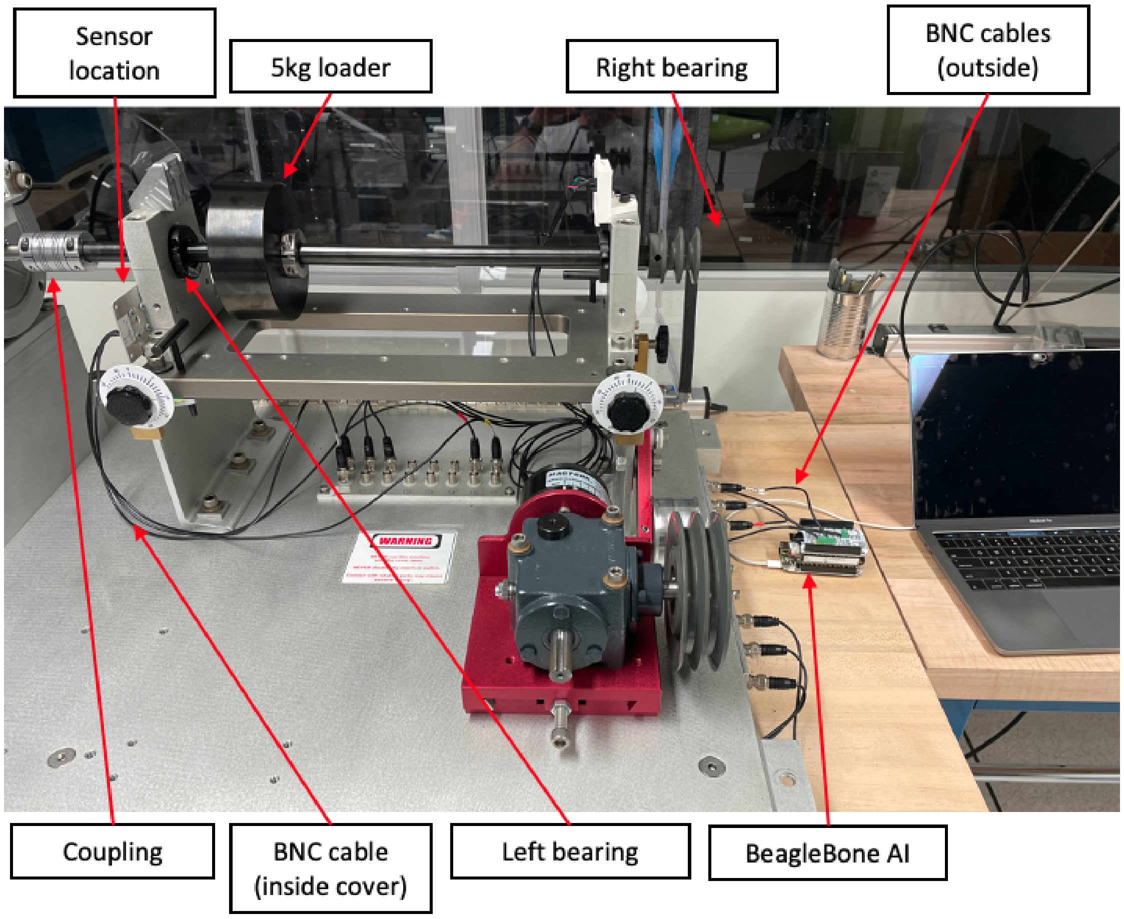


Figure 5.4: Final acquisition setup

predominance of the 15Hz frequency in the spectrum is clear, and it confirms that the acquisition works correctly.

5.1.3 Final data acquisition on the Machine Fault Simulator

As explained in section 5.2.3, we will reduce the study to outer race defect. Data from two different bearing will be acquired: one healthy bearing and one defective bearing. Additionally, two test conditions will be introduced: nothing on the shaft and the addition of a bearing loader on the shaft. Different rotational speed will be tested for those conditions and are summarized in Table 5.1 Additionally, after each change of bearing or installation of the

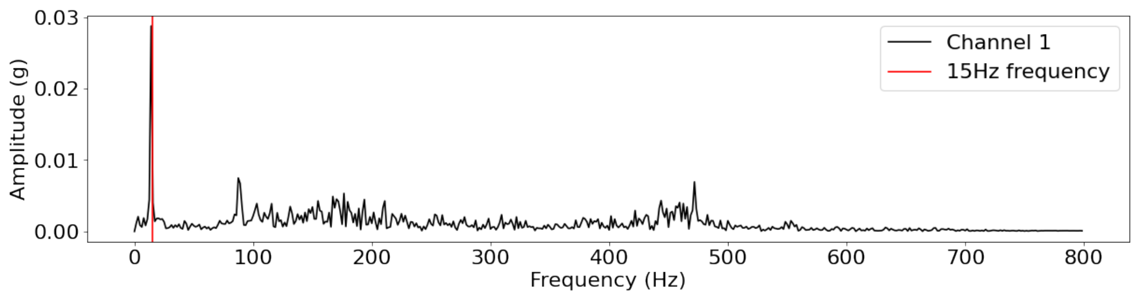


Figure 5.5: Validation of the position of the 1X peak on an imbalance signal

Table 5.1: Data Acquisition conditions for the MFS

RPM \ Condition	**Healthy**	**Outer Race Defect**
10	Load/No Load	Load/No Load
20	Load/No Load	Load/No Load
30	Load/No Load	Load/No Load
40	Load/No Load	Load/No Load

loader in each case the baseline at 0Hz was recorded to validate that there was no change in the new mounting of the sensors. For each of the 16 cases of table 5.1, 500 samples were acquired, this constitutes the dataset for the centralized and federated learning tests. In the following, we present the result in the two different context of learning on the Machine Fault Simulator Dataset

5.1.4 Performance in the centralized learning context

Here, we discuss the result of the centralized learning in the Machine Fault Simulator. To compare the different wavelet transform available, the 5 different models presented in Table 4.5 to 4.10 are trained in a centralized manner on the Machine Fault Simulator Data set. The 500 sample per bearing condition and rotation are placed in the same dataset. Resulting in an initial size of 8000 samples of 1024 data point each. The dataset is then split between a train set and a test set, with a ratio of 30% for the test set and 70% for the

Table 5.2: Accuracy across models and wavelet in a centralized learning context for the Machine Fault Simulator Dataset

Model 0		**Model 1**		**Model 2**	
wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.5	0.9038	rbio3.5	0.8688	rbio3.9	0.8879
bior5.5	0.8929	coif17	0.8663	rbio3.7	0.8854
db29	0.8838	db30	0.8621	bior5.5	0.8808
db24	0.8817	db37	0.8608	coif16	0.8758
coif17	0.8800	coif15	0.8596	db37	0.8738
rbio2.8	0.8783	db38	0.8583	db35	0.8650
db35	0.8758	db32	0.8550	db29	0.8646
db27	0.8758	db29	0.8546	rbio2.8	0.8642
db38	0.8742	rbio3.9	0.8538	db26	0.8600
coif16	0.8729	db31	0.8508	db24	0.8600

Model 3		**Model 4**		**Model 5**	
wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.7	0.8846	rbio3.5	0.8925	coif16	0.8629
coif15	0.8575	bior5.5	0.8871	db37	0.8596
coif17	0.8571	rbio3.9	0.8863	coif17	0.8575
db31	0.8563	db33	0.8704	rbio3.7	0.8558
db36	0.8558	db31	0.8683	db38	0.8554
db38	0.8538	db35	0.8667	db32	0.8517
db29	0.8525	db25	0.8633	db25	0.8492
db34	0.8517	db38	0.8629	db27	0.8475
db33	0.8496	db23	0.8608	db33	0.8471
dmey	0.8488	db27	0.8550	coif12	0.8425

training set. The results of all tested wavelets are presented for the different models on table C.1 to C.6 in the appendix C. Across multiple iteration of training it was seen that taking 100 epochs and a training rate of 0.01 with the Adam optimizer gave the best results. Table 5.2 presents the top 10 accuracy for each of the 5 models with the for a variation of wavelet transform.

However, the results in this case have to be taken with caution as the bearing in the test set in the same as the one for the training set. The main objective here is to obtain a baseline on the Machine Fault simulator in the centralized learning context so that it can be compared with the result in the federated learning context in the same situation.

5.1.5 Performance in the federated learning context

The federated learning library presented in 4.7 is used to train multiple clients in a federated learning context. The Machine Fault Simulator dataset is randomly split across the clients to obtain one non iid dataset per client. Then, the federated learning server is initiated.

Note: Here the concept of overall accuracy is introduced to describe the accuracy of federated learning. It corresponds to the accuracy of models across the dataset constituted by the aggregation of the data from all the clients. This metric is presented more in detail in section 5.3.1.

The performance of each client depends on a number of other parameters than the centralized learning context, such as the number of training rounds, the number of clients per training round and the number of clients. If these parameters are not set properly, it is possible that the clients model do not converge to a minimum of the loss function.

It can be expected that an increase in number of retrained clients per round will increase the final performance of the clients. Similarly, an increase of the number of training rounds can be expected to increase the final accuracy across clients.

However, depending on the situation, an increase of the number of clients can have a different effect. In a simulation context in which the total amount of data stays the same as the number of clients increases, resulting in a diminution of the amount of data per client, it is expected that it will be more difficult for each client to attain a global minimum of the loss function. Consequently, the federated averaging algorithm will not find the global minimum either, resulting in a poor overall performance of the model. If the number of client is increased while maintaining the amount of data per client constant, it is expected that the overall performance of the model will improvement.

However, it is also expected that, if the new client's data is not coherent with the rest of the data, then the performance of the model will not increase and may even decrease. The influence of the parameters should be verified.

In order to reduce the number of variables in this test, we are only considering the wavelet and model which performed the best in the context of centralized learning, that is the rbio3.5, for model 0.

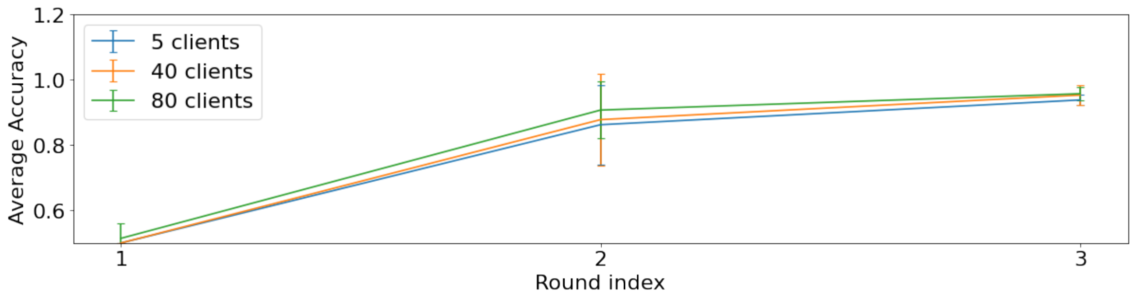
First, we will test the influence of the number of clients over the overall accuracy of the model. Second, we will assess the influence of the number of training round. Finally, we will test the influence of the number of training round.

Influence of the number of clients

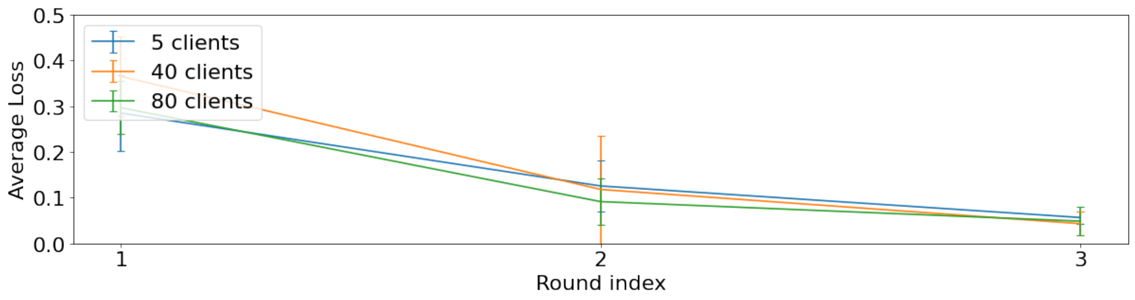
As explained above, two cases should be distinguished, either the total amount of data is kept constant as the number of client is increased, or the total amount of data increases as each new client joins the federated learning process. For the first case, it should be noted that keeping the total amount of data constant is not representative of the reality as each client should be joining the federated learning process with its own data. Thus, the total size of the dataset should be increasing as we add clients.

In the second case, it is important to guarantee that the amount of data per client is random, as in reality, nothing would ensure that the clients have the same amount of data points.

Figure 5.6a present the evolution of the accuracy for 3 consecutive round of federated training for a total number of clients from 3 to 80 by 5 increments.



(a) Accuracy evolution with increase total number of client (average over 10 runs)



(b) Loss evolution with increase total number of client (average over 10 runs)

Figure 5.6: Evolution of the accuracy 5.6a and the loss 5.6b for 5, 40 and 80 clients and three consecutive training rounds.

In figure 5.6a, we can see that the more clients we have the faster the accuracy of the model improves. However, the difference is not significant considering the overlap in the standard deviation.

Additionally, it can be noted that as the number of runs increases, the standard deviation of each of the series is reduced, showing a convergence of the model. Appendix D.3 presents the same trend for 5, 30 and 50; and 15, 35 and 75 total clients in D.4.

Creating the clients datasets: To obtain these results, the machine fault simulator dataset was randomly split across eighty clients, so that we ensure that the amount of data per client is random. Then, each client split it dataset with a ratio of seventy percent for the training set and thirty percent for the test set. This train-test ratio is a usual choice in machine learning. Once each of the 80 clients has obtained a train and a test set, they send a copy of their test set to the federated learning server, which aggregates all these datasets. This is done so that, after a round of federated learning, training the overall accuracy and loss of the model can be obtained by testing the model on this aggregated test set constituted.

Note: In a production mode, the clients would not send their data to the federated server as it would defeat the advantage of data privacy and security that federated learning as over centralized learning. The reason that, in this work, the clients send their data to the server for testing, is that we want to be able to compare the federated learning context to the centralized learning context.

At this point, each of the 80 clients has its own test and train set and the federated learning server has a test set to be able to calculate the accuracy and the loss of the model after the federated averaging of the client's weights.

Federated Learning server with varying number of clients: Then, for a number of clients N varying from 3 to 80, a federated learning server is created with N clients, and the overall loss and accuracy was reported for 10 consecutive training rounds. The reason to stop at 80 clients is that the MFS dataset use is the aggregate of the acquisition of 500 samples of 1024 data points in 16 different cases: load/no load, and rpm varying from 10Hz to 40Hz for a defective bearing or a healthy bearing. This gives 8000 samples, so 80 clients give an average of 100 samples per clients which is enough to ensure training and avoid edge effect of not having enough data points for a client to learn.

Influence of the number of training rounds

Figure 5.7 shows the evolution of the overall loss and accuracy in the test set for an increased amount of clients.

The client's datasets are prepared in the same manner as explained in 5.1.5.

After preparing the client's dataset and the server's test set, a federated learning server is created with the eighty clients and ten rounds of trainings are executed. A training round constitutes one execution of the **do_one_round** of the federated learning API. Between, each of the rounds the server's weights are used for evaluation of the loss and accuracy of the model on the aggregated test set obtained above.

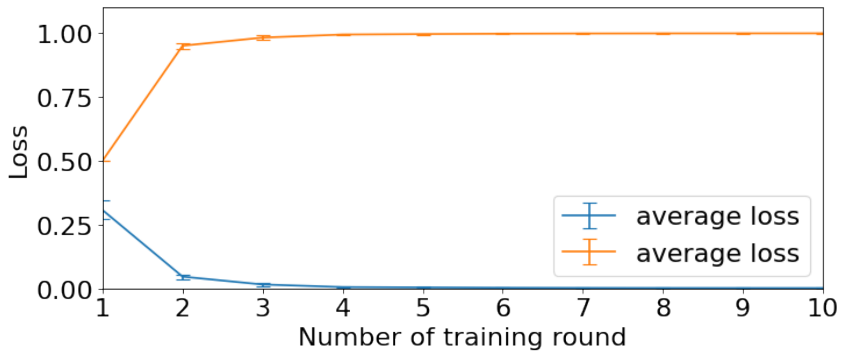


Figure 5.7: Influence of the number of training round for eighty clients

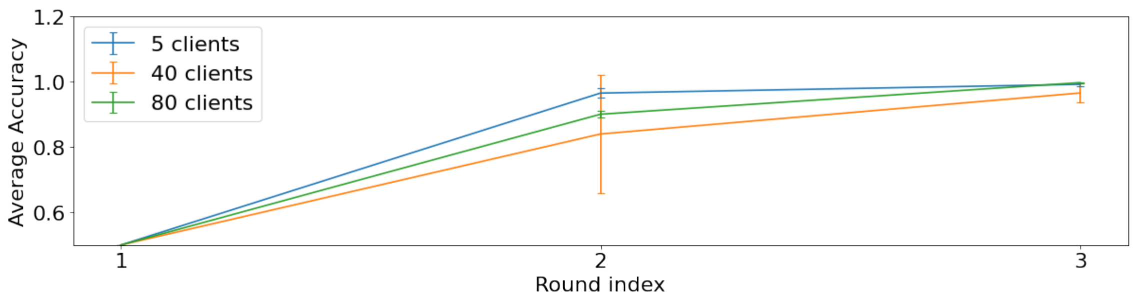
In Figure 5.7, it can be seen that the accuracy of the model is around fifty percent after one round which is as good as random guess as we are considering a binary classification healthy/defect. However, the accuracy quickly increased to 90% for round two and three, which shows that the federated learning approach converges to a minimum of the loss function. This is also visible by the reduction in the standard deviation of across the different runs, which decreases after each round. For round three and after, the accuracy still increases, but the improvement in results is not significant compared to the computational cost. Figure5.7 also illustrates why we limit the analysis to three rounds in 5.1.5 when investigating the influence of the number of clients. In the following, we will also limit the study to three training rounds.

Influence of the number of clients per training round

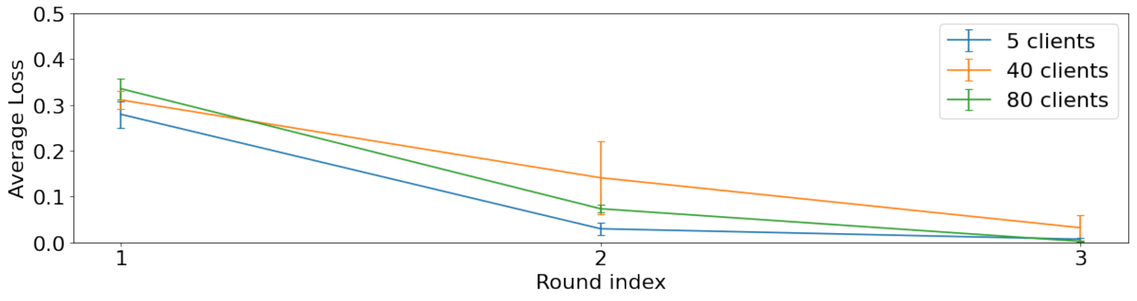
In this section, we study the influence of the number of clients retrained in a pool of clients of a fixed size. This is different from the analysis done in 5.1.5 where it was the size of the pool of clients that was changed. The data is prepared in the same way as presented in 5.1.5. Once the client's dataset are created, a federated server with eighty clients is created for values of the number of training round from three to eighty by increments of five, and three consecutive training rounds are carried out. For each of the training rounds the overall accuracy and loss of the models are recorded. Figure 5.8 presents the results the average loss and accuracy for ten iterations of the above described procedure.

We can see that the accuracy is increasing faster for five clients than for the other case. Then, for forty clients the learning process is slower and also less reliable across the ten runs, (as the standard deviation is greater). Finally, for 80 clients, the learning process is faster again and is the most reliable as the standard deviation is the smallest for every round compared to retraining five or forty clients. Appendix D.2, presents the same graphs for 5, 30 and 50 retrained clients in D.3 and 15, 35 and 75 retrained clients in D.4, and the same trend can be seen.

Figure 5.9, generated with the same data, illustrates this point, it shows the evolution of the average accuracy and loss as a function of the number of retrained clients. Considering two consecutive rounds were we can see that, with an increasing number of retrained clients per round, the accuracy, ini-



(a) Accuracy evolution with increase number of retrained clients



(b) Loss evolution with increase number of retrained clients

Figure 5.8: Evolution of the accuracy 5.8a and the loss 5.8b for 5, 40 and 80 clients and three consecutive training rounds.

tially above 90% in round two, drops to 70% percent when we retrain 35 clients, and increases again when we keep increasing the number of retrained clients per round.

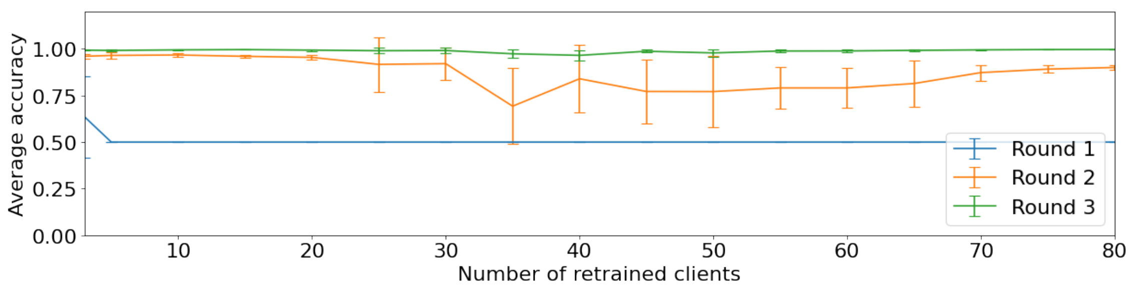


Figure 5.9: Evolution of the average accuracy for the different numbers of retrained client per round

Table 5.3: Confusion Matrix for the Peak Finding approach for the MFS Dataset

actual healthy	3047	953
actual defect	1032	2968
	predicted healthy	**predicted defect**

5.1.6 Performance of the peak finding approach

The peak finding approach developed in [44] is used in order to identify defects in the Machine Fault Dataset and to compare with the proposed solution in the centralized learning and federated learning contexts. This approach identifies the major peak in the spectrum and tries to identify outer race defect signatures in it. Table 5.3 presents the results of this peak finding approach on the Machine Fault Simulator. It can be seen that there is a large amount of healthy signals that were classified as defective. These results are being compared with the results from the other methods in the next sections.

Note: Contrary to the next dataset, in which the trend analysis of the Kurtosis of the signal can be applied; here, the change of status of the bearing in the dataset was simulated by replacing the healthy bearing by a defective bearing of the same model. Consequently, there is no trend information to track on the Machine Fault Simulator.

In this section, we have validated the data acquisition on the Machine Fault simulator as well as the wavelet plus machine learning approach in both the centralized learning context and the federated learning context. It has been shown that in both cases the results are close to 90% of accuracy. However, this dataset is generated from a test bench and the signal in the train and test set are very similar as they come from the same bearing, this explains the very high accuracy obtained in this case.

Finally, we have presented the results of a peak finding approach applied to this same dataset. This approach is not based on machine learning thus; it should not be affected by the afford-mentioned limitation in the dataset. In order to further validate these results, another dataset will be used in the next section.

First, the NASA Dataset is introduced and the selection process for the bearings of this dataset is explained. Second, the neural network and wavelet approach is tested in both the centralized context and federated context. Finally, the results of the peak finding approach are presented as well as the results from the simple application of the eight of the Nelson Rules to the evolution of the Kurtosis of the signal.

5.2 NASA data-set

5.2.1 Introduction of the Dataset

The NASA Bearing dataset is part of the Prognostics Data Repository, where datasets for various assets are hosted online; those datasets are only hosted by NASA and donated by other organizations. The Bearing Dataset was provided by the Center for Intelligent Maintenance Systems (IMS), University of Cincinnati. It has been downloaded over 57 thousand times and has been used in over 60 different papers (as of Spring 2021).

The NASA Bearing Data Set consists of 3 run-to-failure tests. For each test, four bearings are positioned on the same shaft and submitted to a radial load, and spun with a motor. Each bearing is equipped with accelerometers; for the first experiment, Qiu et al. put one accelerometer on both the X and Y axis; and for the second to last experiment, only one accelerometer is placed on each bearing. The accelerometers used are PCB 353B33 High Sensitivity Quartz ICP [38]. Thermocouples are also attached to the bearings, but the thermal data are not provided in the Prognostics Data Repository. Figure 5.10 form [38] present the setup of the experiment.

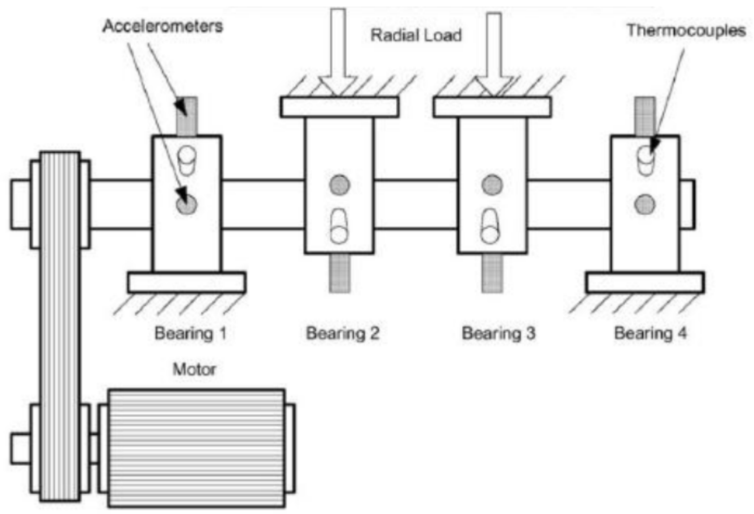


Figure 5.10: Bearing test rig and sensor placement for NASA Dataset from [38].

The data acquisition is carried out every 10 minutes at a frequency of 20kHz for 1 second resulting in 20.480 data points per acquisition file. The experiments were stopped when a defect appeared on one of the four bearings:

Test 1: was stopped after 2,156 samples and an inner-race defect as well a rolling element defect were respectively observed on bearing 3 and 4.

Test 2: was stopped much earlier after 984 samples with an outer race on bearing 1.

Test 3: was stopped at 4,448 samples with an outer race on bearing 3.

5.2.2 A first analysis of the dataset and preprocessing

After decompressing the zip file downloaded from the repository, three folders are obtained, one per experiment. The folder contains 2156, 984, and 6323 CSV files for tests 1 to 3, respectively. The difference in the number of files and the expected number of samples for test 3 is not explained. Documentation's accuracy [38] with the data is validated by plotting the evolution of Kurtosis for each bearing and each experiment.

For Test 1

Figure 5.13 show the Kurtosis trend increase for bearing 3 and 4 starting around sample number 1,500 for bearing 4 and around sample 1,700 for bearing 3. There is no increase in trend during test 3 for bearing 1 and 2, which conform with the documentation from [38].

For Test 2

Figure 5.11 is obtained in the same manner as for test 1 (Figure 5.13). Again, the observations are conformed to the documentation, and the defect is clear in Bearing 1. It should be noted that there is a peak in the Kurtosis value of Bearings 2 and 4 at the end of the acquisition, that could be explained by vibration peak in the vibration of Bearing 1. As the 4 Bearings are connected by the shaft, an important increase vibration in one bearing could propagate along the shaft and be detected by the sensors on the other bearings.

However, the high Kurtosis values in Bearing 3 starting from the beginning of the test cannot be explained.

For Test 3

Figure 5.12 present the longest test of the dataset, Bearing 3 is experiencing a defect as expected. However, the defect region is very little compared to the total duration of the test. Thus, even if this test gives more data, it really

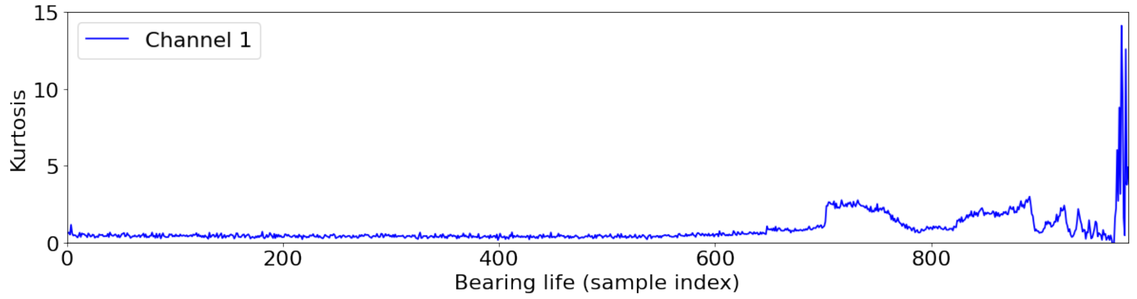


Figure 5.11: Evolution of the Kurtosis over time for the second test of the NASA Dataset for bearing number 1.

provides a large amount of healthy signal but does not provide many defective signals.

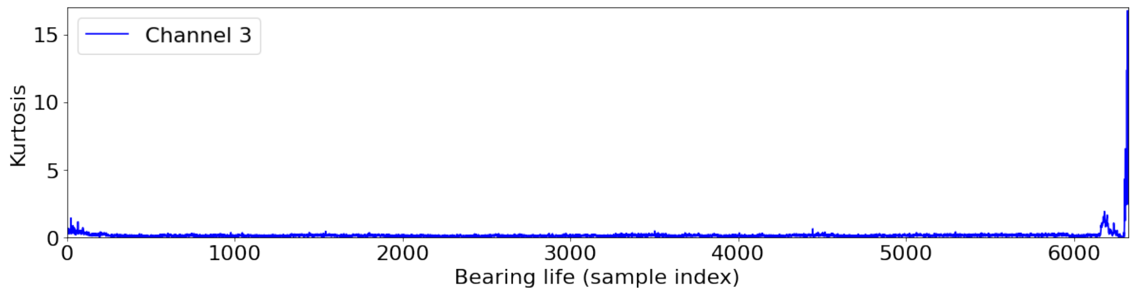
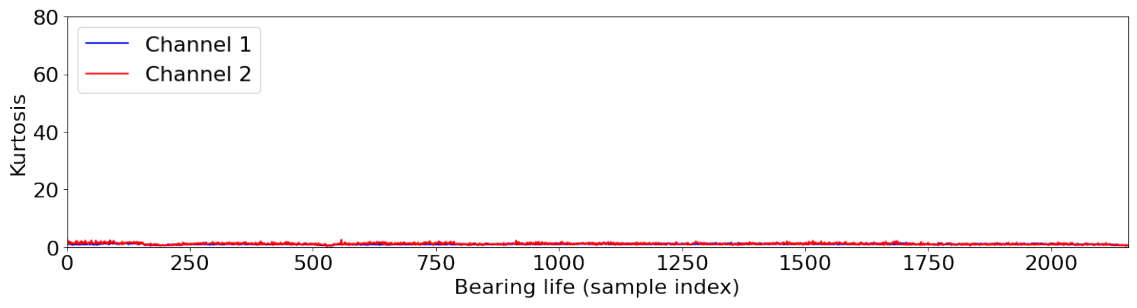


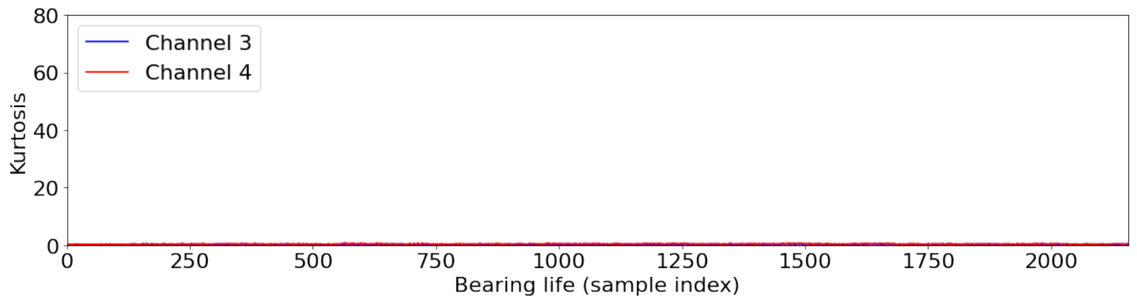
Figure 5.12: Evolution of the Kurtosis over time for the third test of the NASA Dataset for bearing number 3.

Table 5.4: Summary of the defect observed on the dataset

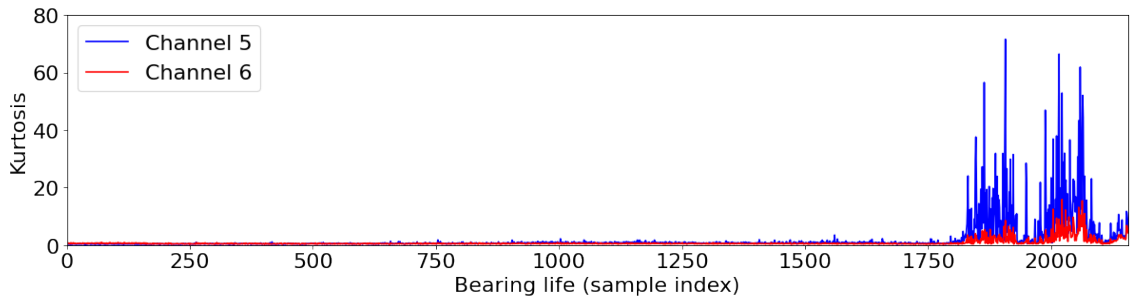
Test #	**Bearing #**	**Fault Type**
Test 1	Bearing 1	No defect
-	Bearing 2	No defect
-	Bearing 3	Inner race defect
-	Bearing 4	Rolling element defect
Test 2	Bearing 1	Outer race defect
-	Bearing 2	No defect
-	Bearing 3	No defect
-	Bearing 4	No defect
Test 3	Bearing 1	No defect
-	Bearing 2	No defect
-	Bearing 3	Outer race defect
-	Bearing 4	No defect



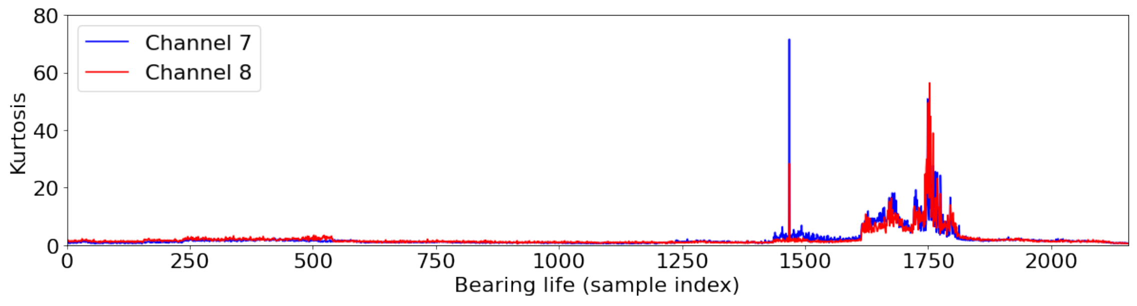
(a)



(b)



(c)



(d)

Figure 5.13: Evolution of the Kurtosis over time for the first test of the NASA Dataset. E.1a channels 1 and 2, for bearing 1 where no defect appeared; E.1b channels 3 and 4, for bearing 2 where no defect appeared; E.1c channels 5 and 6, for bearing 3 where an inner race defect appeared; E.1d channels 3 and 8, for bearing 4 where an outer race defect appeared.

5.2.3 Selection of the data in the dataset

Out of the dataset 4 different faults can be used: 2 outer race defects, one inner race defect and one rolling element defect. To be useful, the machine learning algorithm should be able to recognize the defect on another bearing than the one on which it had been trained. Consequently, we want to be able to compare the developed method across two different bearings of the NASA Dataset, so we will restrict the study to outer race defect. Indeed, on test 2, and 3 of the dataset at least one of the bearing failed with an outer race defect, thus, we could train on the test 2 and validate the approach on test 3. Test 1 only presents inner race and a rolling element defect, so will not be useful in this work. To be able to train an ANN in supervised learning, the data needs to be labeled, and in this case, it needs to obtain the region of the test 2 and 3 where the bearing can be considered to be healthy and where we consider the defect to be present. On this matter, we will make a first selection of the sample index visual on the Kurtosis graphs along the life of the bearing; this is then refined using a visual inspection of the spectrum of the signal.

Selection of the defect and healthy region for test 2 bearing 1

As mentioned earlier, during test 2, bearing 1 experienced an outer race defect. The first selection of this outer race can be seen in Figure 5.14 This selection was refined by analyzing the spectrum on each side of the limit. In the initial selection the spectrum on each side of the limits are presented on Figure 5.15. It can be seen that on the initial region where the bearing is supposed to be healthy, some harmonics from the defect already starts to appear. Moreover, the defect is already very present in the spectrum for the region where, the defect is supposed to only start to wear out. Consequently, these

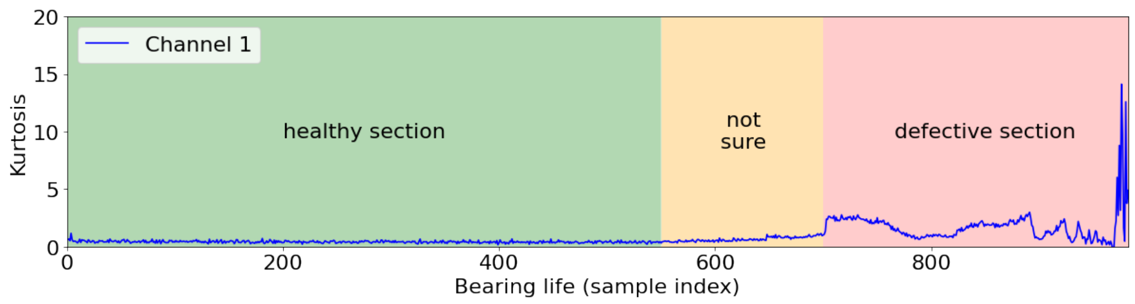
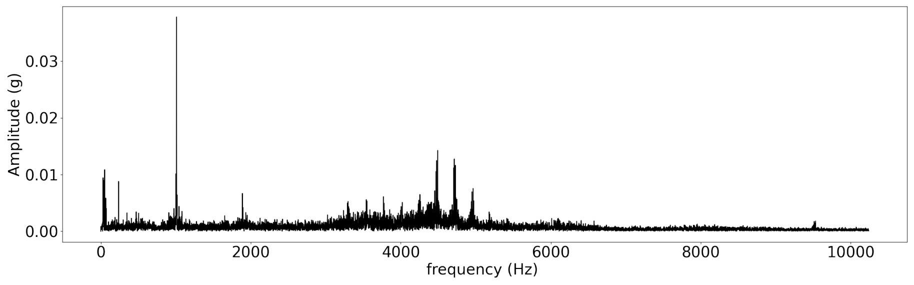
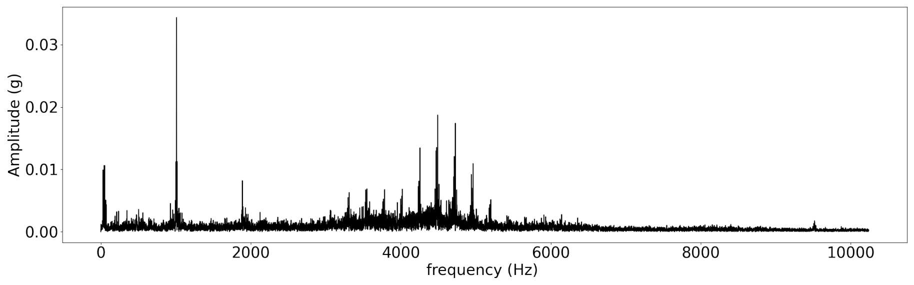


Figure 5.14: The initial selection of the defect zone for bearing 1 of test 2

regions are refined by analyzing the spectrum for all samples around this first guess. Figure 5.16 shows the spectrum of the last healthy signal and the first defective sample. There is still a region where the classification is unsure and this is normal as the degradation of the bearing is a continuous process.



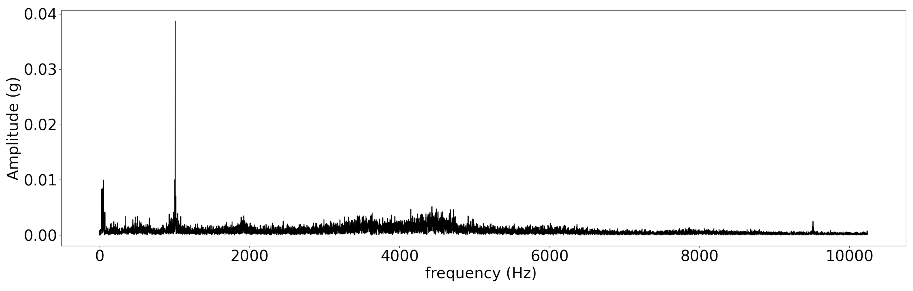
(a)



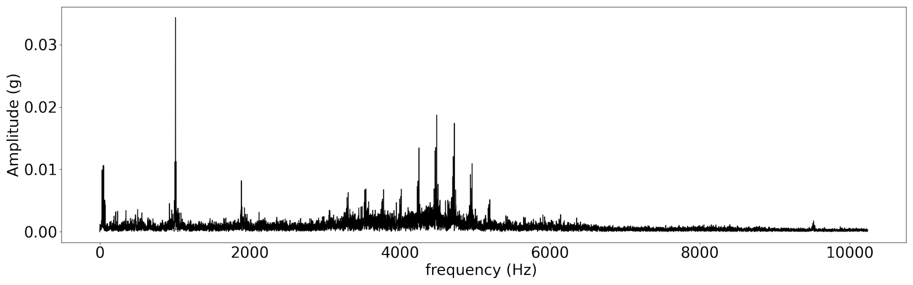
(b)

Figure 5.15: Spectrum for test 2 bearing 1 at the end of the healthy zone (5.15a) and beginning of the defective zone (5.15b)

Figure 5.17 shows the final classification for the test 2 bearing 1. Compared to the initial classification, the healthy region has been reduced, and the defective region has increased.



(a)



(b)

Figure 5.16: Spectrum for test 2 bearing 1 at the end of the healthy zone (5.16a) and beginning of the defective zone (5.16b)

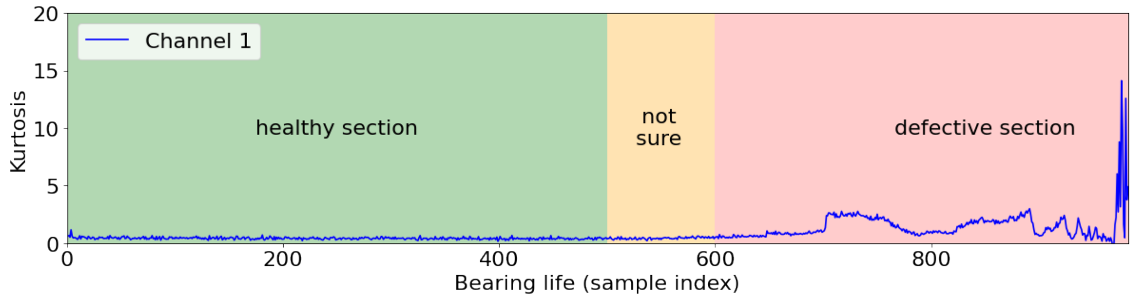


Figure 5.17: The final selection of the defect zone for bearing 1 of test 2

Selection of the defect and healthy region for test 3 bearing 3

Similarly to test 2, we first select the region visually on the Kurtosis, and we then refine this selection by analyzing the spectrum of the signal around this selection. This extra step is even more critical for test 3 as the bearing wears out very abruptly.

The spectrum of the last healthy signal and of the first defective signal for this initial selection are presented in Figure 5.19

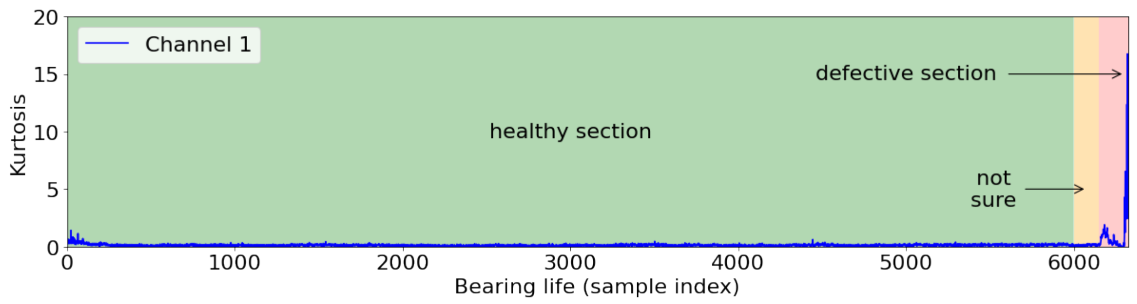


Figure 5.18: The initial selection of the defect zone for bearing 3 of test 3

In this case, the redefinition of the region impacts more the defective region were the defect was not really present in the first selection. Figure 5.20 shows the spectrum of the final selection for the last healthy signal and the first defective signal.

Finally, Figure 5.21 show the selection for the defect region for bearing 3 test 3. The change in terms of sample index is limited, the healthy region's index was increased by 1 samples and the defective region's index was increased by 13 samples (reduction of the number of 13).

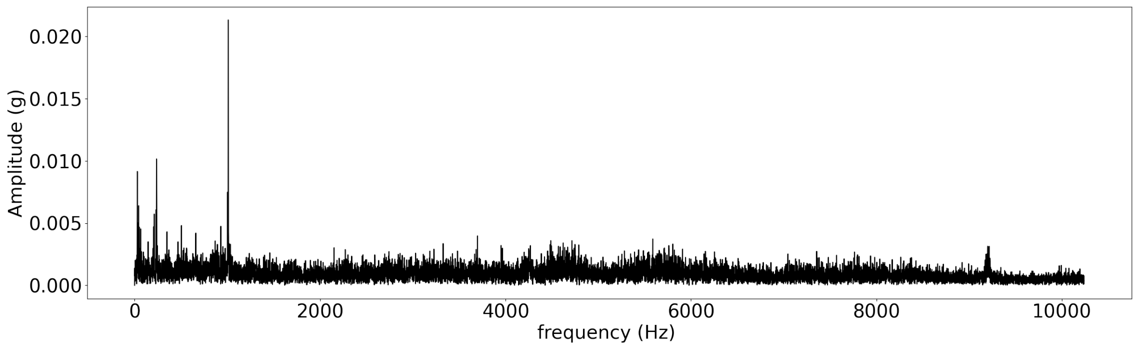
Table 5.5 presents the limits of the final healthy and defective region for both Test 2 and test 3 in terms of sample index.

Table 5.5: Healthy and defective region for test 2 and 3

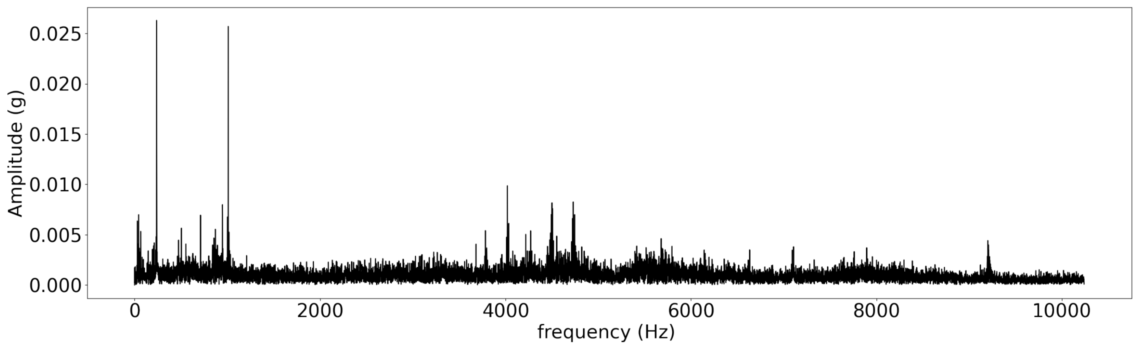
Test #	**2**	**3**
Bearing #	Bearing 1	Bearing 3
Fault Type	Outer race defect	Outer race defect
End of the Healthy region (sample index)	500	6001
Beginning of the defect region (sample index)	600	6163

5.2.4 Down sampling

The NASA Dataset is sampled at 20kHz, however, the other dataset of this work, Machine Fault Dataset, is sampled a 1.6kHz. Consequently, to be able



(a)



(b)

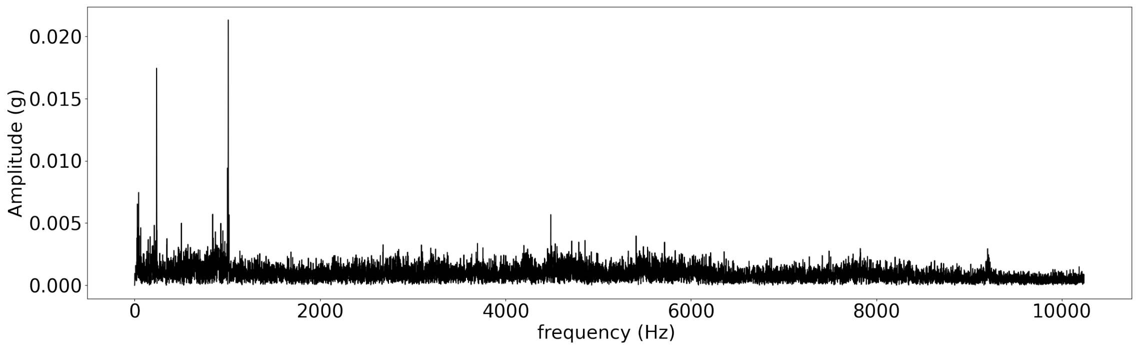
Figure 5.19: Spectrum for test 3 bearing 3 at the end of the healthy zone (5.19a) and beginning of the defective zone (5.19b)

to compare the results on these two datasets, we will down-sample the NASA Dataset down to the around sampling rate. As we have 20480 data points per sample in the NASA Dataset, we can obtain 16 sub-samples for each initial sample, while maintaining at least 1kHz of sampling frequency. First, the initial sample is filtered used a Butherworth filter represented in Figure 5.22.

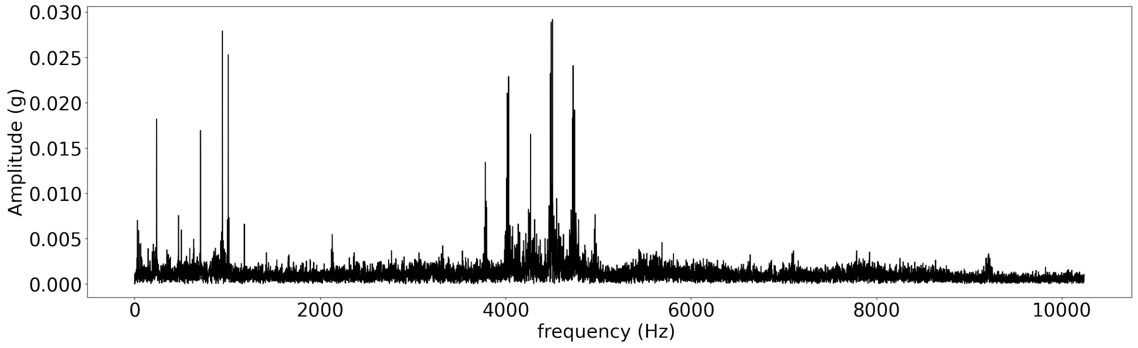
The result of the application of the low-pass filter to spectrum in the defect region of test 2 is presented in figure 5.23. It can be seen that there is no high frequency components that would lead to some kind of aliasing.

5.2.5 Performance in the centralized learning context

The same centralized learning context as for the Machine Fault Simulator: the different Discrete Wavelet Transform and the same models are used here.



(a)



(b)

Figure 5.20: Spectrum for test 3 bearing 3 at the end of the healthy zone (5.20a) and beginning of the defective zone (5.20b)

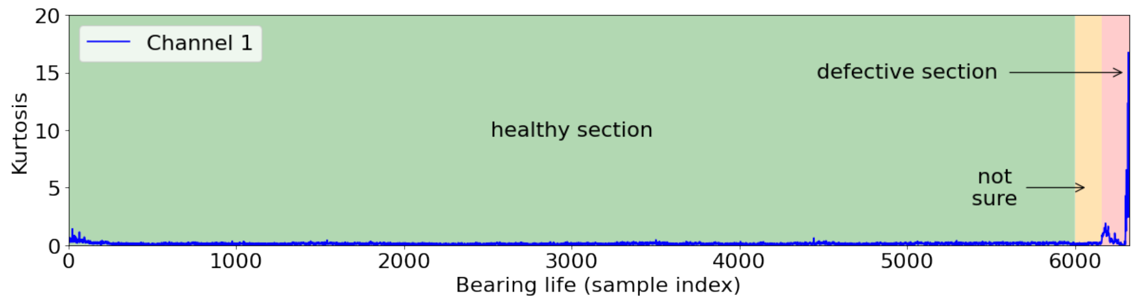


Figure 5.21: The final selection of the defect zone for bearing 3 of test 3

However, this test is performed on the down-sampled NASA Dataset. In order to introduce, more variety, the test number 2 of the NASA Bearing Dataset is used as the train set, and the test number 3 is used as the test set. The advantage of doing so is that the bearing used for training is different from the bearing used for the test set. Consequently, there are 16000 Samples of 1024 data points each. Table 5.6 extracted from the results presented in

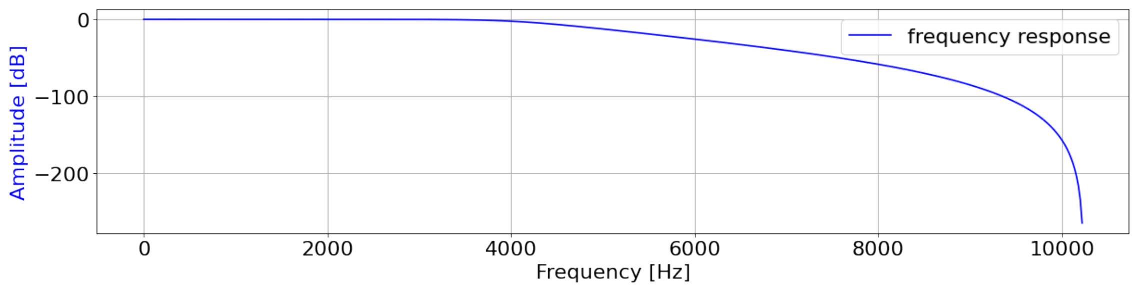
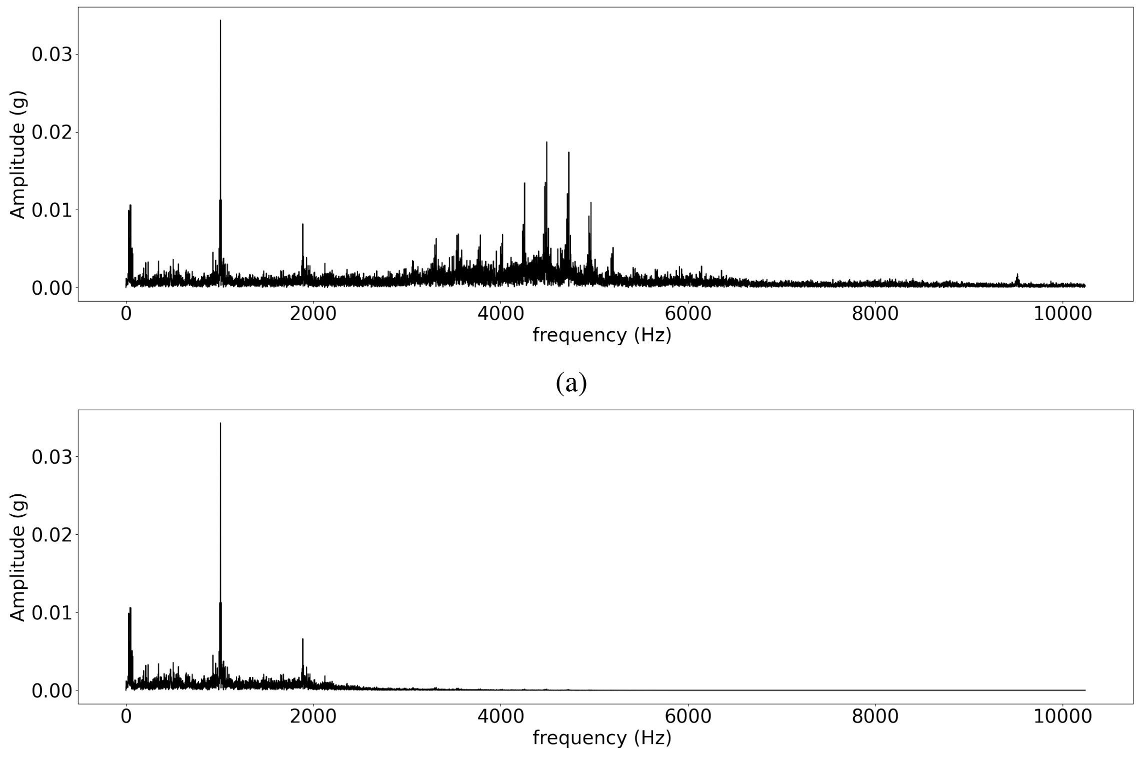


Figure 5.22: Low pass filter frequency response



(a)

(b)

Figure 5.23: Spectrum of Bearing 1 test 2 in the defective region before filtering (5.23a) and after filtering (5.23b)

appendix F shows the top 10 accuracy for each of the models for different wavelet transforms.

The next section present the result in the federated learning context where the results are going to be compared against the centralized learning results.

Table 5.6: Accuracy across models and wavelet in a centralized learning context for the NASA Dataset

Model 0		**Model 1**		**Model 2**	
Wavelet	Accuracy	Wavelet	Accuracy	Wavelet	Accuracy
coif14	0.78125	bior3.3	0.729882813	bior3.5	0.665625
bior3.9	0.7390625	rbio3.5	0.675	bior3.9	0.665429688
bior3.3	0.73671875	bior3.7	0.66875	rbio3.7	0.65625
rbio3.1	0.68359375	bior3.9	0.665429688	coif12	0.653125
bior3.7	0.65	rbio3.3	0.6625	rbio3.9	0.653125
rbio3.5	0.646875	rbio3.1	0.652929688	coif14	0.65
rbio3.3	0.6375	coif15	0.65	rbio3.5	0.646875
db36	0.634375	rbio3.7	0.646875	db12	0.637695313
coif15	0.625	coif17	0.646875	rbio3.1	0.6375

Model 3		**Model 4**		**Model 5**	
Wavelet	Accuracy	Wavelet	Accuracy	Wavelet	Accuracy
rbio3.1	0.822265625	bior3.3	0.848632813	bior3.3	0.74824219
bior3.3	0.815039063	bior3.9	0.75859375	bior3.5	0.678125
bior2.8	0.749804688	rbio3.1	0.755273438	rbio3.9	0.66875
coif17	0.740625	coif12	0.73125	coif15	0.6625
bior3.5	0.740625	coif17	0.721875	db6	0.65917969
bior3.7	0.737304688	coif16	0.71875	bior3.7	0.65625
bior2.4	0.737109375	coif15	0.7125	rbio3.3	0.65
coif12	0.725	coif13	0.7125	db38	0.64375
coif10	0.725	rbio3.5	0.7125	rbio3.1	0.64375
db27	0.721875	bior3.5	0.709375	rbio3.7	0.6375

5.2.6 Performance in the federated learning context

In this section, we will test the federated learning approach on the NASA Dataset. The same federated learning library is used as for the test on the Machine Fault Simulator Dataset. However, as for the previous test on the NASA dataset, the test and train set will be obtained from two different bearings: bearing one of test two will be used as the train set and bearing number three of test three will be used as the test set. Consequently, we can expect the learning process to be harder in this situation that in the test of federated learning on the Machine Fault Simulator where the same bearing was used to generate the train set and the test set.

Creating the clients dataset and the test As mentioned above, the bearing 2 of test 2 will be used to create the train set. The data is labelled as presented in section 5.2.3 and randomly distributed across the different clients. Once all the clients are initialized with their data, the federated server's own dataset is updated with the data from the bearing one of test three. The server processes the wavelet transform of this dataset with the same parameters as the one provided to the clients.

In the following, we present the influence of three parameters of the federated learning approach over the performance of the federated learning. We proceed, in the same order as for the Machine Fault Simulator, first, we study the influence of the number of clients in the federated context, then the influence of the number of training rounds. Finally, we test the influence of the number of client per training round.

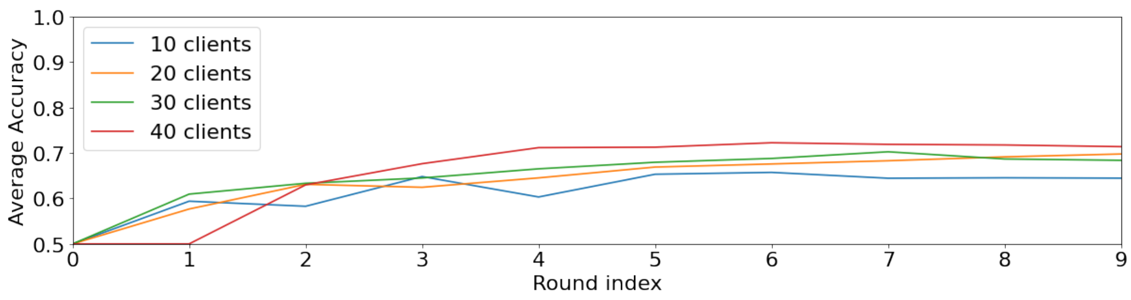
Influence of the number of clients

In Figure 5.26, we present the evolution of the loss and the accuracy for a changing total number of clients. To obtain this, the NASA dataset was split across 40 clients and the federated learning context was regenerated for a number of clients from 10 to 40 by 10 increments. Indeed, with the initial amount of data in the NASA dataset being fixed, we need to ensure that each client has about the same number of data points no matter what is the total number of clients for which we are testing.

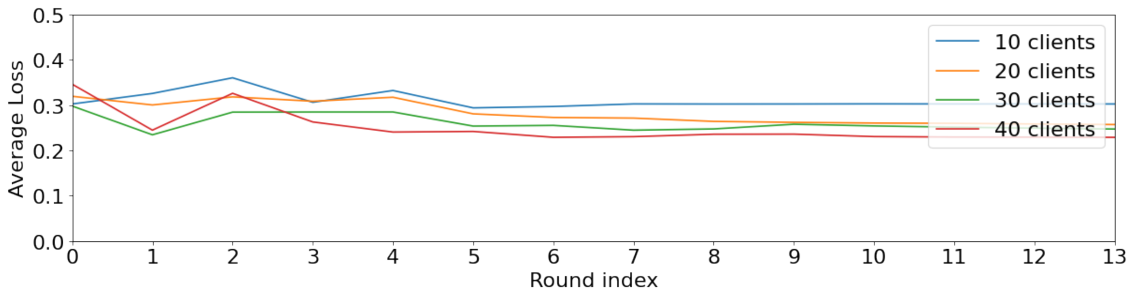
Consequently, it would not be adequate to split the dataset by the number of clients in the process as there would be four times more data points for ten clients than for forty clients. This is the reason why we start by splitting the dataset by forty and then distribute those slices to the clients. Thus, there is four times less data in the federated context when running for ten clients than when running for forty clients, which represents the reality where more clients leads to more data. Moreover instead of keeping the number of retrained client constant to ten retrained clients per round, no matter the context, it is important to keep the ratio of retrained clients to total number of client constant. Indeed, keeping the number of retrained clients constant would result, for a context with 10 clients, to 100% of the clients are retrained for each round, but only 25% of the clients are retrained for the context with forty clients.

Therefore, the experiment was carried out while keeping the ratio of retrained clients to 50% of the total number of clients. The results of this experiment are presented in Figure 5.24

First, it is obvious that the results are not as good as for the Machine Fault Simulator. As the objective of Figure 5.26 is to show the influence of the



(a) Accuracy evolution with increase total number of client (average over 10 runs)



(b) Loss evolution with increase total number of client (average over 10 runs)

Figure 5.24: Evolution of the accuracy 5.6a and the loss 5.6b for 10, 20, 30 and 40 clients and fourteen consecutive training rounds.

number of clients over the accuracy, the current parameters of the federated learning context are not optimized and the accuracy of the method is not optimal. It can be seen that asymptotically, forty clients in the federated context perform much better than 10 clients with an accuracy of 72% compared to just above 60% for ten clients. The test with twenty and thirty clients are very close to the results for forty clients and the difference of accuracy of less that 2% is not really significant. The fact that the runs with 10 clients perform less than the 3 other sets of runs could be explained by the fact that with only 10 clients there is not enough data in the federated context to be able to learn.

Finally, the influence of the number of clients in the federated learning context is not significant as long as there is enough clients for the federated algorithm to obtain a minimum of the loss function.

Influence of the number of training round

In order to study the influence of the number of training rounds over the performance of the federated learning approach, fourteen consecutive training round are executed in the following context. The dataset is split into 40 different clients and 5 of them are retrained. The model's weight is obtained after federated averaging are then used for prediction on the first bearing of test three of the NASA dataset, as described in 5.2.6. The choice of retraining five clients in context with a pool of 40 clients is motivated by the results obtained in Figure 5.26a. Figure 5.25 presents the evolution of the loss and the accuracy with the number of training rounds. It can be seen that the accu-

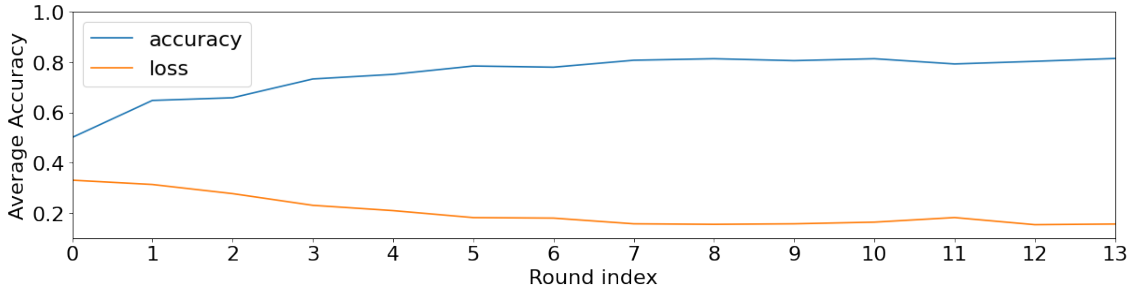
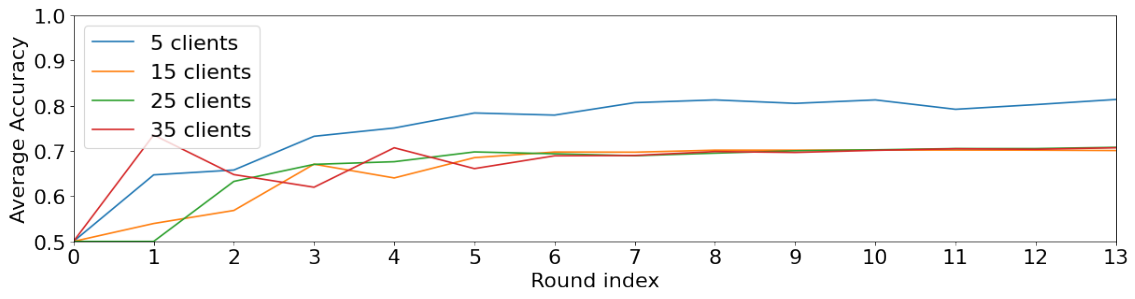


Figure 5.25: NASA Influence of the number of training round for forty clients

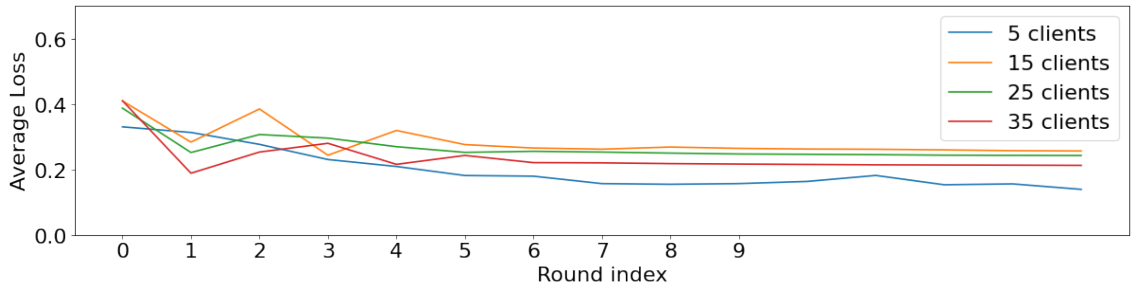
racy on the test set initially of 50%, slowly improve up to 80% in six rounds, which is much slower than the results obtained on the Machine Fault Simulator dataset. The maximum accuracy is found to be 83% for this setting.

Influence of the number of clients per training round

In this case, we study the influence of the number of clients per training round. That is, for a fixed number of clients in the federated learning context, we will change the number of clients that are requested to retrain their model and to send new weights to the federated server. In this setting, 40 clients were generated from the NASA dataset and thirteen training rounds were executed with a number of retrained clients changing from 5 to 35 with 5 increments.



(a) NASA Accuracy evolution with increase number of retrained clients



(b) NASA Loss evolution with increase number of retrained clients

Figure 5.26: Evolution of the accuracy 5.8a and the loss 5.8b for 5, 15, 25, and 35 clients and fourteen consecutive training rounds.

It can be seen that the setting with only five retrained clients per round is performing the best. This is a counterintuitive result as, it could be expected that the more clients are retrained the better the algorithm performs. However, an explanation to that could be that, with an increase in the number of clients per training round there is also an increase in the variation of the weights received by the server.

As in the federated learning algorithm, the server calculates the new weights between each round with a mean, it may become harder to find a global minimum of the loss function with more clients sending their weights.

These observations are similar to the one done on the Machine Fault Simulator dataset, where the best accuracy was obtained with the least amount of retrained clients. The main difference comes from the fact that, in the case of the Machine Fault Simulator dataset, the accuracy re-increases when the number of retrained client become close to the total number of client in the context. Noting that, in the case of the Machine Fault Simulator, the test set comes from the same bearing as the training set, the increase in accuracy could be due to overfitting. Now that the results on the NASA Dataset have been presented on the centralized learning and federated learning context, two other method are going to be tested. First, the same peak finding approach as tested on the Machine Fault Simulator is used on this dataset. Second, a trend analysis approach based on the Nelson Rules is used to detect anomalies in the Kurtosis of the vibration signal. Note that this last approach could not be tested on the Machine Fault Simulator as it can only be used on degradation trend of a bearing.

5.2.7 Performance of the peak finding approach

As for the Machine Fault Simulator Dataset, we apply a peak finding approach that iterates over the spectrum to predominant peaks to identify harmonic families representative of an outer race defect. The dataset used is the results from the region identified in bearing three of test three of the NASA dataset. Table 5.7 shows the confusion matrix resulting from the classification. Compared to the results obtained with this same approach on the NASA dataset, we can see that the accuracy has decreased. This is due to the fact that the NASA dataset contains signals representative of a defect (and labeled as such) were the defect in the outer race is still quite small.

The next section presents the results of outlier detection on the trend of the Kurtosis of this same bearing. Those two methods are presented to be used

Table 5.7: Confusion Matrix for the Peak Finding approach for the NASA Dataset

actual healthy	2412	148
actual defect	1635	925
	predicted healthy	**predicted defective**

as reference compared to the developed methods.

5.2.8 Trend analysis approach

To analyze the Kurtosis evolution over the life of the bearing 3 of test 3, Table 5.8 show the results of classification of the height following Nelson Rules:

Rule 1: One point is more than 3 standard deviations from the mean (outlier)

Rule 2: Nine (or more) points in a row are on the same side of the mean (shift)

Rule 3: Six (or more) points in a row are continually increasing (or decreasing) (trend)

Rule 4: Fourteen (or more) points in a row alternate in direction, increasing then decreasing (bimodal, 2 or more factors in data set)

Rule 5: Two (or three) out of three points in a row are more than 2 standard deviations from the mean in the same direction (shift)

Rule 6: Four (or five) out of five points in a row are more than 1 standard deviation from the mean in the same direction (shift or trend)

Rule 7: Fifteen points in a row are all within 1 standard deviation of the mean on either side of the mean (reduced variation or measurement issue)

Rule 8: Eight points in a row exist with none within 1 standard deviation of the mean and the points are in both directions from the mean (bimodal,

Table 5.8: Precision, Recall and Accuracy for 8 different Nelson Rules

	Precision	**Recall**	**Accuracy**
Rule1	1.00	0.84	0.92
Rule2	0.17	0.37	0.20
Rule3	1.00	0.03	0.68
Rule4	NA	0.00	0.67
Rule5	1.00	0.23	0.71
Rule6	1.00	0.23	0.71
Rule7	0.27	0.20	0.56
Rule8	1.00	0.07	0.69

Table 5.9: Confusion Matrix for the Trend approach for the NASA Dataset

actual healthy	2560	0
actual defect	396	2164
	predicted healthy	**predicted healthy**

2 or more factors in data set)

Table 5.8 shows that the outlier detecting with Rule 1 is working extremely well when on this Dataset. The results of the classification are presented in the confusion matrix in Table 5.9.

In this section, we have presented the results of four different approaches on the NASA Dataset, first we have determined which bearing test should be used in this work as well as the region in which the bearing should be considered as healthy or as defective. The first bearing of the second test and the third bearing of the third test were respectively used as a train and as a test set. The three approach tested on the Machine Fault Simulator Dataset have seen their accuracy drop by around 10% when applied to this dataset. The explanations to this drop are that, the defective and healthy signal on this dataset are not as clean as for the Machine Fault Simulator.

Additionally, the fact that the train and test set are now generated from different bearings. Finally, we have tried a trend approach in the form of the

Nelson Rule, and it was found that the first Nelson Rule stating that *one point is more than 3 standard deviations from the mean* was performing the best. In the next section we will be establishing the metrics to compare those bearing defect detection methods and compare them one to one.

5.3 Comparison of bearing defect detection methods

In this section, we will first define classical metrics used to compare classification methods. Then, we will compare the four above presented methods one to one.

5.3.1 Performance metrics

In the context of bearing defect identification, the problem that we are trying to solve is the one of a binary classification. Thus, it is suitable to use three already existing metric heavily used for these problems.

Accuracy: gives the fraction of correct prediction made by the model over the total number of prediction made by the model.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{5.1}$$

Precision: gives the ratio bearing correctly classified as having a defect. A precision of 1 corresponds to a model that is always correct when it predicts defects.

$$\text{precision} = \frac{tp}{tp + fp} \tag{5.2}$$

Recall: gives the ratio of defective bearing classified as having a defect. A recall of 1 corresponds to a model that classified all the actually defective bearing as being defective.

$$\text{precision} = \frac{tp}{tp + fn} \tag{5.3}$$

Note: In the above equations, fp denotes the false positive (bearing healthy classified as defective), fn denotes the false negative (bearing defective classified as healthy), tp denotes the true positive (bearing defective classified as defective), tn denotes the true negative (bearing healthy classified as healthy)

Adaptation to federated learning Contrary to the other approach, the federated learning context is different as there is a different test set per client. In order to compare this approach to the other, we need to define overall metrics. In the case of the Machine Fault Simulator Dataset, the overall metrics are obtained by executing the inference of the model on the aggregated dataset on the server obtained by pulling the test sets of all the clients. For instance, in the setting in which the federated context has 40 clients each with a 100 data point in their own dataset and a train/test ratio of 70 percent; each client will have 30 samples in its test set and the aggregate test set will contain 120 samples (40·30). It is on this aggregate test set that the model will be tested. In the case of the NASA dataset, we can simply use the data from the first bearing of the third test to evaluate the federated approach

In the following, we report the result of the four different approaches: (1) wavelet and neural network in centralized learning context, (2) wavelet and neural network in federated learning context, (3) peak finding in spectrum, (4) first Nelson rule for outlier detection. We first compare the centralized learning method to the other ones, after we compare the federated learning approach.

5.3.2 Centralized Learning and Trend analysis

Table 5.10 shows the results of the wavelet and neural network in a centralized context compared to the results of the Trend Analysis. As the Trend Analysis is not applicable to the Machine Fault Simulator Dataset, the comparison will be only based on the NASA Dataset.

Overall, the Trend Analysis out-performs the Centralized Learning approach on all metrics. The gap in accuracy between the Trend Analysis and the Centralized learning approach can be explained by the large difference of 0.15 in the Recall of those two methods. Consequently, it can be concluded that the Centralized Learning approach is not as sensitive to bearing defects as the Trend Analysis.

Finally, it should be noted that the Trend Analysis comes with a major shortcoming as it cannot be applied when the rotational speed of the equipment is changing as this change would be labeled as a defect. On the other hand, the results presented for the Centralized Learning Approach on the Machine Fault Simulator are for multiple turning speed of the equipment, and it can be seen that the approach maintains a good precision with different turning speeds.

Additionally, between the two datasets, the Centralized Learning approach's recall is higher by 0.15 in the case of the Machine Fault Simulator. This can be explained by the fact that the NASA Dataset contains early defects that just start to appear in the waveform.

Comparison results

The Peak Analysis performs better than the centralized learning approach however, this method is limited to constant turning speed equip-

Table 5.10: Comparison of precision, recall and accuracy between the Centralized Learning Approach and the Trend Analysis

Dataset	**NASA**		**MFS**	
Approach	**CL**	**Trend**	**CL**	**Trend**
Precision	0.977	1.000	0.962	NA
Recall	0.694	0.845	0.842	NA
Accuracy	0.839	0.923	0.904	NA

ments.

In the next section, we compare the Centralized Learning approach to the peak finding approach.

5.3.3 Centralized Learning and Peak analysis

In table 5.11, the comparison between the wavelet and Neural Network in Centralized Learning approach is compared to the Peak Analysis for both the NASA Dataset and the Machine Fault Simulator Dataset. For all metrics, the Centralized Learning approach performs better than the Peak Analysis. This last method performs poorly on the NASA dataset with a recall of 0.36, meaning that a large portion of actually defective signals are not flagged as such. It is also interesting to note that this recall improves significantly on the Machine Fault Simulator Dataset. This might again be due to the fact that the defect on this dataset is more clear than for the default in the NASA Dataset.

Table 5.11: Comparison of precision, recall and accuracy between the Centralized Learning Approach and the Peak Analysis

Dataset	**NASA**		**MFS**	
Approach	**CL**	**Peak**	**CL**	**Peak**
Precision	0.977	0.862	0.962	0.757
Recall	0.694	0.361	0.842	0.742
Accuracy	0.839	0.652	0.904	0.752

As the peak analysis identifies peak in the vibration spectrum, it is then easier to flag the defects in the Machine Fault Simulator dataset, leading to better recall.

Comparison results

The Centralized Learning context for the wavelet and Neural Networks performs better than the Peak Analysis for both Datasets.

In the next section, we will compare the wavelet and machine learning approach in two different context, the centralized learning context and the federated learning context.

5.3.4 Centralized Learning and Federated Learning

Table 5.12 shows the results for the three metrics.

Table 5.12: Comparison of precision, recall and accuracy between the Centralized Learning Approach and the Federated Learning Approach

Dataset	**NASA**		**MFS**	
Approach	**CL**	**FL**	**CL**	**FL**
Precision	0.977	0.874	0.962	0.947
Recall	0.694	0.625	0.842	0.819
Accuracy	0.839	0.767	0.904	0.887

As expected, the centralized learning performs better than the federated learning, the result is confirmed by [88]. Indeed, the federated learning process introduce another layer off complexity to the learning process with the federated averaging algorithm. From Table 5.12 it appears again that the bearing defects are easier to identify in the Machine Fault Simulator than in the NASA Dataset, as for both federated and centralized learning, the metrics improves from the first Dataset to the Second one. The only metric that does not increase from the NASA dataset to the Machine Fault Simulator dataset is the precision in centralized learning, however, the difference between the two settings is only of 0.01 which is not significant.

Comparison results

The Centralized Learning setting is performing better than the Federated Learning setting for the Wavelet and Neural Network approaches.

This analysis concludes the comparison of the Wavelet and Neural Network in the centralized learning context. In the following section, the Wavelet and Neural Network in the Federated Learning setting are compared to the Trend Analysis and the Peak Analysis.

5.3.5 Federated Learning and Trend analysis

Table 5.13 shows the precision, recall and accuracy for the Wavelet and Neural Networks compared to the Trend Analysis for the two datasets.

Table 5.13: Comparison of precision, recall and accuracy between the Federated Learning Approach and the Trend Analysis

Dataset	**NASA**		**MFS**	
Approach	**FL**	**Trend**	**FL**	**Trend**
Precision	0.874	1.000	0.947	NA
Recall	0.625	0.845	0.819	NA
Accuracy	0.767	0.923	0.887	NA

It shows that the Trend Analysis performs better than the Federated Learning on the NASA Dataset, which is expected as it was performing better than the Centralized Learning approach, which was performing better than the Federated Learning approach (cf. 5.10and 5.12). However, the Federated Learning approach present the same advantages as the centralized learning approach in the fact that they can be applied to varying rotational speeds

Comparison results

The Trend Analysis performs better than the Wavelet and Neural Network approach when applied to constant rotational speed signals.

5.3.6 Federated Learning and Peak analysis

In this section, we conclude with the last of the comparison which is the Wavelet and Neural Network in Federated Learning to the Peak Analysis. Table 5.14 shows the results for the precision, recall and accuracy. As for the Centralized Learning Approach, the Federated Learning approach outperforms the Peak Analysis. The only metric where the two methods are giving similar results is the precision in the NASA Datasets. This is probably due to the fact that the Peak Analysis method is very sensitive on the NASA dataset leading to a high precision, but also a very low recall with an overall low accuracy.

Table 5.14: Comparison of precision, recall, and accuracy between the Federated Learning Approach and the Peak Analysis

Dataset	**NASA**		**MFS**	
Approach	**FL**	**Peak**	**FL**	**Peak**
Precision	0.874	0.862	0.947	0.757
Recall	0.625	0.361	0.819	0.742
Accuracy	0.767	0.652	0.887	0.752

Comparison results

The Wavelet and Neural Network in the federated Learning context outperforms the Peak Analysis approach

5.3.7 Summary of the results across methods

In conclusion, we have compared four different methods for bearing defect detection on two different datasets: the Machine Fault Simulator Dataset and the NASA Dataset. The one to one comparison of the precision, recall, and accuracy of all these methods showed that: for a constant turning speed, a control charts with the first Nelson rule is the best performing method.

However, this method is sensitive to change in turning speed and in load; consequently, its field of application is limited. The Centralized and Federated Learning Context for the Wavelet and Neural Network approaches, which are the focus of this work, also give good results with an accuracy of more than 0.76 across the two datasets. More importantly, those methods were shown to be very precise, which makes them highly scalable as they will not generate large amounts of False Positive. Moreover, the Peak Analysis that is supposed to detect Outer Race Harmonic families performed poorly on both of the datasets used in this study. Finally, it can be concluded that the proposed wavelet and Neural Network performance are satisfying in both the centralized and federated learning contexts.

In this chapter, the proposed data acquisition architecture was validated and the PRU-ICSS of the BeagleBone AI were used to acquire vibration data on the Machine Fault Simulator. Once this dataset set was generated, it was used with another Dataset from NASA to validate the performance of the proposed Wavelet and Neural Network approaches in both federated and centralized learning contexts. The results were compared to a trend outlier detection based on the first Nelson Rule and on the spectrum peak analysis method developed in [44]. It was shown that, in the context of a varying rotational speed, the proposed approach out-performs both the peak analysis and the trend outlier detection. However, the trend outlier detection remains the best technique in the case of a constant rotational speed. In the next chapter, we will first present the contributions and limitation of this work, then the research question will be answered and suggestion for future work in this topic will be made.

CHAPTER 6

CONTRIBUTIONS AND LIMITATIONS

In this chapter, the contributions of this work are detailed, then the limitation of the developed techniques are clarified. Thereafter, the hypothesis made in chapter 3 are validated or rejected in order to answer the research questions. Finally, in order to overcome some limitations, ideas for the future work in this domain are proposed.

6.1 Contributions

The contribution of the work are multiple. First, in terms of implementation and technical contribution. Second, the interest of the developed method is presented. Third, the potential fields of application are detailed. Finally, the novelty of the approach is emphasized.

6.1.1 Technical contributions

In terms of technical contributions, one I2C-PRU driver, the use of this driver for vibration data acquisition, and one federated learning library were implemented.

I2C-PRU Driver: this driver permit to control the I2C compatible sensors with the Process Real-time Units of the BeagleBone AI. The PRU are controlled by the ARM Cortex using the RPMsg driver and the driver enable the use of the I2C bus of the BeagleBone AI with the PRU. This opens the field of use of the BeagleBone AI to deterministic sensing application. Moreover, this driver should be compatible with little adaptation to any device power by the ti-am5729 chip. This driver is configured to use a 400khz clock but can be easily reconfigured to use a 3.2MHz clock. The communication in the I2C bus were validated using an I2C phase analyzer.

Data acquisition: the use of the BeagleBone AI PRU-ICSS for vibration data acquisition was demonstrated. The above-mentioned driver is used to acquire the sample which are then place in the PRU-ICSS memory. Once the acquisition is completed, the non-deterministic Linux host can fetch the samples. This implementation shows the power of the BeagleBone microprocessor when used with the Process Real-time Units: they are low cost microprocessors running Linux which are also able to carry out deterministic task.

Federated Learning Library: this library was implemented as a wrapper around the TensorFlow machine learning framework. It exposes a client and sever architecture for federated learning processes simulations. Even if it is not as complete as the TensorFlow Federated library, it is also less complex and is not strongly type. This is an advantage and a limitation of the library as it makes it much easier to use, but it is also a potential source of error if the user is not rigorous.

The codes for the driver, its use, and the federated library are made available on GitHub and one should feel free to use them and to contact the author in case of questions.

6.1.2 Interest of the developed method

The proposed approach presents multiple interest as opposed to the classical bearing defect detection.

First, considering the results in the Federated Learning context, it can be argued that this bearing defect detection method leverages the power of the most recent microprocessor in order to decouple the bearing defect detection process from cloud computing.

Indeed, the Federated Learning approach was shown to obtain accuracies around 0.75 with a very good precision, which makes the approach very scalable as it reduces the false positives.

Additionally, in a setting where data can be transmitted to the cloud, centralized learning to retrain the edge models was also tested and proven to obtain better results than federated learning. However, this comes with limitations in terms of data privacy.

6.1.3 Potential field of application

The proposed method uses edge inference of bearing defects using Discrete Wavelet Transforms and Artificial Neural Networks. As it leverages the processing power of the edge device, it drastically reduces the requirement in the amount of bandwidth required by the system.

This open a field of applications where the bearing's condition needs to be monitored in an area were the network connectivity is limited or unreliable. For instance, in a remote area like a wind-turbine field, the connectivity to the internet might be limited, but one can imagine a federated learning process between the wind-turbine of the same field which does not require access to the cloud. In this case, only the status of each wind-turbine needs to communicated over to the internet, which significantly reduces the required bandwidth while taking advantage of the learning opportunities offered by the vibration data of each wind-turbine.

Additionally, in a context of increasing cyber-risk, the more data is transmitted to the internet, the more vulnerable a system is to attacks. For sensitive equipment, sending only the diagnostic information and not the full process data mitigates these new risks.

6.1.4 Novelty of the approach

This work presents a novel approach for bearing defect detection that leverages the processing power available at the edge. This approach was proven to work on outer race defect from two different datasets without any knowledge of the bearing configuration. It constitutes a proof of concept of edge federated learning of bearing defect detection. However, there are still some limitations in the current implementation, these limitations are presented in the following.

6.2 Limitations

The proposed approaches present different limitations, in terms of acquisition frequency of the vibration data, and in terms of defect identification accuracy.

6.2.1 Data-rate limitations

The developed PRU-I2C driver is limited in terms of acquisition frequency by the clock of the I2C bus. In the current implementation, the bus is limited at a frequency of 400kHz which gives a maximum sampling rate of 1600Hz while validating the received 16-bit samples as explained in 4.3.3. Additionally, the data point are stored in the PRU memory, which is limited in size, for 8bit data the maximum number of data point achievable is 12288. In addition, the data processing on the BeagleBone AI and the inference takes a couple of seconds as it was implemented on the CPU, which limits the intervals between two samples. However, this limitation is not critical because of the rate at which a bearing usually wears out. Some solutions to address those technical limitations are detailed in 6.4.

6.2.2 Defect identification limitation

Even if the centralized and federated learning approach compare fairly well to the other bearing defect detection methods and benefit from good precision, they still remain limited in terms of accuracy. In this work, the study of the optimization of the Neural Network was not conducted, and it is highly probable that other architecture will achieve much better results.

Finally, the test on the dataset where conducted on train and test set from the same bearings models. Even if, two datasets with two different bearing models where tested, this data remain experimental and generated on a test bench: either at GeorgiaTech on the Machine Fault Simulator or retrieved from the bearing run- to-failure test hosted on the NASA online repository [38]. The approach was not tested on actual production data, which are by nature more complex and less easy to learn from due to the change in the production operations.

In the following section we go back to the hypothesis formulated in chapter 3 and validate or reject them based on the result obtained in the previous chapters.

6.3 Answer to the Research question

6.3.1 Answer to the first research question

Research Question 1

How can bearing defect detection be automated without previous knowledge of their configuration?

To answer the research question we formulated 2 hypotheses which are presented bellow:

Hypothesis 1

Wavelet transforms associated with machine learning enables automated bearing defect detection with the same performance as classical methods without previous knowledge of the bearing configuration.

The result presented in section 5.3 show that we can indeed achieve the similar performance as classical bearing defect detection methods. In addition, in both the federated learning and centralized learning context, no bearing configuration was provided. These approaches do not require the intervention of an operator to predict the state of a bearing thus:

Hypothesis 1 is validated

Hypothesis 2

The wavelet transform step can be merged into the learning approach in order to significantly improve the process. This can be achieved not only at the device level by merging the algorithms, but also at the network level by implementing a federated learning approach between multiple devices.

The choice of the discrete wavelet used as a preprocessor for the Neural Network was based on the accuracy of the obtained algorithm during training. It was shown (C.f.: Table 5.2 and 5.6) that a correct choice of Discrete Wavelet Transform leads to a significant improvement of up to 0.14. However, the developed federated learning approach did not give better results than the centralized learning approach as shown in Table 5.12. Nonetheless, even if Federated learning does not improve the process, it presents other advantages in terms of data privacy and security, as the data is kept local to the device and not shared in the cloud.

Nevertheless:

Hypothesis 2 has to be rejected

Answer to Research Question 1

Finally, wavelet transform associated with machine learning enables automated bearing defect detection with the same performance as classical methods without previous knowledge of the bearing configuration. The choice of the wavelet used is critical to the final performance of the model at the device level. At the network level, the use of Federated Learning will reduce the accuracy of the bearing defect detection approach but will improve the data privacy and security.

6.3.2 Answer to the second research question

The second research question is the following:

Research Question 2

How can this approach be decoupled from cloud computing while maintaining adequate sampling rate and sensitivity for predictive maintenance applications?

It was proposed to address it with the two hypothesis below:

Hypothesis 3

By leveraging the recent developments on microprocessor and System on Chip, a parallel implementation of the above-mentioned techniques can be developed at the edge.

Even if it was initially proposed to use the full capabilities of the BeagleBone AI with the GPU and the DSP, some limitations were encountered and the processing of the wavelet transform is done in parallel on the CPU cores as well as the inference and the retraining of the machine learning model. The computing time of a couple of second is compatible with online execution of the proposed method. Moreover, the use of the PRU-ICSSS of the BeagleBone AI's chip enabled the deterministic data acquisition on the same System on Chip as the one holding the Linux microprocessor. The technique presented as a solution to the first research question was successfully deployed at the edge as presented in 4, thus:

Hypothesis 3 is validated

Hypothesis 4

The parallel wavelet implementation at the edge enables an early detection of the bearing defect that is suitable for predictive maintenance planning.

The results for the centralized learning approach presented in 5.3.2, shows that the proposed approach is comparable to control charts in terms of accuracy and prediction, which are used in predictive maintenance. Moreover, the approach is also applicable in systems in which the rotational speed and the load change along the time, and it maintains 90% of accuracy. Thus:
Hypothesis 4 is validated

Answer to Research Question 2

Finally, the proposed approach can be decoupled from the cloud even without using all the specialized core of the BeagleBone AI. The inference of the model is taking longer that it would have initially on the specialized core, but it is not an issue for online monitoring of bearing defect considering the time at which bearing defects appear.

6.3.3 Answer to the third research question

The last research question addresses the benchmarking with the cloud techniques.

Research Question 3

What is the trade-off between cost and performance (speed, sampling rate, and sensitivity) comparing the edge implementation and the cloud implementation?

The following hypothesis was formulated.

Hypothesis 5

The edge implementation will have comparable or better results than cloud computing services at a lower cost.

Even if the models used in this work were limited in size, thus giving an advantage to the decoupled architecture against a cloud base implementation; the results obtained in section 5.3 show that the decoupled approach has a performance comparable to the cloud computing implementation.

The sampling rate of 1600Hz was enough to detect outer race defect with an accuracy of 0.9 while maintaining a processing time compatible to online monitoring of bearing defects.

Considering the cost of the solution compared to a cloud based solution, no definitive answer is given as too many parameters are at play. On the one hand, considering the cost of the edge solution at around \$200 per asset, one can achieve a cheaper cloud base solution with a strict digital discipline to reduce the cost of the cloud base service while using low cost microcontrollers (around \$25), sensor (around \$15). On the other hand, in the case mentioned in 6.1.3, the cloud solution will become quickly more expensive that the decoupled architecture as most of the cost of the installation will be the cost of data transmission, giving an advantage to the edge solution. As it was established that in some situations the edge implementation is more expensive than the cloud base solution:

Hypothesis 5 is rejected

Answer to Research Question 3

Finally, even if the edge architecture proposed in the work give similar results than cloud base architecture. The trade-off highly depends on the application specific, and it is argued that one of the major factor of decision will be the cost of data transmission

6.3.4 Answer to the main research question

Research Question

How can a new approach for automated bearing defect detection be decoupled from Cloud computing?

Answer to Research Question

By leveraging the most recent SoC and using Wavelet Transform associated to Neural Networks, an edge bearing defect detection approach can be implemented. This approach can be trained in a centralized learning context to obtain a similar accuracy, precision, and recall than a cloud base approach while maintaining a speed compatible with online monitoring of bearing defect for predictive maintenance. Furthermore, this same approach can also be used in a federated learning manner to improve data security and privacy at the cost of a small reduction of accuracy.

6.4 Future Work

In this work, one of the major limitations was the difficulty to use the specialized core of the BeagleBone AI. With a new version of this board expected to be realized in the coming months, future work should adapt the current approach to the new version of the board. Indeed, it is likely that the PRU-I2C driver will become deprecated as the new version of the BeagleBone AI will use another Texas Instrument Chip. This new version of the BeagleBone AI should be able to run the Texas Instrument library “Edge AI”. Adapting this work to the library would open the possibility to run more complex models. Moreover, the PRU-I2C driver could be expanded to support 3.2MHz clocking in order to have a faster communication with the sensors. As there is no such driver on other models of BeagleBone boards, it will be very valuable to adapt the driver to these models.

Finally, the Federated Learning library can be extended to permit any client to become the server of the centralized learning context, which would reduce the sensitivity of the federated learning network to an issue in the server.

CHAPTER 7

CONCLUSION

This dissertation introduced a new bearing defect detection method using wavelet transform and artificial neural networks at the edge.

After considering that the current methods used in the field of bearing defect detection, they are either not automatized or require the use of cloud computing. This work leverages the process Real-time units' deterministic capabilities and CPU cores of the BeagleBone AI to carry out data acquisition and processing in an architecture decoupled from the cloud.

The architecture achieves automatic bearing defects detection with accuracy of up to 90% and a precision of 97% with a speed compatible to online monitoring of bearing defects. These results are similar to the ones obtained with the other bearing defects detection methods used for comparison.

Moreover, while maintaining an accuracy of up to 88% and a precision of up to 94%, this architecture can be deployed in a federated learning manner, which increases the data safety and privacy in world of ever-increasing cyber-risk.

Finally, the use of the proposed approach to process vibration data at the edge significantly reduces the need to transmit large amounts of data to the cloud as only the status of the equipment needs to be reported to the user.

Appendices

APPENDIX A

DERIVATION OF THE BEARING DEFECT FREQUENCIES

A.1 Derivation

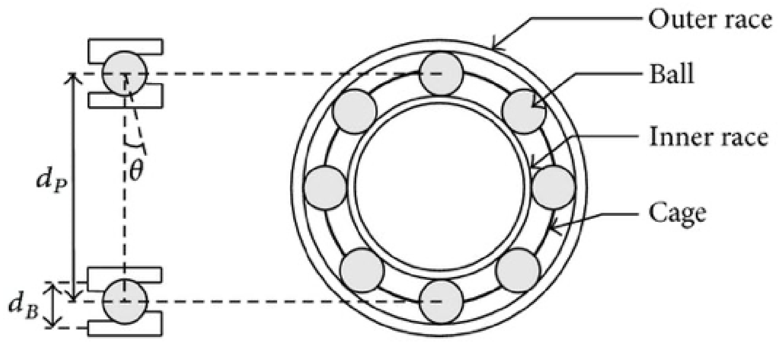


Figure A.1: Components of rolling elements bearings[108]

From Figure A.1, the radius of the outer race is given by $\frac{d_b}{2} + \frac{d_B}{2}\cos\theta$. Denoting N_o the outer race's angular speed, the velocity of a point of the outer race is:

$$V_o = \frac{N_o}{2}\left[d_P + d_B\cos\theta\right] \tag{A.1}$$

The velocity in the inner can be derived similarly to equation A.1 and is:

$$V_i = \frac{N_i}{2}\left[d_P - d_B\cos\theta\right] \tag{A.2}$$

And the velocity at the center of the rolling element can be computer as the mean between the between V_i and V_o such that $V_c = \frac{V_o+V_i}{2}$. Then we can obtain the rotational speed at the center of the rolling element with:

$$\begin{aligned} N_c &= \frac{V_o + V_i}{2} \frac{2}{d_P} \\ &= \frac{1}{2d_P} \left[N_o \left(d_P + d_B \cos\theta \right) + N_i \left(d_P - d_B \cos\theta \right) \right] \\ &= \frac{1}{2} \left[N_o \left(1 + \frac{d_B}{d_P} \cos\theta \right) + N_i \left(1 - \frac{d_B}{d_P} \cos\theta \right) \right] \end{aligned} \tag{A.3}$$

From the above computed rotational speed we can compute the frequency at which a single element passes at a given point of the outer race:

$$\begin{aligned} N_{o/b} &= N_o - N_c \\ &= N_o - \frac{1}{2} \left[N_o \left(1 + \frac{d_B}{d_P} \cos\theta \right) + N_i \left(1 - \frac{d_B}{d_P} \cos\theta \right) \right] \\ &= \frac{1}{2} \left(N_o - N_i \right) \left[1 - \frac{d_B}{d_P} \cos(\theta) \right] \end{aligned} \tag{A.4}$$

Considering that they are n rolling elements in the bearing, we obtain the frequency of any rolling element passing on one given point of the outer race:

$$N_{o/b}^n = \frac{n}{2} \left| N_o - N_i \right| \left[1 - \frac{d_B}{d_P} \cos(\theta) \right] \tag{A.5}$$

Similarly to A.5, we derive the passage frequency of any rolling element on the inner-race:

$$N_{i/b}^n = \frac{n}{2} \left| N_i - N_o \right| \left[1 + \frac{d_B}{d_P} \cos(\theta) \right] \tag{A.6}$$

Denoting N_r the rotational speed of a rolling element we can express V_o as in A.1 but also:

$$V_o = \frac{1}{2} N_r d_B + \frac{1}{2} N_c \left[d_P + d_B \cos(\theta) \right] \tag{A.7}$$

Equating A.1 and A.7 and solving for N_r:

$$N_r = [N_o - N_c]\left[\frac{d_P}{d_B} + \cos(\theta)\right] \tag{A.8}$$

Then we can use A.8 and reinject the value of N_c from A.3 to find:

$$\begin{aligned} N_o\left(1+\frac{d_B}{d_P}\cos\theta\right) = & \\ & N_r d_B + \frac{1}{2}\left(1+\frac{d_B}{d_P}\cos\theta\right) \\ & \times\left[N_o\left(1+\frac{d_B}{d_P}\cos\theta\right) + N_i\left(1-\frac{d_B}{d_P}\cos\theta\right)\right] \end{aligned}$$

After solving for N_r the rotational speed of a rolling element we find:

$$\begin{aligned} N_r &= \frac{1}{2}\left|N_o - N_i\right|\left[\frac{d_P}{d_B} + \cos\theta\right]\left[1-\frac{d_B}{d_P}\cos\theta\right] \\ &= \frac{1}{2}\left|N_o - N_i\right|\frac{d_P}{d_B}\left[1-\left(\frac{d_B}{d_P}\cos\theta\right)^2\right] \end{aligned} \tag{A.9}$$

A.2 Summary of the bearing defect frequencies

Finally, we have obtained the excitation frequency for defects on the inner and outer-races, those frequencies are called:

BPFO for Ball Pass Frequency Outer-race:

$$\boxed{BPFO = \frac{n}{2}\left|N_o - N_i\right|\left[1-\frac{d_B}{d_P}\cos(\theta)\right]} \tag{A.10}$$

BPFI for Ball Pass Frequency Inner-race:

$$\boxed{BPFI = \frac{n}{2}\left|N_i - N_o\right|\left[1+\frac{d_B}{d_P}\cos(\theta)\right]} \tag{A.11}$$

BPFI for Ball Pass Frequency Inner-race:

$$\boxed{FTF = n\,|N_i - N_o| \left[1 + \frac{d_B}{d_P}\cos(\theta)\right]} \qquad \text{(A.12)}$$

With:

n	: the number of rolling elements in the bearing
N_o	: the rotational frequency of the outer-race
N_i	: the rotational frequency of the inner-race
d_B	: the diameter of the rolling element
d_P	: the diameter of the circle described by the rolling element
θ	: the contact angle

Note: These formulas use the $|N_i - N_o|$ which is usually referred as f_r

APPENDIX B

MACHINE FAULT SIMULATOR

Here we present some anoted pictures of the Machine Fault Simulator, which are not essential to the understanding of the system but can help to clarify it.

B.1 Imbalance loader installed

This picture represents the machine fault simulator with the imbalance loader in operation. Some weight can be screwed on the imbalance loader.

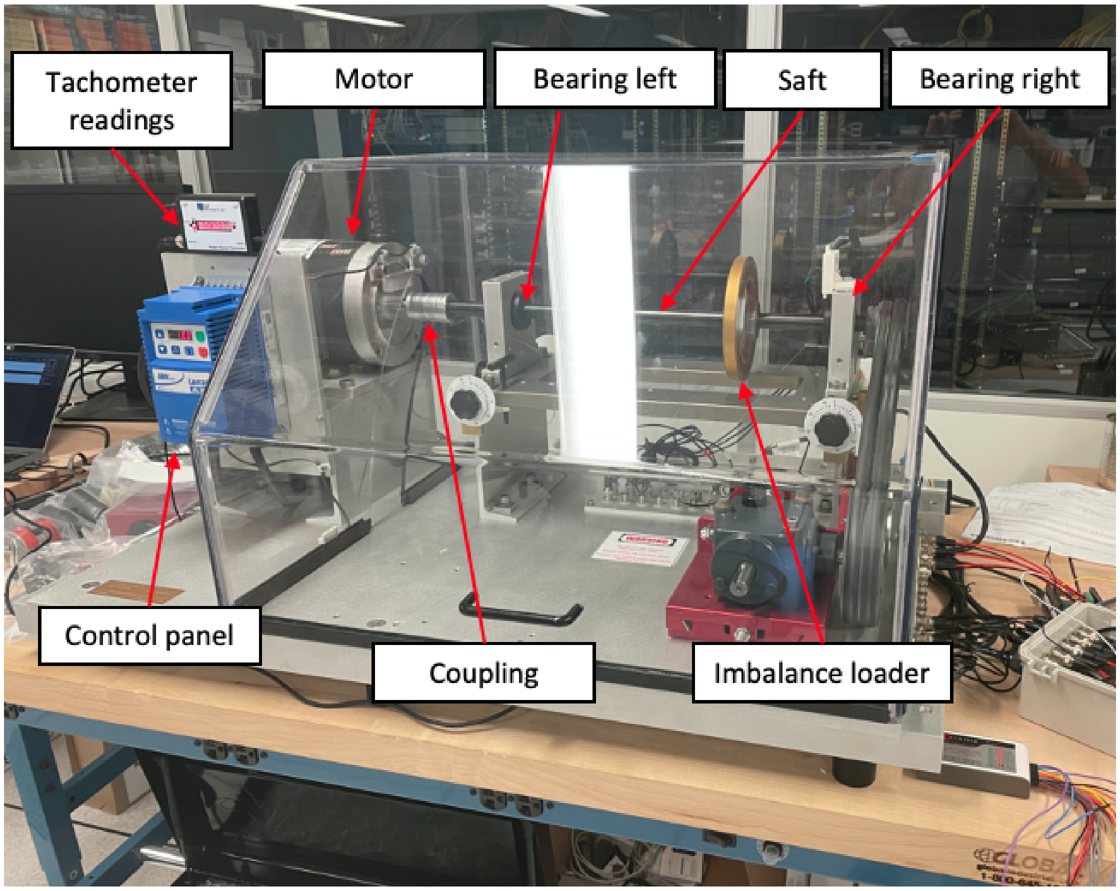


Figure B.1: The MFS with the imbalance loader installed

B.2 Components of the Machine Fault Simulator

Here we present another view of the major components of the Machine Fault Simulator

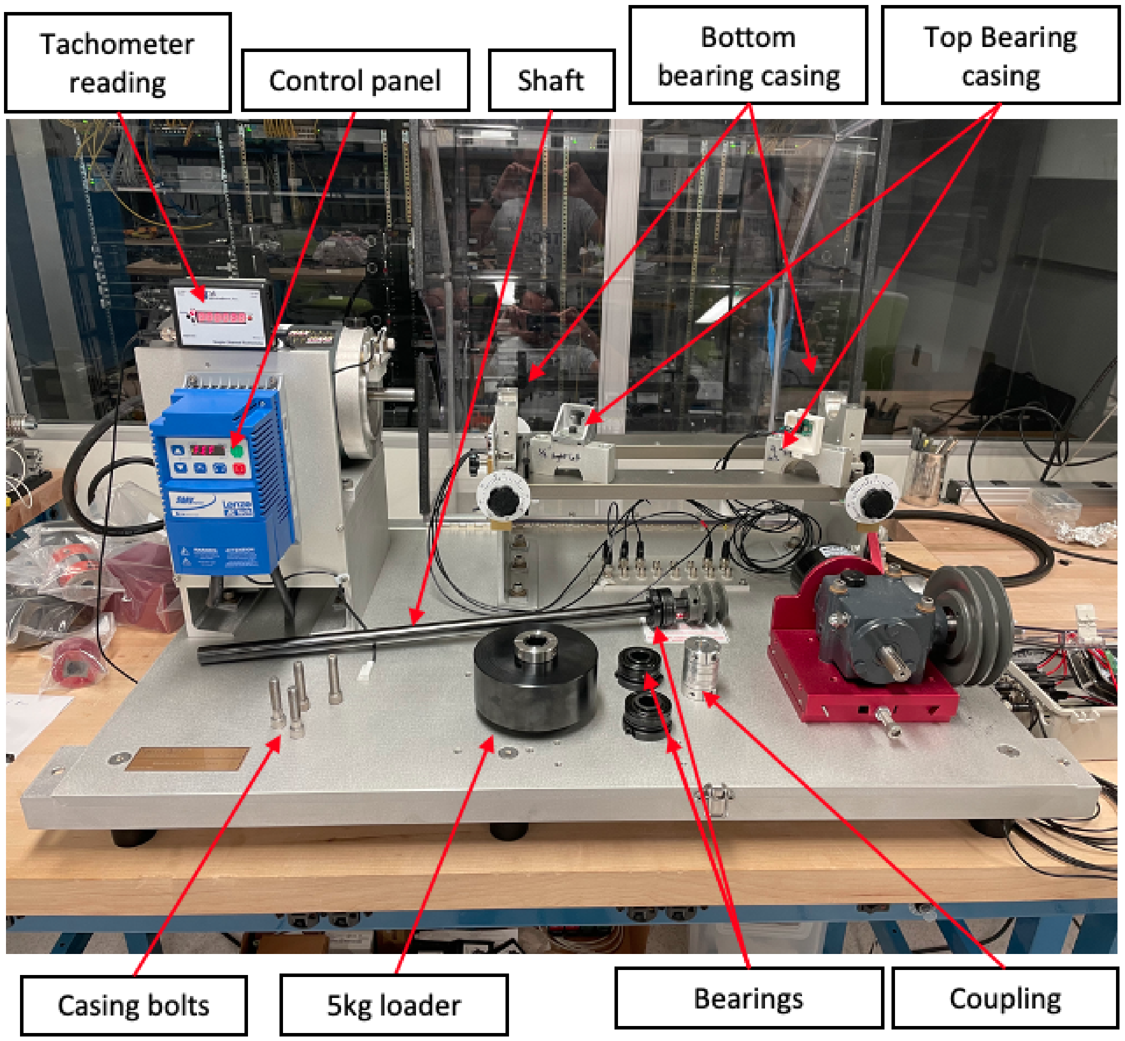


Figure B.2: The MFS with the imbalance loader installed

APPENDIX C

CENTRALIZED LEARNING ON MACHINE FAULT SIMULATOR DATASET

C.1 Model 0

Accuracy of Model 0 on the Machine Fault Simulator for different families of discrete wavelet transforms.

Table C.1: Accuracy of Model 0 in centralized learning for different wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.5	0.9038	bior3.7	0.8533	sym16	0.8221	coif2	0.7421
bior5.5	0.8929	rbio3.3	0.8504	db18	0.8204	bior6.8	0.735
db29	0.8838	db33	0.85	db1	0.8204	rbio1.3	0.7325
db24	0.8817	sym13	0.8492	sym4	0.82	sym5	0.7025
coif17	0.88	rbio5.5	0.8488	bior2.2	0.8175	db5	0.6825
rbio2.8	0.8783	coif4	0.8488	db4	0.8171	coif6	0.6821
db35	0.8758	sym14	0.8479	sym12	0.8167	db3	0.6792
db27	0.8758	db19	0.8467	db37	0.8129	db2	0.6633
db38	0.8742	coif8	0.8467	bior3.3	0.8113	coif13	0.6504
coif16	0.8729	sym19	0.8467	db12	0.8092	rbio3.9	0.5958
coif9	0.8717	sym10	0.8438	db31	0.8071	rbio3.1	0.5733
db22	0.8692	coif1	0.8433	db30	0.8	coif5	0.5671
db23	0.8671	db7	0.8421	sym15	0.7996	sym9	0.5592
coif10	0.8667	db25	0.8408	sym7	0.7954	db6	0.5163
coif7	0.8663	bior1.5	0.8404	db15	0.7892	sym11	0.5067
db36	0.8654	dmey	0.8392	db16	0.7846	db13	0.5063
coif11	0.8633	coif3	0.8379	rbio4.4	0.7792	rbio2.6	0.5063
db26	0.8617	sym18	0.8375	sym17	0.7738	sym6	0.5063
db14	0.86	sym2	0.8367	rbio2.2	0.7671	haar	0.5063
bior3.9	0.8588	sym8	0.8354	db20	0.7625	db34	0.5063
db32	0.8583	bior3.5	0.8354	sym20	0.7608	coif15	0.5063
coif14	0.8575	db28	0.8283	rbio6.8	0.7575	coif12	0.5063
db11	0.8558	bior2.4	0.8283	rbio1.5	0.7567	bior2.8	0.5063
db17	0.855	bior4.4	0.8271	rbio2.4	0.7542	bior3.1	0.5063
db21	0.855	bior1.1	0.8271	bior2.6	0.7533	bior1.3	0.5063
db9	0.8533	rbio3.7	0.8242	db10	0.7475		
db8	0.8533	rbio1.1	0.8238	sym3	0.7433		

C.2 Model 1

Accuracy of Model 1 on the Machine Fault Simulator for different families of discrete wavelet transforms.

Table C.2: Accuracy of Model 1 in centralized learning for different wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.5	0.8688	db15	0.8138	sym6	0.7667	db8	0.7008
coif17	0.8663	db13	0.8125	db20	0.7650	coif7	0.6792
db30	0.8621	coif6	0.8083	db3	0.7650	bior1.1	0.6683
db37	0.8608	bior3.7	0.8067	rbio1.1	0.7642	coif12	0.6667
coif15	0.8596	db16	0.8046	db4	0.7633	sym14	0.6508
db38	0.8583	coif5	0.8029	sym2	0.7625	rbio4.4	0.6379
db32	0.8550	sym17	0.8029	db17	0.7600	bior1.3	0.6046
db29	0.8546	bior2.6	0.8013	rbio1.5	0.7596	db1	0.5958
rbio3.9	0.8538	sym11	0.7963	coif1	0.7592	db11	0.5925
db31	0.8508	rbio6.8	0.7942	db21	0.7583	db33	0.5513
db25	0.8488	db14	0.7925	rbio2.2	0.7579	rbio2.4	0.5067
db27	0.8446	db28	0.7921	sym4	0.7575	bior5.5	0.5067
coif10	0.8421	db36	0.7875	db2	0.7554	haar	0.5063
coif16	0.8400	bior2.4	0.7850	bior4.4	0.7542	db34	0.5063
db35	0.8358	coif2	0.7846	db5	0.7538	bior2.2	0.5063
coif13	0.8354	coif4	0.7821	rbio1.3	0.7492	coif3	0.5063
coif9	0.8317	rbio5.5	0.7821	sym13	0.7483	bior3.3	0.5063
db18	0.8313	db7	0.7796	sym10	0.7475	db24	0.5063
rbio2.6	0.8308	bior2.8	0.7771	sym8	0.7471	db23	0.5063
db22	0.8283	bior6.8	0.7750	db6	0.7450	sym18	0.5063
bior3.9	0.8246	sym9	0.7746	db9	0.7421	coif11	0.5063
coif14	0.8246	sym7	0.7725	sym20	0.7392	sym15	0.5063
rbio3.3	0.8238	bior3.5	0.7700	db26	0.7379	db12	0.5063
db19	0.8233	db10	0.7696	sym3	0.7254	rbio3.1	0.5063
rbio2.8	0.8200	sym16	0.7692	coif8	0.7246	bior3.1	0.4942
rbio3.7	0.8167	sym12	0.7692	sym5	0.7075		
bior1.5	0.8154	dmey	0.7671	sym19	0.7058		

C.3 Model 2

Accuracy of Model 2 on the Machine Fault Simulator for different families of discrete wavelet transform

Table C.3: Accuracy of Model 2 in centralized learning for different wavelet transforms

wavelet	**accuracy**	**wavelet**	**accuracy**	**wavelet**	**accuracy**	**wavelet**	**accuracy**
rbio3.9	0.8879	coif9	0.8375	bior3.3	0.7942	rbio1.5	0.7313
rbio3.7	0.8854	rbio5.5	0.8371	rbio3.1	0.7925	db7	0.7213
bior5.5	0.8808	sym9	0.8371	db20	0.7900	rbio3.5	0.7096
coif16	0.8758	db6	0.8363	sym7	0.7888	coif8	0.7079
db37	0.8738	bior3.7	0.8354	db11	0.7888	bior3.9	0.7029
db35	0.8650	rbio2.4	0.8333	coif3	0.7871	coif7	0.6913
db29	0.8646	db9	0.8308	coif12	0.7838	db13	0.6804
rbio2.8	0.8642	sym19	0.8250	sym10	0.7783	sym8	0.6550
db26	0.8600	sym16	0.8242	bior1.1	0.7746	coif6	0.6488
db24	0.8600	sym13	0.8188	rbio2.2	0.7725	rbio4.4	0.6471
coif15	0.8588	bior2.8	0.8179	db12	0.7708	bior1.3	0.6346
db25	0.8579	rbio1.1	0.8167	sym17	0.7704	db15	0.6300
coif14	0.8567	sym12	0.8158	db2	0.7675	db32	0.5850
coif11	0.8563	bior6.8	0.8146	sym18	0.7642	bior2.6	0.5642
coif10	0.8558	db18	0.8146	db5	0.7633	sym4	0.5633
rbio2.6	0.8550	sym3	0.8108	rbio1.3	0.7608	db21	0.5367
db36	0.8546	db8	0.8075	sym2	0.7596	db22	0.5329
db19	0.8542	rbio6.8	0.8058	db3	0.7575	coif17	0.5063
db33	0.8529	coif5	0.8038	sym20	0.7521	db28	0.5063
coif13	0.8467	sym6	0.8033	sym14	0.7517	bior3.1	0.5063
bior1.5	0.8442	coif4	0.8033	sym15	0.7488	db34	0.5063
db17	0.8429	coif1	0.8008	bior2.2	0.7442	db38	0.5063
db30	0.8425	db31	0.8000	haar	0.7421	sym11	0.5063
db14	0.8413	db27	0.7971	bior2.4	0.7413	db23	0.5063
dmey	0.8413	db10	0.7963	bior4.4	0.7379	coif2	0.5063
bior3.5	0.8396	sym5	0.7950	db1	0.7367		
db16	0.8396	db4	0.7946	rbio3.3	0.7342		

C.4 Model 3

Accuracy of Model 3 on the Machine Fault Simulator for different families of discrete wavelet transforms

Table C.4: Accuracy of Model 3 in centralized learning for different wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.7	0.8846	sym13	0.8250	sym5	0.7996	db24	0.7542
coif15	0.8575	db15	0.8246	coif9	0.7988	db18	0.7538
coif17	0.8571	bior5.5	0.8242	rbio3.9	0.7988	db19	0.7488
db31	0.8563	bior3.7	0.8233	rbio2.2	0.7967	sym16	0.7358
db36	0.8558	bior2.8	0.8225	coif11	0.7954	sym12	0.7317
db38	0.8538	bior3.5	0.8192	sym2	0.7938	coif3	0.7308
db29	0.8525	rbio6.8	0.8188	sym14	0.7933	rbio2.4	0.7296
db34	0.8517	coif10	0.8188	sym4	0.7921	bior1.1	0.7192
db33	0.8496	db13	0.8188	db9	0.7917	sym7	0.7146
dmey	0.8488	db6	0.8167	coif5	0.7900	sym6	0.7017
coif14	0.8458	coif4	0.8158	sym20	0.7879	rbio1.3	0.7004
rbio3.3	0.8454	sym3	0.8146	db30	0.7850	db3	0.6908
db25	0.8438	db7	0.8138	sym15	0.7833	db23	0.6821
rbio2.8	0.8433	db17	0.8125	bior2.4	0.7792	rbio3.1	0.6517
db28	0.8421	bior2.6	0.8117	db4	0.7792	coif8	0.6479
rbio2.6	0.8400	db35	0.8113	sym9	0.7767	bior4.4	0.6363
db21	0.8379	db8	0.8113	coif2	0.7729	bior1.3	0.6213
coif12	0.8371	sym11	0.8104	db5	0.7704	db10	0.6108
bior3.9	0.8346	db20	0.8104	sym10	0.7696	db12	0.5667
db37	0.8346	db14	0.8096	sym8	0.7646	db1	0.5567
coif7	0.8338	db2	0.8088	haar	0.7625	bior3.1	0.5138
db32	0.8338	sym19	0.8083	db22	0.7621	bior3.3	0.5063
coif16	0.8321	bior2.2	0.8050	bior6.8	0.7621	db26	0.4938
coif13	0.8317	rbio3.5	0.8046	db16	0.7617	db11	0.4938
rbio5.5	0.8304	db27	0.8046	rbio1.5	0.7608	coif1	0.4938
bior1.5	0.8300	sym18	0.8017	sym17	0.7575		
coif6	0.8283	rbio4.4	0.8004	rbio1.1	0.7563		

C.5 Model 4

Accuracy of Model 4 on the Machine Fault Simulator for different families of discrete wavelet transforms

Table C.5: Accuracy of Model 4 in centralized learning for different wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.5	0.8925	sym16	0.8367	sym6	0.8063	sym14	0.7163
bior5.5	0.8871	coif7	0.8367	db20	0.8033	db9	0.6875
rbio3.9	0.8863	db15	0.8363	sym20	0.8025	rbio4.4	0.6800
db33	0.8704	db12	0.8354	bior2.2	0.8008	rbio3.1	0.6683
db31	0.8683	sym9	0.8350	coif2	0.7967	coif1	0.6663
db35	0.8667	db21	0.8346	bior4.4	0.7883	sym12	0.6563
db25	0.8633	sym8	0.8333	coif10	0.7879	db3	0.6517
db38	0.8629	db1	0.8329	sym18	0.7875	bior2.6	0.6367
db23	0.8608	db14	0.8313	bior2.8	0.7854	db36	0.6275
db27	0.8550	rbio6.8	0.8304	db5	0.7850	bior3.5	0.6017
coif13	0.8504	coif11	0.8300	dmey	0.7842	sym13	0.5950
coif9	0.8504	db22	0.8288	coif14	0.7754	bior3.3	0.5679
coif12	0.8471	db10	0.8271	coif5	0.7754	rbio2.2	0.5450
sym17	0.8467	bior6.8	0.8267	rbio1.3	0.7738	rbio2.6	0.5158
coif16	0.8458	db6	0.8258	sym5	0.7713	bior3.1	0.5088
db32	0.8458	db8	0.8246	rbio1.5	0.7667	coif3	0.5063
rbio5.5	0.8458	bior2.4	0.8242	db29	0.7667	db13	0.5063
db26	0.8450	sym11	0.8238	rbio2.4	0.7663	sym15	0.4958
bior1.5	0.8446	db4	0.8233	db19	0.7650	coif17	0.4938
db11	0.8438	db2	0.8204	sym3	0.7633	coif15	0.4938
db17	0.8438	sym10	0.8196	bior1.1	0.7625	db7	0.4938
coif8	0.8438	db18	0.8196	db30	0.7604	haar	0.4938
rbio3.3	0.8433	db34	0.8175	coif4	0.7575	rbio3.7	0.4938
sym4	0.8425	sym7	0.8158	sym2	0.7563	db16	0.4938
bior3.9	0.8425	db28	0.8158	coif6	0.7421	bior3.7	0.4917
rbio2.8	0.8421	bior1.3	0.8142	rbio1.1	0.7300		
sym19	0.8413	db37	0.8100	db24	0.7271		

C.6 Model 5

Accuracy of Model 5 on the Machine Fault Simulator for different families of discrete wavelet transforms

Table C.6: Accuracy of Model 5 in centralized learning for different wavelet transforms

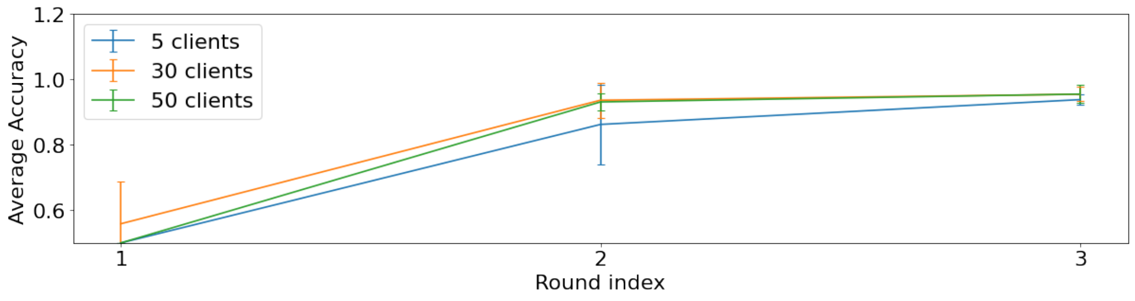
wavelet	**accuracy**	**wavelet**	**accuracy**	**wavelet**	**accuracy**	**wavelet**	**accuracy**
coif16	0.8629	bior1.3	0.8138	sym19	0.7863	sym8	0.7558
db37	0.8596	coif6	0.8092	rbio5.5	0.7858	rbio2.2	0.7554
coif17	0.8575	db28	0.8088	sym3	0.7846	db6	0.7546
rbio3.7	0.8558	coif11	0.8071	db14	0.7804	bior3.7	0.7542
db38	0.8554	sym18	0.8050	rbio3.5	0.7800	bior6.8	0.7533
db32	0.8517	db22	0.8038	db10	0.7771	sym2	0.7529
db25	0.8492	rbio3.3	0.8038	db8	0.7771	bior3.3	0.7517
db27	0.8475	rbio2.8	0.8029	db34	0.7767	sym5	0.7488
db33	0.8471	rbio6.8	0.8021	bior2.2	0.7721	rbio4.4	0.7479
coif12	0.8425	bior3.9	0.8017	db3	0.7717	rbio1.3	0.7471
coif15	0.8421	sym20	0.8013	db5	0.7713	sym16	0.7467
db31	0.8421	sym15	0.7967	coif4	0.7704	bior4.4	0.7463
db24	0.8379	rbio1.1	0.7963	sym14	0.7704	dmey	0.7346
coif13	0.8371	db23	0.7958	db18	0.7692	db12	0.7304
db35	0.8371	sym12	0.7954	rbio2.4	0.7692	coif5	0.7250
db36	0.8367	db30	0.7946	haar	0.7688	bior2.4	0.7192
db21	0.8321	sym13	0.7933	sym6	0.7667	db4	0.7179
db20	0.8279	coif14	0.7921	sym9	0.7642	sym7	0.7154
coif9	0.8271	db1	0.7908	bior1.1	0.7617	db11	0.7017
bior2.8	0.8254	sym17	0.7908	bior3.5	0.7617	coif1	0.7004
coif8	0.8250	rbio3.1	0.7904	sym4	0.7617	db13	0.6825
db29	0.8217	coif7	0.7896	db2	0.7608	db15	0.6075
rbio2.6	0.8183	db7	0.7896	coif2	0.7604	db26	0.5538
rbio3.9	0.8183	db16	0.7892	sym10	0.7600	db19	0.5063
coif10	0.8163	db9	0.7883	sym11	0.7592	bior3.1	0.4938
bior2.6	0.8146	bior5.5	0.7879	rbio1.5	0.7575		
db17	0.8142	coif3	0.7875	bior1.5	0.7563		

APPENDIX D

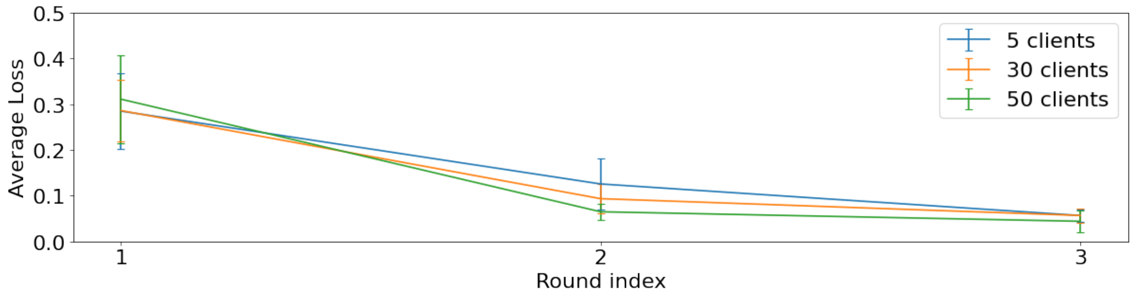
FEDERATED LEARNING ON MACHINE FAULT SIMULATOR DATASET

D.1 Influence of the total number of client

Figure D.3 and D.4 show the evolution of the accuracy and the loss with an increase in the number of clients. The representation is obtained for 10 runs of the same experiment and shows the average of the accuracy and loss per round with the standard deviation. The representation the intuitive result that the more we have in the federated learning context the better the accuracy gets. However, the improvement is not significant in this case, as the error bar overlap quite often with the average of the other series.

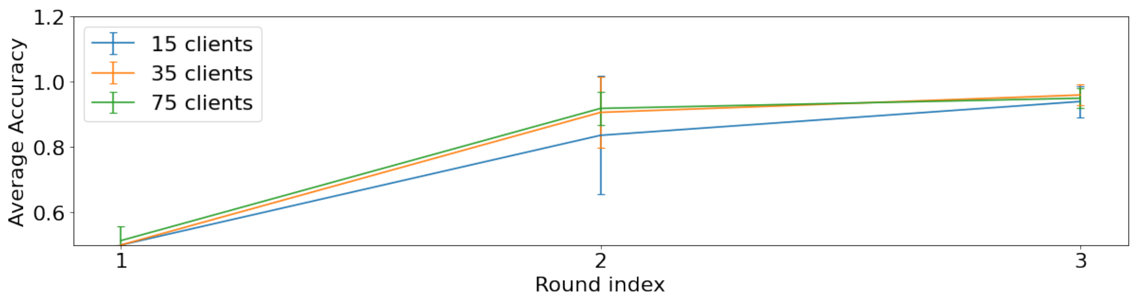


(a) Accuracy evolution for 5, 30 and 50 total clients

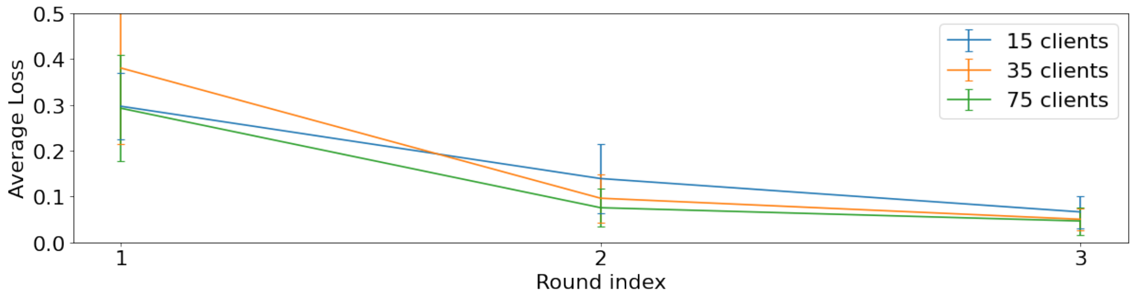


(b) Loss evolution for 5, 30 and 50 total clients

Figure D.1: Evolution of the accuracy D.3a and the loss D.3b for 3 training rounds



(a) Accuracy evolution for 15, 35 and 75 total clients

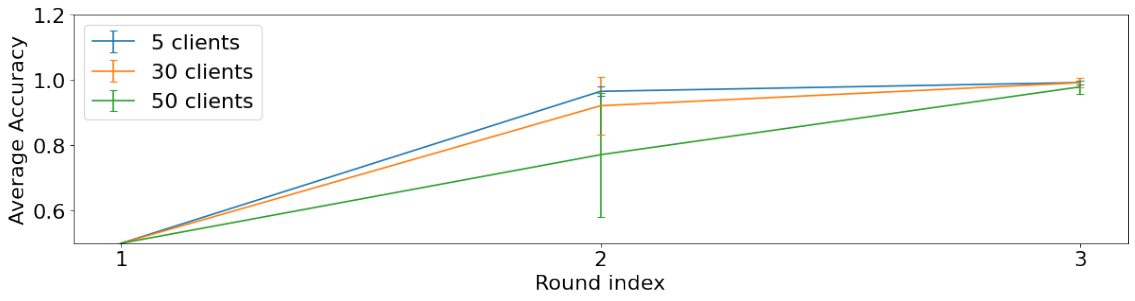


(b) Loss evolution for 15, 35 and 75 total clients

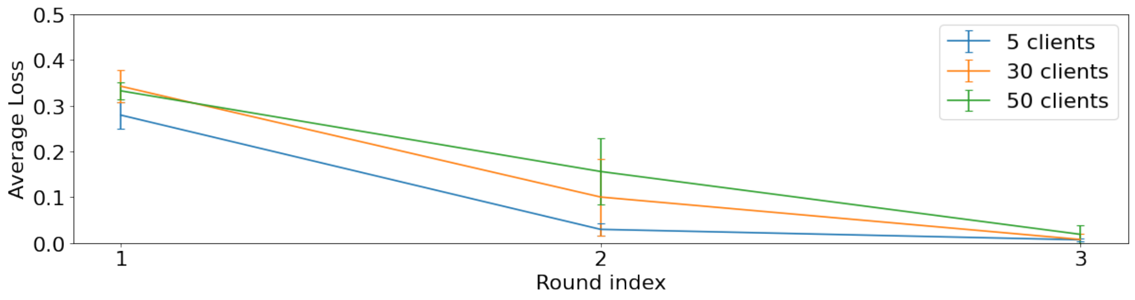
Figure D.2: Evolution of the accuracy D.2a and the loss D.2b for 3 training rounds

D.2 Influence of the number of retrained client per round

Figure D.3 and Figure D.4 present the evolution of the loos and the accuracy for a changing number of retrained clients and a constant number (80) clients in the pool and for three consecutive rounds of training. For the first round of the accuracy one round of

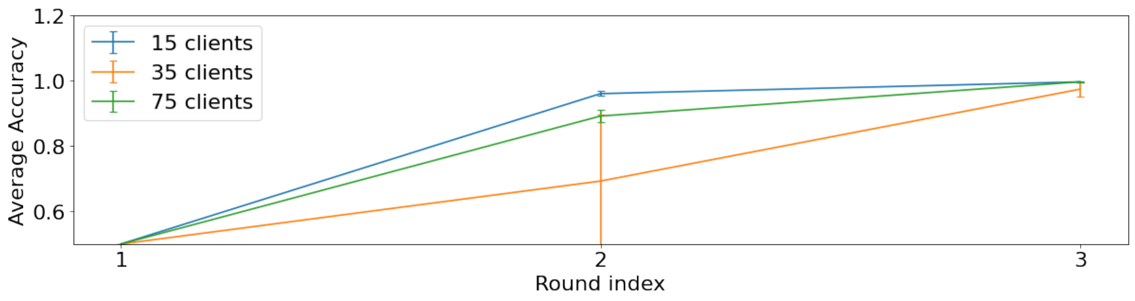


(a) Accuracy evolution for 5, 30 and 50 total clients

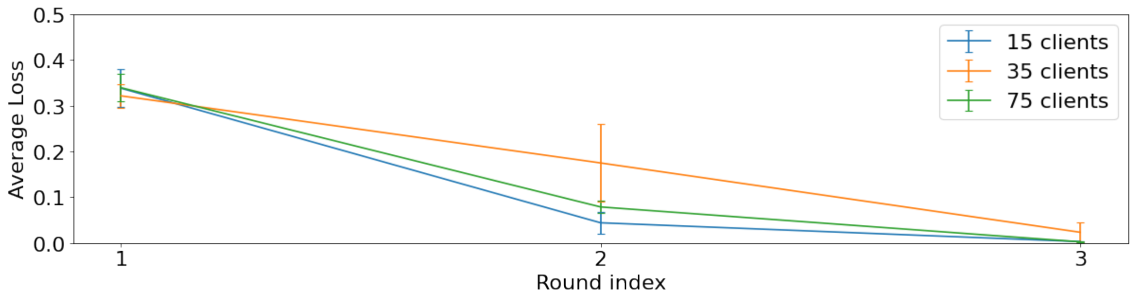


(b) Loss evolution for 5, 30 and 50 total clients

Figure D.3: Evolution of the accuracy D.3a and the loss D.3b for 3 training rounds



(a) Accuracy evolution for 15, 35 and 75 total clients



(b) Loss evolution for 15, 35 and 75 total clients

Figure D.4: Evolution of the accuracy D.4a and the loss D.4b for 3 training rounds

APPENDIX E

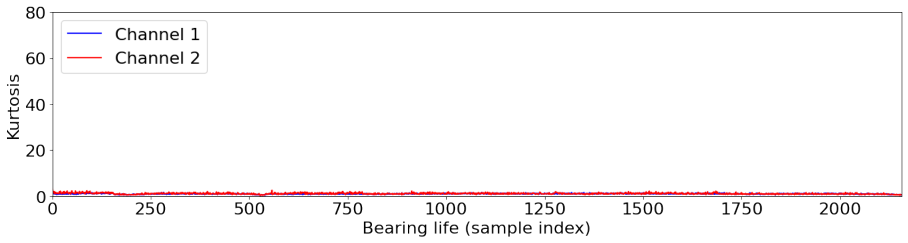
KURTOSIS EVOLUTION OVER BEARING LIFE FOR NASA DATASET

The following pages present the evolution of the Kurtosis for the 3 runs to failure test form [38] As a reminder, table E.1 presents the final state of each bearing for each test.

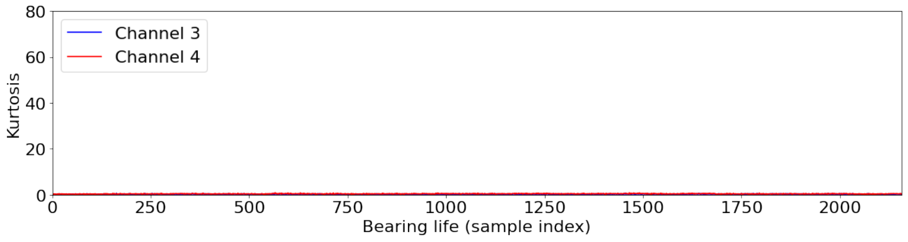
Table E.1: Summary of the defect observed on the dataset

Test #	**Bearing #**	**Fault Type**
Test 1	Bearing 1	No defect
-	Bearing 2	No defect
-	Bearing 3	Inner race defect
-	Bearing 4	Rolling element defect
Test 2	Bearing 1	Outer race defect
-	Bearing 2	No defect
-	Bearing 3	No defect
-	Bearing 4	Outer race defect
Test 3	Bearing 1	No defect
-	Bearing 2	No defect
-	Bearing 3	Outer race defect
-	Bearing 4	No defect

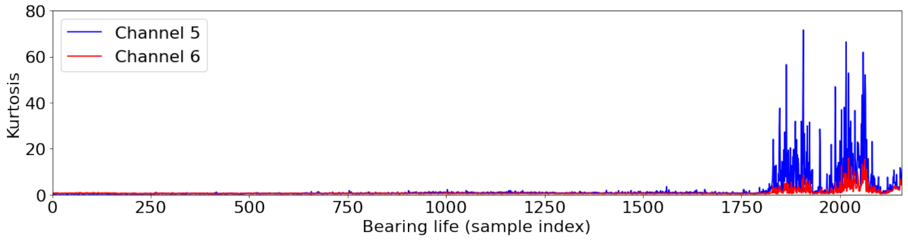
E.1 Kurtosis evolution over time for the 3 bearing run to failure test of [38]



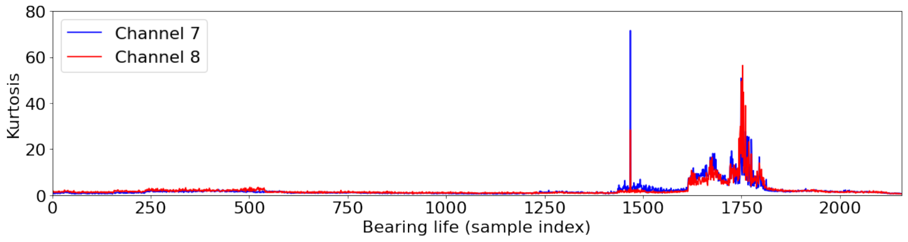
(a)



(b)



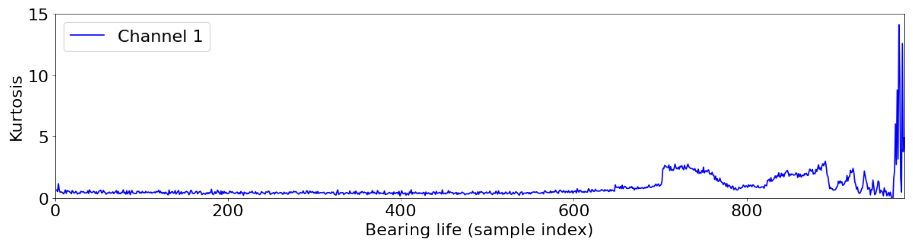
(c)



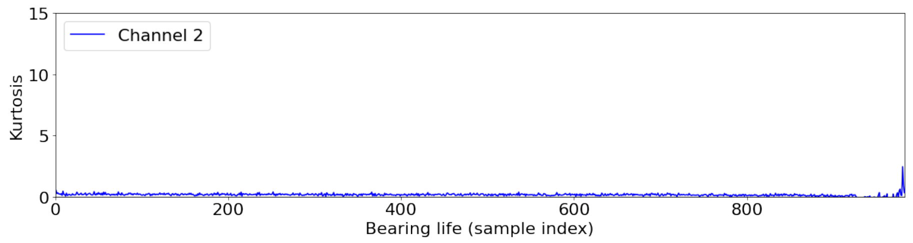
(d)

Figure E.1: Evolution of the Kurtosis over time for the first test of the NASA Dataset. E.1a bearing 1, no defect appeared, E.1b bearing 2, no defect appeared, E.1c bearing 3, an inner race appeared, E.1d bearing 4, a rolling element defect appeared.

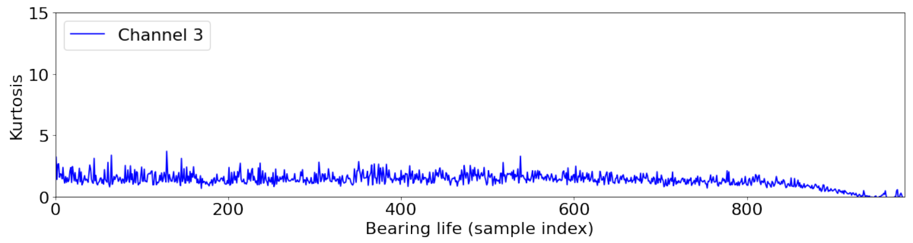
E.2 Kurtosis evolution over time for the 3 bearing run to failure test of [38]



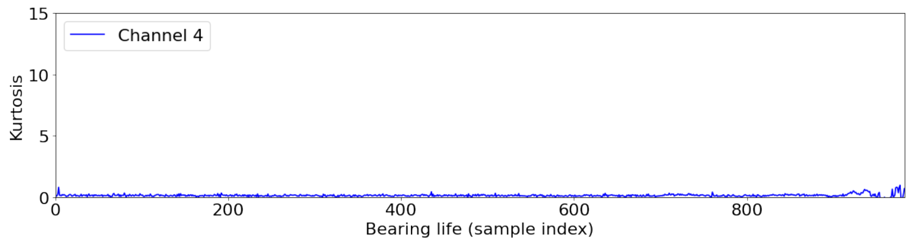
(a)



(b)



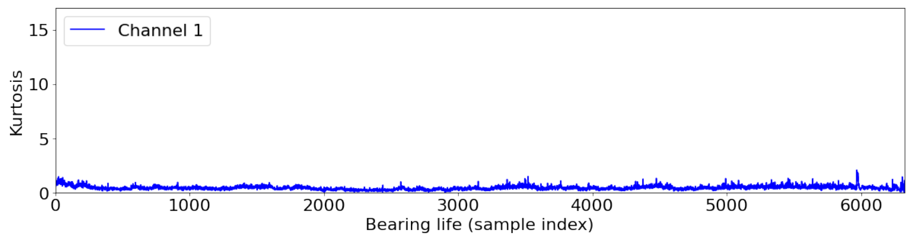
(c)



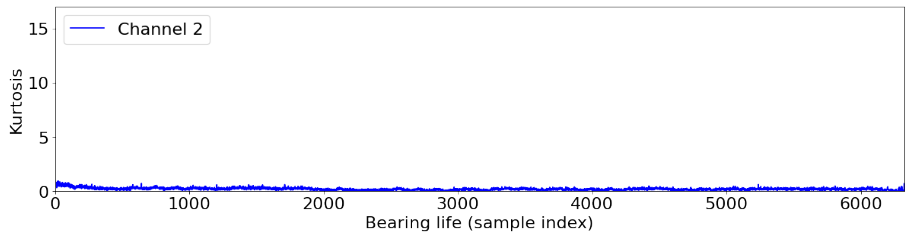
(d)

Figure E.2: Evolution of the Kurtosis over time for the second test of the NASA Dataset. E.2a bearing 1, an outer defect appeared, E.2b bearing 2, no defect appeared, E.2c bearing 3, no defect appeared, E.2d bearing 4, no defect appeared.

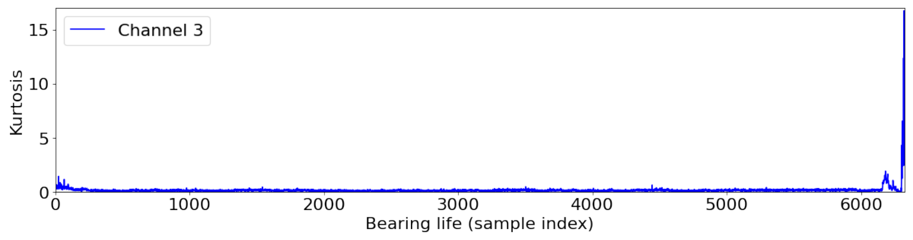
E.3 Kurtosis evolution over time for the 3 bearing run to failure test of [38]



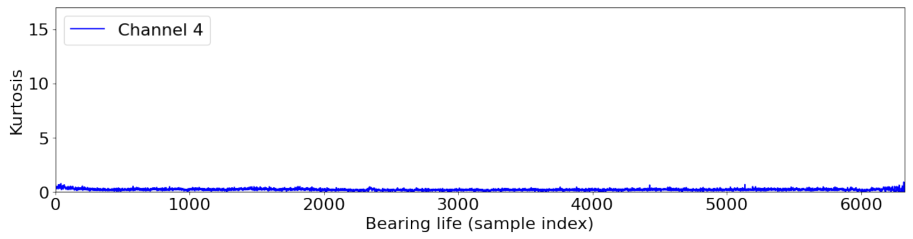
(a)



(b)



(c)



(d)

Figure E.3: Evolution of the Kurtosis over time for the second test of the NASA Dataset. (E.2a) bearing 1, no defect appeared, (E.2b) bearing 2, no defect appeared, (E.2c) bearing 3, an outer race defect appeared, (E.2d) bearing 4, no defect appeared.

APPENDIX F

CENTRALIZED LEARNING ON NASA DATASET

F.1 Model 0

Accuracy of Model 0 on the NASA Dataset for different families of discrete wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
coif14	0.7813	sym12	0.6031	sym10	0.5813	bior3.1	0.5641
bior3.9	0.7391	bior2.4	0.6031	sym7	0.5813	sym15	0.5625
bior3.3	0.7367	db28	0.6000	db7	0.5781	bior1.3	0.5625
rbio3.1	0.6836	db29	0.6000	sym13	0.5781	sym6	0.5625
bior3.7	0.6500	db38	0.6000	bior4.4	0.5781	sym17	0.5594
rbio3.5	0.6469	db13	0.5969	db16	0.5750	coif6	0.5594
rbio3.3	0.6375	sym14	0.5969	bior2.6	0.5750	rbio1.1	0.5594
db36	0.6344	db19	0.5938	rbio2.8	0.5750	sym2	0.5563
coif15	0.6250	sym3	0.5938	rbio2.6	0.5750	sym16	0.5563
dmey	0.6250	coif16	0.5938	bior1.1	0.5750	rbio1.5	0.5563
db24	0.6188	db11	0.5938	db10	0.5719	db4	0.5500
coif17	0.6156	sym20	0.5906	db23	0.5719	bior5.5	0.5469
rbio3.7	0.6156	coif3	0.5906	db6	0.5719	rbio6.8	0.5438
bior2.2	0.6156	sym19	0.5875	coif9	0.5719	coif5	0.5406
db37	0.6156	db9	0.5875	db3	0.5719	bior1.5	0.5406
db21	0.6125	sym18	0.5875	db15	0.5719	db34	0.5367
bior2.8	0.6125	coif1	0.5875	sym9	0.5719	sym4	0.5000
db30	0.6125	db25	0.5875	db22	0.5719	bior3.5	0.5000
db27	0.6125	rbio1.3	0.5875	db12	0.5719	rbio4.4	0.5000
db35	0.6125	db31	0.5875	db8	0.5688	db5	0.5000
coif7	0.6094	haar	0.5875	rbio5.5	0.5688	coif11	0.5000
sym11	0.6072	db18	0.5844	db14	0.5688	db20	0.5000
coif13	0.6063	coif4	0.5844	rbio2.4	0.5688	coif8	0.5000
rbio3.9	0.6063	db1	0.5844	sym5	0.5688	rbio2.2	0.5000
coif10	0.6063	db32	0.5844	db33	0.5688	coif2	0.5000
db26	0.6063	bior6.8	0.5813	sym8	0.5656		
coif12	0.6031	db17	0.5813	db2	0.5656		

F.2 Model 1

Accuracy of Model 1 on the NASA Dataset for different families of discrete wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
coif14	0.7813	sym12	0.6031	sym10	0.5813	bior3.1	0.5641
bior3.9	0.7391	bior2.4	0.6031	sym7	0.5813	sym15	0.5625
bior3.3	0.7367	db28	0.6000	db7	0.5781	bior1.3	0.5625
rbio3.1	0.6836	db29	0.6000	sym13	0.5781	sym6	0.5625
bior3.7	0.6500	db38	0.6000	bior4.4	0.5781	sym17	0.5594
rbio3.5	0.6469	db13	0.5969	db16	0.5750	coif6	0.5594
rbio3.3	0.6375	sym14	0.5969	bior2.6	0.5750	rbio1.1	0.5594
db36	0.6344	db19	0.5938	rbio2.8	0.5750	sym2	0.5563
coif15	0.6250	sym3	0.5938	rbio2.6	0.5750	sym16	0.5563
dmey	0.6250	coif16	0.5938	bior1.1	0.5750	rbio1.5	0.5563
db24	0.6188	db11	0.5938	db10	0.5719	db4	0.5500
coif17	0.6156	sym20	0.5906	db23	0.5719	bior5.5	0.5469
rbio3.7	0.6156	coif3	0.5906	db6	0.5719	rbio6.8	0.5438
bior2.2	0.6156	sym19	0.5875	coif9	0.5719	coif5	0.5406
db37	0.6156	db9	0.5875	db3	0.5719	bior1.5	0.5406
db21	0.6125	sym18	0.5875	db15	0.5719	db34	0.5367
bior2.8	0.6125	coif1	0.5875	sym9	0.5719	sym4	0.5000
db30	0.6125	db25	0.5875	db22	0.5719	bior3.5	0.5000
db27	0.6125	rbio1.3	0.5875	db12	0.5719	rbio4.4	0.5000
db35	0.6125	db31	0.5875	db8	0.5688	db5	0.5000
coif7	0.6094	haar	0.5875	rbio5.5	0.5688	coif11	0.5000
sym11	0.6072	db18	0.5844	db14	0.5688	db20	0.5000
coif13	0.6063	coif4	0.5844	rbio2.4	0.5688	coif8	0.5000
rbio3.9	0.6063	db1	0.5844	sym5	0.5688	rbio2.2	0.5000
coif10	0.6063	db32	0.5844	db33	0.5688	coif2	0.5000
db26	0.6063	bior6.8	0.5813	sym8	0.5656		
coif12	0.6031	db17	0.5813	db2	0.5656		

F.3 Model 2

Accuracy of Model 2 on the NASA Dataset for different families of discrete wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
bior3.5	0.6656	db24	0.6094	db15	0.5906	db16	0.5781
bior3.9	0.6654	coif9	0.6094	db34	0.5906	sym12	0.5781
rbio3.7	0.6563	db18	0.6094	db10	0.5906	haar	0.5781
coif12	0.6531	coif10	0.6094	db13	0.5906	sym2	0.5781
rbio3.9	0.6531	sym17	0.6094	sym10	0.5906	rbio5.5	0.5781
coif14	0.6500	sym16	0.6094	rbio4.4	0.5906	db37	0.5750
rbio3.5	0.6469	db25	0.6063	rbio1.3	0.5906	sym14	0.5750
db12	0.6377	bior4.4	0.6031	sym9	0.5906	db2	0.5719
rbio3.1	0.6375	db21	0.6031	bior1.3	0.5906	coif8	0.5719
rbio3.3	0.6375	coif4	0.6031	sym3	0.5906	db33	0.5719
db29	0.6313	db32	0.6031	db11	0.5875	db5	0.5688
coif17	0.6313	dmey	0.6031	sym6	0.5875	db4	0.5656
bior3.7	0.6281	sym18	0.6031	sym4	0.5875	rbio2.6	0.5656
bior2.8	0.6281	db38	0.6000	rbio6.8	0.5875	sym7	0.5656
coif15	0.6250	bior5.5	0.6000	sym5	0.5867	sym11	0.5625
db35	0.6250	coif6	0.6000	coif2	0.5844	db20	0.5594
db31	0.6219	rbio2.8	0.6000	sym13	0.5844	db9	0.5500
bior2.2	0.6219	rbio1.1	0.6000	db3	0.5813	coif1	0.5438
db27	0.6188	bior2.4	0.5969	db19	0.5813	sym8	0.5438
coif11	0.6188	coif7	0.5969	coif5	0.5813	bior1.5	0.5375
db28	0.6188	bior6.8	0.5969	sym15	0.5813	bior3.1	0.5178
db26	0.6188	rbio2.2	0.5969	rbio1.5	0.5813	db23	0.5000
sym19	0.6188	db30	0.5938	coif3	0.5781	db14	0.5000
coif16	0.6156	db22	0.5938	db17	0.5781	bior3.3	0.5000
db36	0.6125	bior1.1	0.5938	db7	0.5781	sym20	0.5000
coif13	0.6125	db1	0.5938	db8	0.5781		
rbio2.4	0.6125	bior2.6	0.5906	db6	0.5781		

F.4 Model 3

Accuracy of Model 3 on the NASA Dataset for different families of discrete wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
rbio3.1	0.8223	db9	0.6906	db17	0.6625	bior1.3	0.6344
bior3.3	0.8150	db26	0.6906	db2	0.6625	rbio2.2	0.6313
bior2.8	0.7498	rbio1.5	0.6875	db7	0.6563	coif7	0.6313
coif17	0.7406	sym16	0.6875	sym18	0.6563	db1	0.6281
bior3.5	0.7406	sym3	0.6844	sym20	0.6563	rbio4.4	0.6281
bior3.7	0.7373	coif9	0.6844	rbio2.8	0.6563	db15	0.6250
bior2.4	0.7371	db18	0.6813	db29	0.6563	coif5	0.6250
coif12	0.7250	sym7	0.6813	bior1.1	0.6563	haar	0.6219
coif10	0.7250	sym5	0.6781	db34	0.6531	db19	0.6156
db27	0.7219	rbio1.3	0.6781	coif14	0.6531	db22	0.6156
bior2.2	0.7188	db14	0.6750	bior3.1	0.6504	sym4	0.6156
coif6	0.7188	db24	0.6750	sym8	0.6500	coif1	0.6156
db38	0.7156	coif15	0.6750	db35	0.6500	sym2	0.6154
coif16	0.7125	db16	0.6750	db33	0.6469	rbio2.4	0.6125
dmey	0.7094	coif13	0.6750	sym12	0.6469	sym6	0.6094
bior5.5	0.7094	rbio3.3	0.6748	db37	0.6469	rbio6.8	0.6063
sym14	0.7063	sym15	0.6719	rbio3.9	0.6469	sym11	0.6063
db21	0.7061	db32	0.6719	db8	0.6438	bior1.5	0.6000
rbio3.5	0.7031	sym19	0.6719	db36	0.6438	db28	0.5938
coif11	0.7031	db20	0.6688	db5	0.6438	sym13	0.5906
rbio3.7	0.7031	db30	0.6688	db31	0.6406	db23	0.5875
bior6.8	0.7031	db6	0.6656	bior4.4	0.6406	db4	0.5750
coif8	0.7031	db13	0.6656	coif4	0.6406	coif3	0.5688
bior3.9	0.7031	rbio1.1	0.6656	sym9	0.6375	rbio5.5	0.5656
db12	0.6969	coif2	0.6656	db3	0.6375	db10	0.5000
rbio2.6	0.6969	db11	0.6625	bior2.6	0.6375		
sym17	0.6938	sym10	0.6625	db25	0.6344		

F.5 Model 4

Accuracy of Model 4 on the NASA Dataset for different families of discrete wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
bior3.3	0.8486	db15	0.6750	db8	0.6500	haar	0.6313
bior3.9	0.7586	db32	0.6750	db14	0.6469	sym4	0.6313
rbio3.1	0.7553	coif10	0.6750	db2	0.6469	rbio2.4	0.6250
coif12	0.7313	sym13	0.6719	sym5	0.6469	rbio2.6	0.6250
coif17	0.7219	db17	0.6688	db28	0.6438	db3	0.6219
coif16	0.7188	db37	0.6688	db23	0.6438	bior6.8	0.6188
coif15	0.7125	db9	0.6688	bior2.2	0.6438	sym2	0.6133
coif13	0.7125	db38	0.6688	sym9	0.6438	db1	0.6125
rbio3.5	0.7125	db11	0.6688	rbio2.2	0.6438	sym6	0.6125
bior3.5	0.7094	db7	0.6656	rbio3.7	0.6438	rbio5.5	0.6094
rbio3.3	0.7090	coif5	0.6656	rbio6.8	0.6438	sym7	0.6063
db22	0.7063	db26	0.6656	db4	0.6406	db19	0.6000
db25	0.7031	db13	0.6656	bior1.3	0.6406	rbio1.5	0.6000
coif11	0.7031	sym19	0.6656	sym3	0.6406	sym10	0.6000
db35	0.7031	sym8	0.6656	sym18	0.6406	bior1.5	0.5969
sym17	0.7000	rbio2.8	0.6656	db31	0.6375	db24	0.5938
dmey	0.6969	rbio3.9	0.6656	coif4	0.6375	bior5.5	0.5938
sym16	0.6938	bior2.6	0.6652	sym14	0.6375	db29	0.5813
db27	0.6875	coif6	0.6625	rbio4.4	0.6375	db16	0.5781
coif8	0.6875	sym11	0.6625	bior2.4	0.6344	sym20	0.5719
coif1	0.6875	rbio1.3	0.6625	coif9	0.6344	coif3	0.5688
db30	0.6844	coif7	0.6594	db18	0.6344	db10	0.5688
db33	0.6844	bior2.8	0.6594	bior1.1	0.6344	coif14	0.5656
bior3.7	0.6840	db5	0.6563	sym15	0.6344	coif2	0.5654
db36	0.6813	rbio1.1	0.6563	sym12	0.6344	bior3.1	0.5129
db21	0.6781	db12	0.6531	db20	0.6313		
db34	0.6781	db6	0.6500	bior4.4	0.6313		

F.6 Model 5

Accuracy of Model 5 on the NASA Dataset for different families of discrete wavelet transforms

wavelet	accuracy	wavelet	accuracy	wavelet	accuracy	wavelet	accuracy
bior3.3	0.7482	db37	0.6219	coif7	0.6031	bior5.5	0.5906
bior3.5	0.6781	db34	0.6219	rbio2.6	0.6000	db9	0.5875
rbio3.9	0.6688	coif8	0.6219	sym12	0.6000	db11	0.5875
coif15	0.6625	db24	0.6219	rbio1.3	0.6000	rbio1.1	0.5844
db6	0.6592	bior2.6	0.6219	sym3	0.6000	sym6	0.5844
bior3.7	0.6563	sym13	0.6188	db16	0.6000	db19	0.5844
rbio3.3	0.6500	db29	0.6188	db23	0.5969	haar	0.5844
db38	0.6438	db32	0.6188	db4	0.5969	db13	0.5813
rbio3.1	0.6438	coif13	0.6188	sym11	0.5969	rbio4.4	0.5813
rbio3.7	0.6375	db30	0.6188	rbio6.8	0.5969	sym9	0.5813
db36	0.6375	coif4	0.6188	sym2	0.5969	db7	0.5813
rbio3.5	0.6375	db27	0.6188	bior2.4	0.5969	db15	0.5813
db22	0.6375	sym19	0.6156	sym17	0.5969	rbio2.2	0.5813
db33	0.6344	bior6.8	0.6125	sym10	0.5938	coif3	0.5781
bior2.8	0.6344	sym15	0.6125	db21	0.5938	db2	0.5781
coif16	0.6313	db14	0.6125	db25	0.5938	sym7	0.5781
coif10	0.6313	sym14	0.6094	db18	0.5938	bior4.4	0.5781
bior3.9	0.6313	coif11	0.6094	db1	0.5938	bior1.3	0.5781
sym18	0.6281	sym16	0.6063	coif2	0.5938	sym8	0.5750
coif6	0.6250	db3	0.6063	rbio2.4	0.5906	sym4	0.5750
db31	0.6250	db35	0.6063	db12	0.5906	bior1.5	0.5750
dmey	0.6250	coif5	0.6063	rbio1.5	0.5906	sym5	0.5719
coif12	0.6250	db10	0.6063	bior1.1	0.5906	db5	0.5656
bior2.2	0.6250	sym20	0.6063	db26	0.5906	rbio5.5	0.5625
coif9	0.6219	coif14	0.6031	db17	0.5906	bior3.1	0.5354
db20	0.6219	rbio2.8	0.6031	db28	0.5906		
coif1	0.6219	coif17	0.6031	db8	0.5906		

REFERENCES

[1] *Unlocking the potential of the internet of things*, https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world, Article, Accessed: 2019-11-24, 2015.

[2] *Léonard de vinci mourrait il y a 500 ans: Ses 5 inventions les moins connues*, https://www.arcinfo.ch/articles/lifestyle/techno-et-sciences/leonard-de-vinci-mourrait-il-y-a-500-ans-on-vous-presente-ses-5-inventions-les-moins-connues-835293, Website, Accessed : 2020-09-25.

[3] S. Nandi, H. A. Toliyat, and X. Li, "Condition monitoring and fault diagnosis of electrical motors—a review," *IEEE Transactions on Energy Conversion*, vol. 20, no. 4, pp. 719–729, 2005.

[4] M. M. Khonsari and E. R. Booser, *Applied Tribology Bearing Design and Lubrication*. John Wiley & Sons, 2008.

[5] *Mobius institute - rolling element bearings*, https://www.mobiusinstitute.com/site2/default.asp, Website, Accessed : 2020-04-07.

[6] X. Jin, D. Siegel, B. Weiss, E. Gamel, W. Wang, J. Lee, and J. Ni, "The present status and future growth of maintenance in us manufacturing: Results from a pilot survey," *Manufacturing Review*, vol. 3, p. 10, Jan. 2016.

[7] G. Morales-Espejel and A. Gabelli, "A major step forward in life modelling," *SKF Evolution*, vol. 4, pp. 21–27, 2015.

[8] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *Internet of Things Journal, IEEE*, vol. 1, pp. 349–359, Aug. 2014.

[9] W. Lidong and W. Guanghui, "Big Data in Cyber-Physical Systems, Digital Manufacturing and Industry 4.0," *International Journal of Engineering and Manufacturing*, vol. 6, no. 4, pp. 1–8, 2016.

[10] P. D. Urbina Coronado, R. Lynn, W. Louhichi, M. Parto, and E. Wescoat, “Part data integration in the Shop Floor Digital Twin: Mobile and cloud technologies to enable a manufacturing execution system,” *Journal of Manufacturing Systems*, vol. 48, pp. 25–33, 2018.

[11] *Indoor and outdoor geolocation based on ai for industry*, https://www.zozio.tech/en/, Website, Accessed : 2020-10-02.

[12] R. B. Randall, *Vibrations Based Condition Monitoring*. John Wiley & Sons, 2012.

[13] *Mqtt.org*, http://mqtt.org/, Website, Accessed : 2020-04-07.

[14] *Mtconnect.org*, https://www.mtconnect.org/, Website, Accessed : 2020-04-07.

[15] D. Wu, S. Liu, L. Zhang, J. Terpenny, R. X. Gao, T. Kurfess, and J. A. Guzzo, “A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing,” *Journal of Manufacturing Systems*, vol. 43, pp. 25–34, 2017.

[16] P. Rauby, “Developing a smart and low cost device for machining vibration analysis,” Ph.D. dissertation, The George W. Woodruff School of Mechanical Engineering, GeorgiaTech, 2018.

[17] *What is arduino?* https://www.arduino.cc/en/Guide/Introduction, Website, Accessed : 2021-06-25.

[18] *Socs*, https://www.espressif.com/en/products/socs, Website, Accessed : 2021-06-25.

[19] *Our brief history*, https://www.particle.io/about-particle/, Website, Accessed : 2021-06-25.

[20] *About us*, https://www.pjrc.com/about/about_us.html, Website, Accessed : 2021-06-25.

[21] *Sparkfun products*, https://www.sparkfun.com/products, Website, Accessed : 2021-06-25.

[22] *Nvidia jetson nano 2gb developer kit*, https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit, Website, Accessed : 2021-06-25.

[23] *Beaglebone ai*, `https://beagleboard.org/ai`, Website, Accessed : 2021-06-25.

[24] M. Elangovan, V. Sugumaran, K. I. Ramachandran, and S. Ravikumar, "Effect of SVM kernel functions on classification of vibration signals of a single point cutting tool," *Expert Systems with Applications*, vol. 38, no. 12, pp. 15 202–15 207, 2011.

[25] C. Drouillet, J. Karandikar, C. Nath, A.-C. Journeaux, M. El Mansori, and T. Kurfess, "Tool life predictions in milling using spindle power with the neural network technique," *Journal of Manufacturing Processes*, vol. 22, pp. 161–168, 2016.

[26] Y. Fu, Y. Zhang, Y. Gao, H. Gao, T. Mao, H. Zhou, and D. Li, "Machining vibration states monitoring based on image representation using convolutional neural networks," *Engineering Applications of Artificial Intelligence*, vol. 65, no. July, pp. 240–251, 2017.

[27] P. O'Donovan, C. Gallagher, K. Bruton, and D. T. O'Sullivan, "A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications," *Manufacturing Letters*, vol. 15, pp. 139–142, 2018.

[28] C.-A. Azencot, *Foundations of machine learning chapter 9: Tree-based approaches*, 2017.

[29] ——, *Foundations of machine learning chapter 10: Support vector machines*, 2017.

[30] F. Pérez-Cruz and O. Bousquet, "Kernel methods and their potential use in signal processing," *IEEE Signal Processing Magazine*, vol. 21, no. 3, pp. 57–65, 2004.

[31] F Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in . . .," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[32] C.-A. Azencot, *Foundations of machine learning chapter 11: Artificial neural networks*, 2017.

[33] W. Hoffmann, "Some experience with ferrography in monitoring the condition of aircraft engines," *Wear*, vol. 65, no. 3, pp. 307–313, 1981.

[34] X. Zhu, C. Zhong, and J. Zhe, *Lubricating oil conditioning sensors for online machine health monitoring – A review*, 2017.

[35] R. B. Randall and J. Antoni, “Rolling element bearing diagnostics—a tutorial,” *Mechanical Systems and Signal Processing*, vol. 25, no. 2, pp. 485 –520, 2011.

[36] N. J. Kessissoglou and Z. Peng, “Integrating Vibration and Oil Analysis for Machine Condition Monitoring,” *Practicing Oil Analysis*, no. MAR./APR. 2003.

[37] S. Al-Dossary, R. I. Hamzah, and D. Mba, “Observations of changes in acoustic emission waveform for varying seeded defect sizes in a rolling element bearing,” *Applied Acoustics*, 2009.

[38] J. Lee, H. Qiu, J. Lin, and R. T. Services, *”bearing dataset”*, NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA, 2007.

[39] B. Zhang, S. Zhang, and W. Li, “Bearing performance degradation assessment using long short-term memory recurrent network,” *Computers in Industry*, vol. 106, pp. 14–29, 2019.

[40] P. E. William and M. W. Hoffman, “Identification of bearing faults using time domain zero-crossings,” *Mechanical Systems and Signal Processing*, vol. 25, no. 8, pp. 3078–3088, 2011.

[41] A. Trip and J. E. Wieringa, “Individuals charts and additional tests for changes in spread,” *Quality and Reliability Engineering International*, vol. 22, no. 3, pp. 239–249, 2006.

[42] E. S. PAGE, “CONTROL CHARTS WITH WARNING LINES,” *Biometrika*, vol. 42, no. 1-2, pp. 243–257, Jun. 1955.

[43] C. Jw and J Tukey, “An algorithm for the machine calculation of complex fourier series,” *Math Comput*, vol. 19, pp. 297–301, 1965.

[44] E. Prevost, “Detection of bearing defects with approximate bearing configuration,” Ph.D. dissertation, The George W. Woodruff School of Mechanical Engineering, GeorgiaTech, 2019.

[45] A. Oppenheim and R. Schafer, “From frequency to quefrency: A history of the cepstrum,” *Signal Processing Magazine, IEEE*, vol. 21, pp. 95 –106, Oct. 2004.

[46] H. Konstantin-Hansen and H. Herlufsen, *Envelope and cepstrum analyses for machinery fault identification*, 2010.

[47] G. D. White, *Introduction to Machine Vibration*. DLI Engineering Corp, 1993.

[48] J. Antoni and R. B. Randall, “The spectral kurtosis: Application to the vibratory surveillance and diagnostics of rotating machines,” *Mechanical Systems and Signal Processing*, vol. 20, no. 2, pp. 308–331, 2006.

[49] N. Sawahli, “Diagnostics, prognostics and fault simulation for rolling element bearings,” Ph.D. dissertation, The University of New South Wales, School of Mechanical and Manufacturing Engineering, 2007.

[50] D. Gabor, “Theory of communication. part 1: The analysis of information,” *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[51] J. N. Kutz, *Data-Driven Modeling & Scientific Computation*. Oxford, 2013.

[52] S. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[53] G. Strang and T. Nguyen, *Wavelets and filter banks*. SIAM, 1996.

[54] P. Goel, M. Chandra, A. Anand, and A. Kar, “An improved wavelet-based signal-denoising architecture with less hardware consumption,” *Applied Acoustics*, vol. 156, pp. 120–127, 2019.

[55] C. V. Garzón, G. B. Moncayo, D. H. Alcantara, and R. Morales-Menendez, “Fault Detection in Spindles using Wavelets - State of the Art,” *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 450–455, 2018.

[56] A. Kumar, Y. Zhou, C. Gandhi, R. Kumar, and J. Xiang, “Bearing defect size assessment using wavelet transform based deep convolutional neural network (dcnn),” *Alexandria Engineering Journal*, vol. 59, no. 2, pp. 999–1012, 2020.

[57] B. T. Holm-Hansen, R. X. Gao, and L. Zhang, “Customized wavelet for bearing defect detection,” *Journal of dynamic systems, measurement, and control*, vol. 126, no. 4, pp. 740–745, 2004.

[58] P. Kankar, S. C. Sharma, and S. Harsha, “Rolling element bearing fault diagnosis using wavelet transform,” *Neurocomputing*, vol. 74, no. 10, pp. 1638–1645, 2011.

[59] B. Paya, I. Esat, and M. Badi, “Artificial neural network based fault diagnostics of rotating machinery using wavelet transforms as a preprocessor,” *Mechanical Systems and Signal Processing*, vol. 11, no. 5, pp. 751–765, 1997.

[60] O. José, L. Castro, C. C. Sisamón, J. Carlos, and G. Prada, “Bearing Fault Diagnosis based on Neural Network Classification and Wavelet Transform,” pp. 22–29, 2006.

[61] H. Wang, J. Chen, and G. Dong, “Feature extraction of rolling bearing’s early weak fault based on EEMD and tunable Q-factor wavelet transform,” *Mechanical Systems and Signal Processing*, vol. 48, no. 1-2, pp. 103–119, 2014.

[62] Y. Wang, Z. He, and Y. Zi, “Enhancement of signal denoising and multiple fault signatures detecting in rotating machinery using dual-tree complex wavelet transform,” *Mechanical Systems and Signal Processing*, vol. 24, no. 1, pp. 119–137, 2010.

[63] N. G. Kingsbury, “The dual-tree complex wavelet transform: a new technique for shift invariance and directional filters,” paper 86, 1998.

[64] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, “The dual-tree complex wavelet transform,” *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[65] X. Zhang, Z. Liu, J. Wang, and J. Wang, “Time–frequency analysis for bearing fault diagnosis using multiple Q-factor Gabor wavelets,” *ISA Transactions*, vol. 87, pp. 225–234, 2019.

[66] Y. Zhang, K. Xing, R. Bai, D. Sun, and Z. Meng, “An enhanced convolutional neural network for bearing fault diagnosis based on time–frequency image,” *Measurement: Journal of the International Measurement Confederation*, vol. 157, p. 107 667, 2020.

[67] M. M. Islam and J. M. Kim, “Automated bearing fault diagnosis scheme using 2D representation of wavelet packet transform and deep convolutional neural network,” *Computers in Industry*, vol. 106, pp. 142–153, 2019.

[68] W. He, Q. Miao, M. Azarian, and M. Pecht, "Health monitoring of cooling fan bearings based on wavelet filter," *Mechanical Systems and Signal Processing*, vol. 64-65, pp. 149–161, 2015.

[69] S. Haidong, J. Hongkai, Z. Ke, W. Dongdong, and L. Xingqiu, "A novel tracking deep wavelet auto-encoder method for intelligent fault diagnosis of electric locomotive bearings," *Mechanical Systems and Signal Processing*, vol. 110, pp. 193–209, 2018.

[70] L. Gelman, B. Murray, T. H. Patel, and A. Thomson, "Vibration diagnostics of rolling bearings by novel nonlinear non-stationary wavelet bicoherence technology," *Engineering Structures*, vol. 80, pp. 514–520, 2014.

[71] Z. Liu, H. Cao, X. Chen, Z. He, and Z. Shen, "Multi-fault classification based on wavelet SVM with PSO algorithm to analyze vibration signals from rolling element bearings," *Neurocomputing*, vol. 99, pp. 399–410, 2013.

[72] M. Borova, M. Prauzek, J. Konecny, and K. Gaiova, "Environmental WSN Edge Computing Concept by Wavelet Transform Data Compression in a Sensor Node," *IFAC-PapersOnLine*, vol. 52, no. 27, pp. 246–251, 2019.

[73] *Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy*, https://obamawhitehouse.archives.gov/sites/default/files/privacy-final.pdf, Article, Accessed: 2021-09-24, 2012.

[74] K. Ebrahimi, G. F. Jones, and A. S. Fleischer, "A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities," *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 622–638, 2014.

[75] Microsoft, *Project natick*, https://natick.research.microsoft.com, blog post, Accessed : 2021-03-04.

[76] H. B. Mcmahan and D. Ramage, "Communication-Efficient Learning of Deep Networks from Decentralized Data," vol. 54, 2017. arXiv: arXiv:1602.05629v3.

[77] M. Chen and T. Ouyang, "Federated Learning Of Out-Of-Vocabulary Words," pp. 1–6, 2019. arXiv: arXiv:1903.10635v1.

[78] H. R. Roth, K. Chang, P. Singh, N. Neumark, W. Li, V. Gupta, S. Gupta, L. Qu, A. Ihsani, B. C. Bizzo, Y. Wen, V. Buch, M. Shah, F. Kitamura, M. Mendonça, V. Lavor, A. Harouni, C. Compas, J. Tetreault, P. Dogra, Y. Cheng, S. Erdal, R. White, B. Hashemian, T. Schultz, M. Zhang, A. McCarthy, B. M. Yun, E. Sharaf, K. V. Hoebel, J. B. Patel, B. Chen, S. Ko, E. Leibovitz, E. D. Pisano, L. Coombs, D. Xu, K. J. Dreyer, I. Dayan, R. C. Naidu, M. Flores, D. Rubin, and J. Kalpathy-Cramer, "Federated learning for breast density classification: A real-world implementation," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, vol. 12444, Cham: Springer International Publishing, 2020, pp. 181–191, ISBN: 9783030605476.

[79] A. Durrant, M. Markovic, D. Matthews, D. May, and J. Enright, "The Role of Cross-Silo Federated Learning in Facilitating Data Sharing in the Agri-Food Sector," arXiv: arXiv:2104.07468v1.

[80] M Parimala, S. P. R. M, Q.-v. Pham, and K. Dev, "Fusion of Federated Learning and Industrial Internet of Things : A Survey," pp. 1–24, 2021. arXiv: arXiv:2101.00798v1.

[81] K. Nandury, A. Mohan, and F. Weber, "Cross-silo federated training in the cloud with diversity scaling and semi-supervised learning," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3085–3089.

[82] R. Kanagavelu, Z. Li, J. Samsudin, S. Hussain, F. Yang, Y. Yang, R. S. M. Goh, and M. Cheah, "Federated learning for advanced manufacturing based on industrial iot data analytics," in *Implementing Industry 4.0: The Model Factory as the Key Enabler for the Future of Manufacturing*, C. Toro, W. Wang, and H. Akhtar, Eds. Cham: Springer International Publishing, 2021, pp. 143–176, ISBN: 978-3-030-67270-6.

[83] T. Hiessl, D. Schall, J. Kemnitz, and S. Schulte, "Industrial Federated Learning – Requirements and System Design," pp. 1–12, 2021. arXiv: arXiv:2005.06850v1.

[84] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2021.

[85] N. Ge, G. Li, L. Zhang, and Y. Liu, “Failure prediction in production line based on federated learning: an empirical study,” *Journal of Intelligent Manufacturing*, 2021.

[86] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, “Opportunities of federated learning in connected, cooperative, and automated industrial systems,” *IEEE Communications Magazine*, vol. 59, no. 2, pp. 16–21, 2021.

[87] M. Dhada and A. K. Parlikad, “Federated Learning for Collaborative Prognosis,” no. Copen, pp. 1–6, 2019.

[88] W. Zhang, X. Li, H. Ma, Z. Luo, and X. Li, “Federated learning for machinery fault diagnosis with dynamic validation and self-supervision,” *Knowledge-Based Systems*, vol. 213, p. 106 679, 2021.

[89] F. RasperryPi, *About us*, `https://www.raspberrypi.org/about/`, Website, Accessed : 2021-02-01.

[90] F. BeagleBoad, *About us*, `https://beagleboard.org/about`, Website, Accessed : 2021-02-01.

[91] T. Instrument, *Am572x, block diagram*, `https://www.ti.com/product/AM5729`, Website, Accessed : 2021-02-01.

[92] Newark, *Beaglebone ai*, `https://www.newark.com/beagleboard/bbone-ai/sbc-beagle-bone-ai-am5729-plus/dp/10AH2651`, Website, Accessed : 2021-02-01.

[93] ——, *Beaglebone black*, `https://www.newark.com/element14/bbone-black-wireless/beaglebone-black-wireless-rohs/dp/95AC0788`, Website, Accessed : 2021-02-01.

[94] *Tidl on beagleboneai*, `https://beagleboard.org/p/175809/tidl-on-beaglebone-ai-1ee263`, Website, Accessed : 2020-10-02.

[95] *Ring oscillator*, `https://pub.pages.cba.mit.edu/ring/`, Website, Accessed : 2021-10-02.

[96] R. Birkett, *Enhancing real-time capabilities with the pru*, `https://elinux.org/images/1/1c/Birkett--enhancing_rt_capabilities_with_the_pru.pdf`, Website, Accessed : 2021-02-01.

[97] T. Instrument, *Am572x, technical reference manual*, https://www.ti.com/lit/ug/spruhz6l/spruhz6l.pdf, Technical Documentation, Accessed : 2021-03-04.

[98] ——, *Processor sdk,remoteproc and rpmsg*, https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components/PRU-ICSS/Linux_Drivers/RemoteProc_and_RPMsg.html, Website, Accessed : 2021-02-01.

[99] ——, *Pru assembly language tools v2.1*, https://www.ti.com/lit/ug/spruhv6a/spruhv6a.pdf, Technical Documentation, Accessed : 2021-03-04.

[100] *Beaglebone ai am57x*, https://esys.ir/images/img_Item/1745/Files/BeagleBone-AI_sch.pdf, Technical Drawing, Accessed: 2021-05-03.

[101] *Sparkfun triple axis accelerometer breakout - kx132 (qwiic)*, https://www.sparkfun.com/products/17871, Website, Accessed : 2021-09-25.

[102] *Tidl api user's guide, introduction*, https://downloads.ti.com/mctools/esd/docs/tidl-api/intro.html, Website, Accessed : 2021-10-02.

[103] *Processor sdk linux, ti deep learning tidl*, https://software-dl.ti.com/processor-sdk-linux/esd/docs/05_00_00_15/linux/Foundational_Components_TIDL.html, Website, Accessed : 2021-10-02.

[104] *Texasinstruments edgeai*, https://github.com/TexasInstruments/edgeai, Website, Accessed : 2021-10-02.

[105] *Tensorflow*, https://www.tensorflow.org, Website, Accessed : 2021-10-02.

[106] *Introduction to tensorflow*, https://www.tensorflow.org/learn, Website, Accessed : 2021-10-02.

[107] *Machine fault simulator*, https://spectraquest.com/machinery-fault-simulator/details/mfs, Website, Accessed : 2021-09-25.

[108] G. Gautier, R. Serra, and J.-M. Mencik, “Roller bearing monitoring by new subspace-based damage indicator,” *Shock and Vibration*, vol. 2015, Aug. 2015.