POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Meccanica (LM-33)

Tesi di Laurea Magistrale

Development of a machine learning-based automated system for the detection of rub in gas turbines

Relatore
Prof. Eliodoro Chiavazzo, PoliTo
Ing. Alejandro Silva Bernandez, ETSI UPM
Prof. Stefano Marchesiello, PoliTo
Prof. Muñoz Guijosa Juan Manuel, ETSI UPM

Candidato
Antonio Squicciarini

October 2020

Abstract

During the last decades, there was an increase in the net efficiency and performance of turbomachines, attained through minimization of component mass, size, and dimensional tolerances, the use of cutting-edge materials with top-class mechanical and thermal properties, and the introduction of the newest cooling and lubrication technologies. However, the higher performance came inevitably with a trade-off in reliability, since new designs and technologies made the turbomachines more vulnerable to malfunction. A breakdown during the operation can carry over high economic loss or even human losses. Hence, the study and the development of preventive maintenance programs that allow for early malfunction detection are increasingly important in today's industry.

The reduction in backlash between the stator and the rotor has made it possible to minimize flow losses inside the turbines. On the order side, this structural improvement has made more likely the occurrence of contact between stator and rotor. The rotor-casing rub is one of the most common failures in rotating machinery. This phenomenon causes wear, overheating, and variation in machine performance, and, when unnoticed, it can compromise the entire mechanical system. In most gas and steam turbines, the rub is easily detected, thanks to the possibility to install displacement sensors in the hydrodynamic shaft supports. However, in aeroderivative gas turbines, due to different construction solutions, accelerometers on their casing are the only means available for condition monitoring.

The aim of this Master Thesis is the development of a Machine Learning program based on a "Deep Neural Network" architecture for the very early detection of rotor-casing single-point rub in aeroderivative gas turbines with accelerometers on the casing, most often the only means available for their monitoring of this engine type. The net has been trained only with data generated by a Finite-Element Numerical Model of a rotating machine. To carry out an optimized model, different combinations of data processing and neural network architecture solutions have been tested and compared to evaluate the prediction performances. Then, in order to understand if the software is able to well generalize the information contained into the acceleration signal, the best models have been proved with a database created with an experimental rig reproducing of a gas turbine with flexible casing and rotor-casing single-point rub.

Acknowledgements

Vorrei dedicare questo spazio del mio elaborato per ricordare tutte le persone che, grazie al loro supporto e affetto, mi sono state vicine durante tutto questo percorso di laurea.

In primis, vorrei ringraziare tutti i familiari i quali, grazie alla loro vicinanza, mi hanno sempre aiutato durante questi ultimi 5 anni lontano da casa.

Un agradecimiento especial a todo mi maravilloso equipo de investigación de la ETSII Politécnica de Madrid - Silva Alejandro, Mateos de Porras Cosano Antonio, Alejandro Zarzo, Juan M. Munoz-Guijosa, Gguillen Carlos - por el apoyo fundamental y constante que me han dado durante estos últimos y difíciles seis meses. Sin ellos todo esto no habría sido posible.

Un ringraziamento ai relatori del progetto di tesi del Politecnico di Torino, il Prof. Eliodoro Chiavazzo e Prof. Stefano Marchesiello.

Un ringraziamento a tutti i miei amici, reputo di essere una persona molto fortunata e privilegiata di essere circondato da tante persone meravigliose su cui potrò sempre contare.

Grazie anche ai miei cari coinquilini di Torino, con i quali siamo sopravvissuti a quattro anni importantissimi e bellissimi della nostra vita.

Ringrazio i miei amici universitari, con i quali abbiamo condiviso questo lungo e stupendo viaggio.

A special thanks to all the friends I have met in this last unique and magic Erasmus year in Madrid.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Framework of the project

The Mechanical Engineering Department of the *Universidad Politecnica de Madrid* is carrying out a research n the field of mechanical systems monitoring, with the aim of improving the prevention maintenance of *Aeroderivative Gas Turbines*. The doctoral student Alejandro Silva Bernárdez is working on a procedure to detect single-point rub in its very early stages. The aim of his Doctoral Thesis is the development of a signal-processing methodology for the very early detection of rotor-casing single-point rub in aeroderivative gas turbines. The signal used is generated by only the accelerometers located on the outside of the turbine casing, being this the only technological monitoring solution available with this type of engine. The monitoring procedure is tested and different detection methods compared through a Finite-Element mathematical model of an aeroderivative gas turbine, which allows the generation of syntactic accelerometer data for operating conditions with and without rub failure. The best procedure have to detect very weak rotor-stator contact at very small time scales and be implementable in real time on a real machine.
This thesis project is plugged in this framework, and it has the objective to developing a new monitoring method that uses a Deep Neural Network to performs an online classification prediction about the operating status of the machine, to detect the possible presence of the rub malfunctioning. The correct operation of the network will then be validated with the data of an experimental rotor rig with flexible casing monitored with both accelerometers and proximity sensors.

1.2 Turbine rub: an overview of the problem

One of the most important design's parameters of a gas turbine is the clearance between the rotor and stator. To maximize the efficiency and power output in modern aeroengines it is supposed to be kept as small as possible at specific positions of the engine, for example between different stages or at the tip of bladed wheels. [1]. The coefficient η_v "volumetric efficiency" [12] is used to evaluate the leaks of the flow, by multiplying this coefficient with the ideal power generated by the turbine. It is not always possible to determine the exact value of η_v, due to the several parts of the engine where gas leaks can occur. Others consider also different $\eta_v^{(i)}$ for each stage of the turbine. η_v is defined as the flow rate of fluid that passes through the blades and the total flow that passes from one stage to the subsequent.
As a consequence of the clearance reduction, the impact between the pales of the rotor's blades and stator's seals is more likely to happen, which may cause the variation of engine performance, undermine the safety of the turbine, and even lead to catastrophic failures [2]. In 1973, the National Transportation Safety Board (NTSB) reported a case that one of the engine fan assemblies was disintegrated during the flight for the interaction between the fan blade tip and the fan casing. In June 23, 2014, F-35A engine fired and leaded to a fleetwide grounding due to the excessive rubbing between the turbine blades and the cowling occurred.

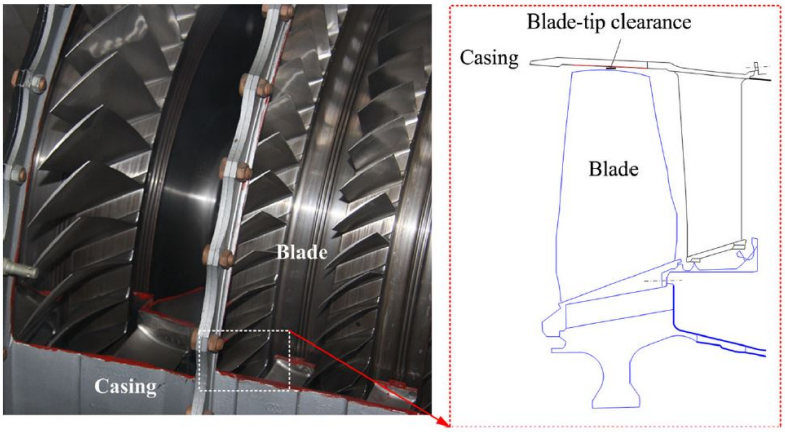


Figure 1.1: A schematic diagram of blade–casing clearance [2]

Some causes that can lead to this type of malfunction are: wear, overheating, shaft thermal deformation, vibration instability with machine destruction (in the most severe cases of continuous rub), etc. In this framework can occur two main types of malfunctioning: [3]

The first one is characterized by a contact between the tip of the blades and the casing that present a relatively low force levels. The consequence of this condition can be different, based to the rotor's speed of rotation, the load, the system's

geometry and physical parameters. These states may induce unstable phenomena whose consequences can be disastrous.

The second configuration, associated with high unbalance levels, is generally consecutive to an accidental blade loss. This problem is not of interest for this project, because it is not a malfunction that generates a rub of low intensity that gradually increase, allowing the code to detect the problem in time to plan a preventive maintenance of the system.

The different states associated with rubbing conditions can be classified as damped (the contacts disappear through time), divergent (the amplitude of the vibrations increases constantly) and self-maintained (the contacts do not disappear through time). [3]

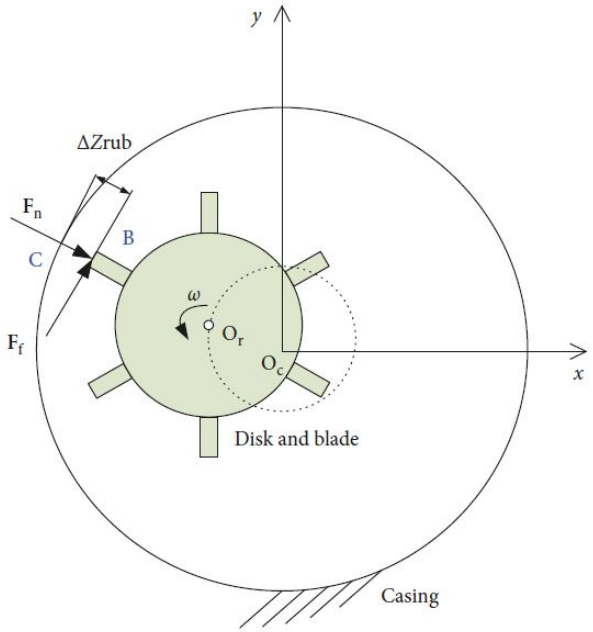
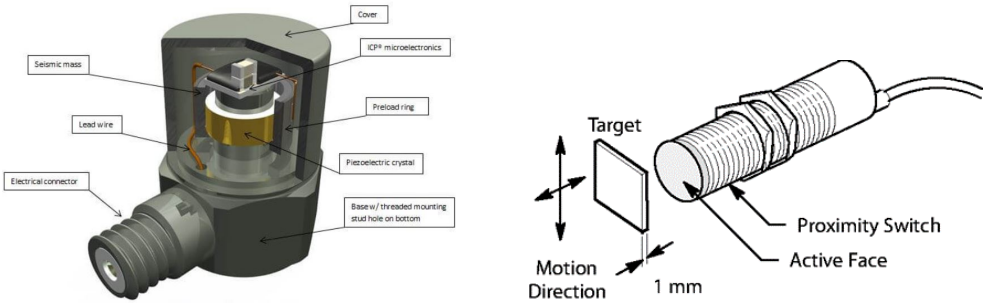


Figure 1.2: Model of a rotor rubbing against the casing [15]

The most common form of rub is the partial rub, which can also be called single-point or fixed-point rub. It is a divergent type of rub, and it can also grow over an extended period of time. Typical consequences of this problem are increased vibration, noise, energy losses and overheating and wear of machine components. Afterwards, due to a positive feedback loop, the intensity of the rub may increase until the gas turbine is compromised.

For all these reasons, to guarantee the correct behavior of the rotating machine is necessary a continuous control of working parameters. The best source of information on the operating state of a gas turbine are the vibrations induced on the shaft. In order to have a direct measurement of this physical quantity it is possible to use two proximity type sensors, placed on the rotor supports and positioned on

the same plane with a angular separation of 90°, so as to have a representation of the 2D orbit of the axis. The proximity sensor, that is able to detect the presence of nearby objects without any physical contact, measures the relative displacement of the reciprocal surfaces. This solution can be applied in standard heavy gas turbines, where hydrodynamic bearings are adopted as supports, allowing to measure the relative displacement between surfaces. On the other hand, in the case of aeroderivative gas turbines, the use of simpler supports, such as ball bearings, does not allow this type of detection to take place since the two elements (rotor and supports) have a rigid connection. This limitation forces the user to look for another type of technological solution of monitoring system condition, which can be carried out by applying accelerometers to the outer casing. This solution leads an indirect measurement of the vibrations of the rotor and the signal of the acceleration presents noise as a result of the turbulent gas flow. This is an important complication to detect the rub when it is still of low intensity and localized in a section of the rotation.



(a) Piezoelectric accelerometer sensor [4] (b) Proximity Sensor [44]

Figure 1.3: Proximity and Accelerometer Sensors Image

1.3 Objectives

1.3.1 Main Objectives

The general objective of the master thesis project is to create a tool to detect the present of a single-point rub into a gas turbine even when it is low intensity rub with high background withe noise. The program will be a Machine Learning code based on the deep neural network technology. The ML software code must be able to run on a real machine even if the program will be trained with data from the finite element numerical model that simulates the behavior of the turbine in question. At application level, the code can be defined ad a preventive maintenance software for aeroderivate gas turbine, which allows the company to save money and improve the safety of the plant.

1.3.2 Specific Objectives

Analyzing the specific objectives of this thesis project in detail we can identify the following tasks:

- Compare different signal processing methods and record which one and which combination gives the best prediction results.
- Try different structures of the neural network and analyze which returns the best classification.
- Analyze the best models and see when starts to detect the defect based on the intensity of the vibrations induced by the rub phenomena.
- Try the best models on experimental data to understand witch network better generalize the information of the finite element numerical model

1.4 Industrial importance

To understand the Industrial relevance of this project it is important to analyze the application areas of this technology. The main advantages that this asset can bring are connected with the reduction of the maintenance time of the machine, creating economic benefits to the various sectors.
Starting with an analysis of the energy sector, where gas turbines have countless applications. Especially now, solutions with a combined cycle configuration or even in cogeneration configurations are very important in the industry, due to their much higher efficiency compared to a simple gas cycle. A very important feature that makes them versatile in this economic sector is the possibility, within simple cycles (gas turbine only), to be activated and shut down in a few minutes, allowing to respond to peaks in energy demand. Especially in relation to the latter application, the possibility of increasing the availability of the facility ensures a better capacity to respond to unforeseen demand, guaranteeing a better economic return.
In addition, as we can see in the figure below, the whole energy sector will see significant growth in demand in the coming years, particularly in non-OECD countries, making the ability to have reliable power generation an essential element. [**?**]

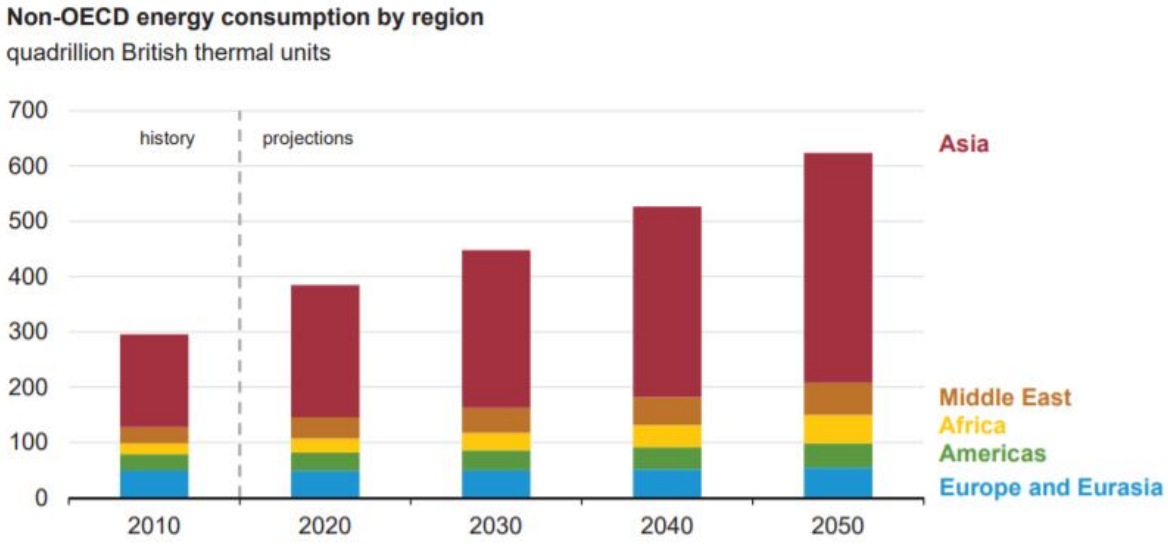


Figure 1.4: Non-OECD energy consumption prevision 2050 [14]

A large single-cycle gas turbine typically produces 100 to 400 megawatts of electric power and has 35-40% thermodynamic efficiency. [16].
The type of gas turbines that will be considered in this work are *Aeroderivative Gas Turbines*: these turbines for power generation are engines for aeronautical applications converted for power generation. This technology thus guarantees an excellent

power-to-weight ratio, however, requires a readjustment of the rotating machine status monitoring technology, which must be adapted for this new technological solution.
The other sector where gas turbines play a key role is the air transport sector, where safety is a pivotal element. In this application, a technology that improves the speed of diagnosis of a failure is a key asset. Currently the project is focused on turbines for power generation, later it could be expanded to include this other application.

1.5 Structure of the project

The picture below shows the flow chart of the thesis project.

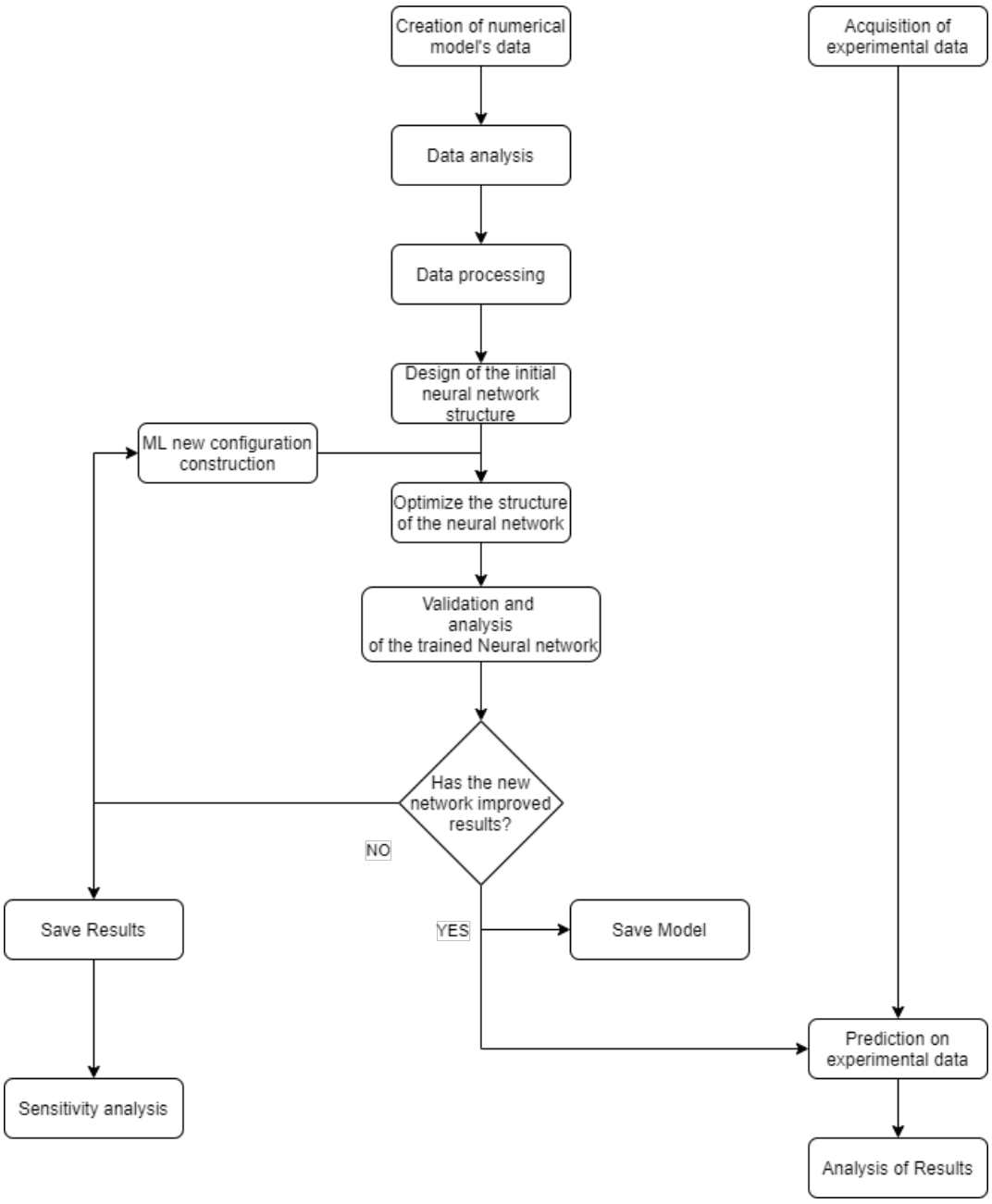


Figure 1.5: Flow chart of the project

It is possible to divide into five phases in which this project has been carried out.

1. The first one is composed of the acquisition of experimental data from the model of a rotating machine

2. Simultaneously with the first phase, in the second phase the software architecture of the machine learning has been built on the python scientific environment Spyder program.

3. In the third phase the synthetic data of the numerical model were analyzed and processed.

4. Subsequently the network was trained and probed **only** with the synthetic data of the numerical model and then, comparing the results of various neural network structures, those that gave the best results were saved.

5. In the last phase of the thesis project, the best trained networks were analyzed, evaluating prediction and recall according to the threshold and testing the final deep network prediction on the data generated with the real rotating machine prototype.

Chapter 2

THEORETICAL FRAMEWORK

2.1 Gas turbines

2.1.1 Main components

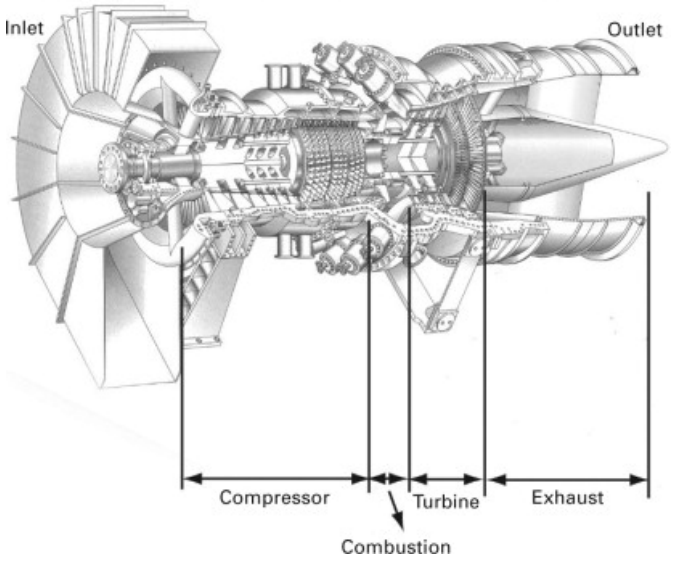


Figure 2.1: Gas Turbines Components [6]

A gas turbine, also called a combustion turbine, is a continuous and internal combustion engine which can produce a thrust, in the case of an aircraft engine, or a torque at the shaft for electricity generation.
The main elements common to all gas turbine engines are:
- An upstream rotating gas compressor
- A combustor
- A downstream turbine on the same shaft of the compressor

The **compressor** is the engine component that increases the inlet fluid pressure and reduces its volume. The working fluid (air in gas turbines) is sucked into the system by the effect of the rotating blades of the compressor and compressed to very high pressure up to 40 atmospheres. In gas turbines it can be of axial or centrifugal type. The energy required for its operation is supplied by a turbine to which it is connected by a crankshaft. Part of the air flow of the compressor is tapped for the cooling of the engine (typically the turbine) or, in aeronautical applications, by the air conditioning and pressurization system. The compressor, unlike the turbine, operating in an adverse pressure gradient (the pressure increases as the flow progresses). For this reason it has a much higher number of stages than the turbine even though it operates on practically the same pressure jump, because each stage can achieve a lower pressure jump than an expander stage. The vane profiles are low curvature, to avoid the detachment of the fluid vein and each stage allows a modest compression ratio. The first stages of the compressor have warped vanes, with a variable shrinkage angle from the root to the end, to adjust the fluid inlet direction to the different peripheral speed.

In the **combustion chamber** the chemical energy possessed by the fuel (usually kerosene or methane) is released through its combustion process with the oxygen present in the compressed, raising the temperature (and therefore the enthalpy) of the flue gases. The flame temperature to achieve a stoichiometric relation for the combustion is about 2200 °C, well above the temperature bearable by the materials of the turbine. For this reason only a part (less than half) of the air participates to the combustion, while the remaining part is used to decrease the temperature of the flow that invests the turbine by diluting the flue hot gases.

Speaking of the **turbine**, it is important not to confuse the expander alone (which we will call turboexpander) with the whole system. Due to the extreme working conditions, the turboexpander is the critical element of the entire turbine. The temperature of the gases coming from the combustion chamber can reach, in the modern engines, even at 1600 °C. [21] The rapid rotation of the turbine, then, induces additional mechanical stress to the vanes which, coupled with thermal stress, can cause the creep problem.

In order to allow the expander to operate at such extreme temperatures, it is necessary to cool the blading. There are two methods for cooling: the internal cooling or the film cooling. In the first, the air tapped from the compressor slid into the hollow vanes, thus cooling from the inside. In film cooling, the vane has small holes, suitably oriented, by which the air tapped from a compression stage with a higher pressure than the turbine stage that will be cooled is insert. The air flow passes into the hollow vane cooling it from the inside, and then it comes out and follows a direction that allows it to be adherent to the surface of the blade and to create a film layer that acts as an insulator between the incandescent gases and the metal surface of the blade.

2.1.2 Principle of operation

The ideal cycle (Ideal thermodynamic transformations + ideal fluid) behind the gas turbine system is the Bayton cycle. The working fluid is air. The cycle is composed by:

- (1 -> 2) - Isoentropic Compression
- (2 -> 3) - Constant $p = p_2$ heat assumption
- (3 -> 4) - Isoentropic Expansion
- (4 -> 1) - Constant $p = p_1$ heat release

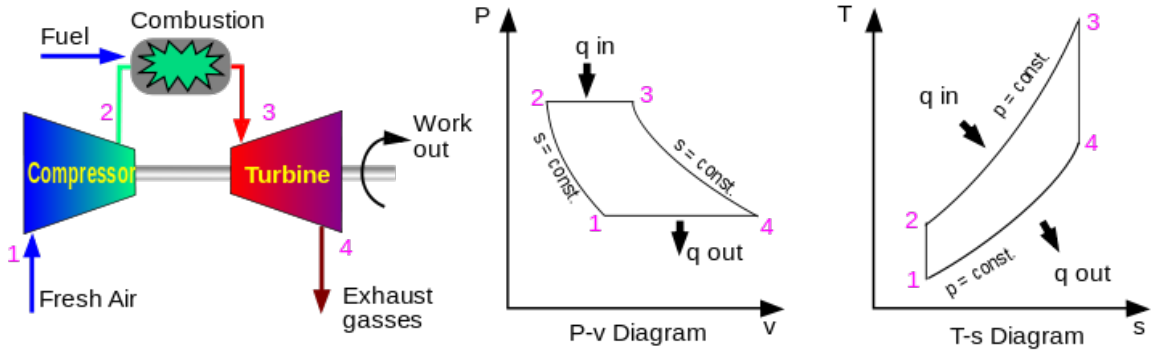


Figure 2.2: Brayton Ideal cycle [7]

The energy that enables the system to operate is supplied by the combustion of the fuel. This, injected into the air flow in the combustor, igniting it allows to achieve a high temperature pressurized flow. Then the glowing flow enters into the turbine, producing a shaft work output which is also used to supply the compressor.

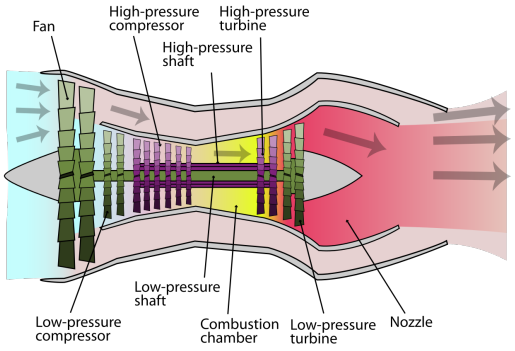
A real gas turbine uses an open cycle, where the fourth flow cooling transformation is not present. Exhaust gases can be recovered and reused, in the case of aeronautical engines, for the generation of thrust, thanks to the law of conservation of momentum and, in the case of electricity generation plants, to power a combined steam cycle (with a heat recovery steam generator) or another downstream gas turbine. In the design of a gas turbine it is essential to divide the energy output between the torque at the shaft and the thrust generated at the output, according to the industrial application of the engine. By analyzing the different applications, in the case of industrial generation or helicopter rotor powering, the turbine design must try to keep the outlet pressure close to the air inlet conditions and sufficient

to overcome the pressure losses in the exhaust ducting, so as to reduce the losses due to exhaust gases. In the case of turbojet engines, the energy extracted from the shaft is sufficient to power the compressor and other components, and the remaining energy contained in the high-pressure gases is all used to accelerate the flow in the nozzle to provide a get to propel an aircraft.

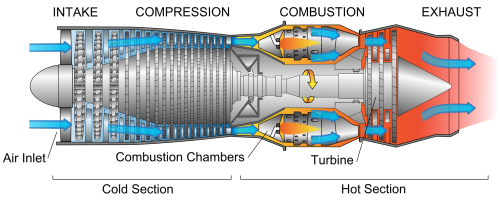
2.1.3 Types

Turbojet and Turbofan

An airbreathing jet engine is a gas turbine optimized for thrust generation by exhaust gas. These can be called turbojets if they produce the thrust directly from the exhaust gases, or they can be called turbofans if they also use in addition a ducted fan to increase the thrust. This fan is positioned in front of the engine, and it accelerates the air before entering in the compressor. Part of the accelerated air flow bypasses the core gas turbine engine, not entering in the compressor and directly providing part of the thrust. Turbofan engines are currently highly used on medium or long-range airliners, because they are more efficient than turbo jets for subsonic speed travel applications. This solution is not optimal for travel at a supersonic speed because their larger frontal area causes a significantly increase in drug resistance overcoming the sound barrier. Turbofan turbines can be divided into two category, low-bypass or high-bypass depending on the amount of flow bypassing the core engine. Low-bypass turbofans have a bypass ratio around 2:1 or less. Turbojet engines are widely used in the military, where efficiency does not play a central role in the design, and other performance factors play a more important role.



(a) Turbofan



(b) Turbojet

Figure 2.3: Jet turbines [8]

The element that converts the energy contained in the exhaust gas into airplane thrust is the nozzle at the end of the turbine. Propelling nozzles are able to turn

the internal and pressure energy into kinetic energy. The total pressure and temperature do not change along the axis of the nozzle but their static values drop as the gas speeds up. Nozzles can be designed for subsonic, transonic or supersonic speeds, depending on inlet and outlet nozzle thermodynamics conditions and on the turbine power output. In a convergent-divergent nozzle ("de Laval nozzle") the converging section of the nozzle can accelerate the subsonic flow, while a diverging part is necessary to accelerate the supersonic part of the flow. Turbofan engines may have an additional and separate propelling nozzle which further accelerates the bypass air.

Industrial gas turbines for power generation

Gas turbines for industrial plants can have very variable dimensions and powers. In the energy sector, the power turbines of larger industrial machines operate at 3000 or 3600 rpm, depending on the electrical frequency of the grid, thus avoiding the necessity of a gearbox. When gas turbines are inserted into combined cycle plants, where exhaust gas' energy is used in CHP (Combined Heat and Power) configurations, economic (non-thermodynamic) efficiencies of more than 60% can be achieved. For example the 605 MW General Electric 9HA achieves a 62.22% efficiency rate with temperatures as high as 1,540°C.[22]. Simple gas turbines, on the other hand, have a lower efficiency (about 40%), but they have a faster start-up period.

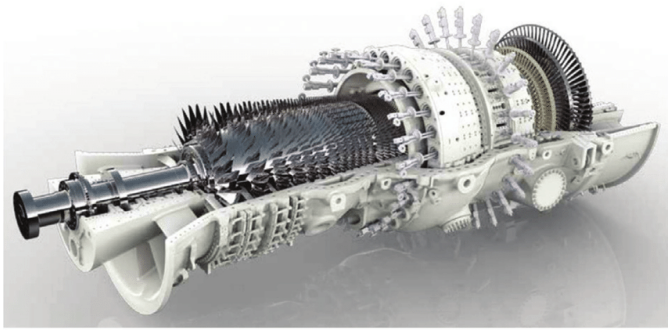


Figure 2.4: Heavy duty gas turbine GT26 [20]

Aeroderivative gas turbines

Figure 2.5: Siemens SGT-A65 Aeroderivative gas turbines [19]

Aeroderivative gas turbines GTs are turbines used for power electricity generation. They are inspired by advanced technology and materials of aircraft engine. They are currently a very popular choice in new gas turbine plants due to their reliability, efficiency and flexibility. The main advantages over heavy industrial GT turbines are lighter weight, faster response speed, better power-to-weight ratio and superior thermodynamic efficiency.
They can have a secondary turbine (known as the Power Turbine or PT) which is not directly connected to the gas turbine shaft but is, also, induced to rotate by the expanding exhaust gases. The most important construction difference compared to industrial heavy frame turbines is the presence of multiple independent shafts to run at optimal speed with PT matched to generator speed, instead of a single shaft fixed to generator speed with multiple variable compressor vanes to control airflow. [24]
Another aspect that differentiates these two types of gas turbines are the types of bearings used to support the shaft: while the heavy gas turbines use hydrodynamic bearings, aeroderivative gas turbines have the more classic ball bearings. The consequence of this different construction solution is the impossibility in the aeroderivatives to use proximity-type sensors to monitor the precession orbit of the shaft, which is a crucial information to know the health state of the system.
The latter reaches efficiency values close to 45%, compared to 35 % of a Heavy Gas Turbines. They are a excellent choice for small plants (up to 100 MW) and they can work with combinations of natural gas and liquid fuel operation, ensuring excellent fuel flexibility. [16]. They can also be run in a cogeneration configuration: the exhaust gases are used for space or water heating, or drives an absorption

chiller for cooling the inlet air and increase the power output, technology known as "turbine inlet air cooling". Due to their fast start-up, stop and response times (the LM6000 family for instance, is five-minute start capable) this sub-group of turbines are perfect to compensate for peak load applications.
Base load operations for this category of engines are typically characterized by continuous operations with only two shutdowns a year for an inspections [23] Primary concerns for base load operations are maintaining and enhancing turbine performance, ensuring operational reliability and minimizing the associated costs of these efforts. When aeroderivatives engines are applied to generate electricity to cover the grid energy demand peak, this type of operation task can require more frequent maintenance than base load operations. This is due to multiple starts and shutdowns in a short timeframe. MTU Maintenance for instance has observed that parts of turbines used for peak operation are likely to see more thermal distress and material fatigue (thermal cycling). [23]

2.1.4 Industrial applications of Aeroderivatives Gas Turbines

The global aeroderivative GT market is expected to grow at an annual growth rate of over 4% between 2016 and 2020 [17]. Asia, in particular, deploys many of these machines in power trains for Liquefied Natural Gas (LNG) plants. In the U.S., aeroderivatives are mainly being used in peaker operations, or to compensate for fluctuations in the grid caused by renewables or extreme weather conditions, which do not guarantee a constant energy output. [16]. Thanks to their rapid start-up period, plants using this type of turbine can be combined with other non-constant energy sources (e.g. wind power) to ensure constant power output to the electricity grid by compensating the other source when the latter should have a reduction in generation. [18]. They are also used in the marine industry to reduce weight.

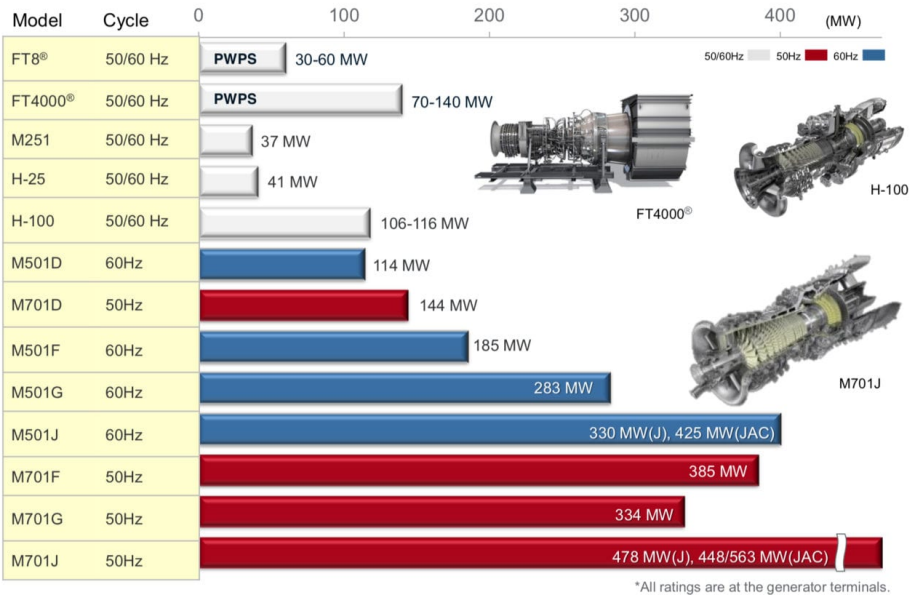


Figure 2.6: Mitsubishi Hitachi Power Systems' aeroderivative gas turbine lineup. Courtesy: MHPS [9]

2.2 Vibration fundamentals

Vibration is a mechanical phenomenon whereby oscillations occur about an equilibrium point. An oscillation is the variation over time of a physical quantity such as, for exampleF, the position of a pendulum at rest that is struck. The oscillations can be periodic, assuming values that repeat exactly at regular intervals, or randomly, without any signal periodicity. In mechanical systems, vibrations involve a periodic conversion of potential energy into kinetic energy and the opposite. The vibration of a rotor system involves the storage and release of energy in the deformation of various spring-like elements. The potential energy can be temporarily stored in shaft deflection, bearing deflection, stator and turbine structure deformation. A shaft in a rotational machine describes a circular or elliptical orbit when subjected to a single vibration frequency. A single vibration can be described by two sizes: *Frequency* and *Amplitude*.

The unit of measurement of the frequency for periodic oscillations is the Hertz [Hz], which corresponds to how many times the same configuration occurs again in a second.

$$f(Hz) = \frac{1}{T}(\frac{cycles}{s})$$

In rotating machine applications, the frequency is often expressed in *cycles per minute*, or *cpm*, so that it can be directly compared to the machine's rotation speed, measured in *revolutions per minute*, or *rpm*.

$$f(cdm) = \frac{60}{T}(\frac{cycles}{min})$$

The frequency can also be expressed in radians/seconds:

$$\omega(rad/s) = \frac{2\pi}{T}(\frac{rad}{s})$$

The ω frequency is called the circular frequency. This frequency description is used to represent the vibration frequency of the rotor system.

The amplitude expresses the magnitude of the vibration, and it can be defined in many ways: for a displacement signal is used the double amplitude, or *peak-to-peak*, and it consists in the measurement of the difference between the maximum and minimum amplitude of the signal voltage.

Mechanical systems are subjected to vibrations due to internal and external forces acting on it. These involve the periodic motion of all the parts that compose it (rotor, casing, piping, foundation system, etc.). Normally the magnitude of the vibrations is so small that it is necessary to use instrumentations to detect them. To give an example, a vibration of 130 μm (about the diameter of a human hair) is not acceptable in a gas turbine of the same length of a house [37]. Vibrations generate fatigue loads on the components of a mechanical system. If the intensity

is high enough, the oscillations can cause undesirable contacts between parts of the rotating machine, resulting in wear and damage.

All system components (rotor, housing, supports, etc.) are subject to vibration and they can move in many different directions. The radial (or lateral) vibrations occur in the XY plane perpendicular to the shaft axis. The axial vibrations propagate in a direction parallel to the rotor axis (Z axis). The various components of the system may periodically change orientation, resulting in angular vibrations, which have a radial component. Torsional vibrations may also be present, but they do not have a lateral component.

Very important for this thesis work are vibrations of the casing: these are generated by the vibrations of the rotor transmitted by the bearings, the vibrations of the piping system and the basic structure transmitted by the foundations, also generated by another machine nearby. The intensity of the stator vibrations depends on the inertia of the system, the stiffness and damping of the bearings and the basic structure. Since it is not possible to have a completely stationary housing, it is not possible to obtain an absolute measurement of the rotor vibration, and we will refer to the *relative shaft vibration*, as illustrated in the figure 2.7.

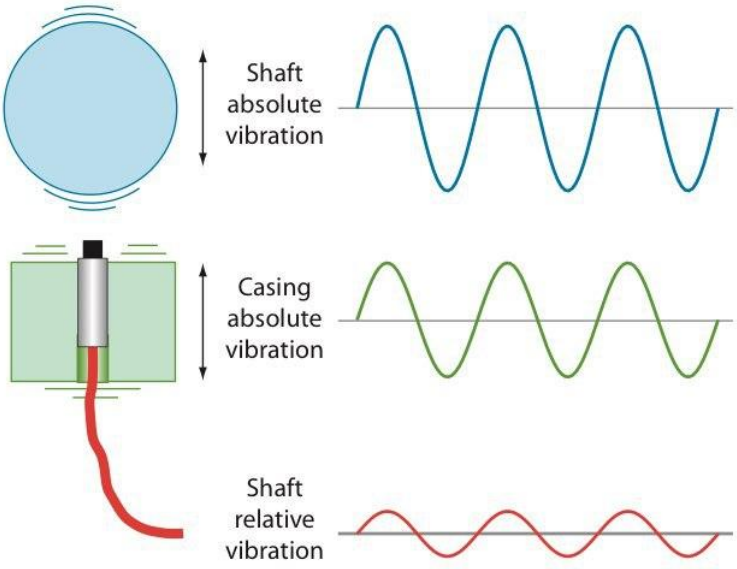


Figure 2.7: Relative shaft vibration [37]

2.2.1 Rotation and Precession

It is important to distinguish the rotation movement from the precession movement of the turbine shaft.

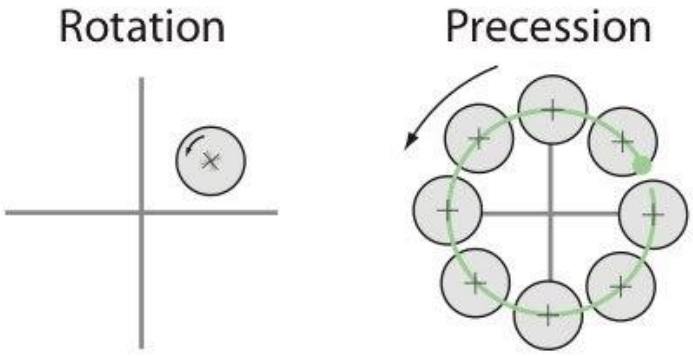


Figure 2.8: Rotation and Precession [37]

Rotation is an angular movement of the shaft around its axis. The rotation can theoretically take place without any lateral movement of the element, in a perfectly balanced system without external forces with ideal components.

Precession is a lateral movement of the geometric center of the rotor in the XY plane. When we talk about the vibration of the rotor, we refer to its precession movement. The orbit is the trajectory that the geometric center of the shaft runs through, and it can have a circular shape or it can take on different complex shapes that contain several frequencies of vibration.

Precession and rotation are two independent phenomena that can also occur individually. They usually occur simultaneously.

When precession and rotation have the same direction of rotation, the movement is called *forward precession.* Instead, when they move in the opposite direction, there is a *reverse precession.* [37]

2.2.2 Types of vibrations

Free Vibration

When an underdamped mechanical system is displaced from its equilibrium condition and subsequently released, it begins to oscillate, as shown in the picture 2.9, at a frequency called the *damped natural frequency* ω_d, defined as:

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}$$

With ω_n *natural frequency* , defined as:

$$\omega_n = \sqrt{\frac{k}{m}}$$

where ζ is the *damping ratio*, and it describes the ability of a component to dampen vibrations. The oscillation of the system continues until the system dissipates all the energy or it is excited again. The energy that generates the free oscillation, contained in the initial system deformation from the equilibrium condition, can be supplied either with an impulse or with a step function.

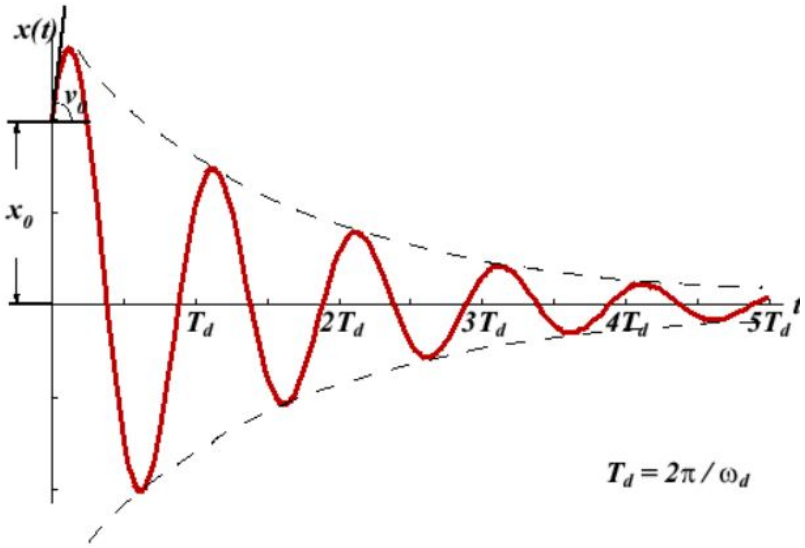


Figure 2.9: SDOF UnderDamped Response [10]

Forced Vibration

The tendency of one object to force another adjoining or interconnected object to vibrate is referred to as a forced vibration. Forced vibrations are generated by periodic forces acting through a dynamic stiffness of the rotor system. Dynamic stiffness is a combination of inertia, damping and static stiffness of an element. The resulting vibration is the ratio of exciting force to dynamic stiffness. In contrast to a free vibration, the oscillation frequency of a forced vibration depends solely on the frequency of the input force of the system. In linear systems, the output frequency is the same as the input one. In non-linear systems, however, the output contains fundamental force frequency and also higher-order harmonics. Rotor system can exhibit both linear and non-linear systems [37]. Another difference between the two type of vibrations is that, at a given constant frequency and amplitude of the excitation force, the oscillation amplitude of the system will not reduce over time. The oscillation amplitude of the system varies according to the frequency of the forcing force, even if the amplitude of the force is kept constant.

One of the most common causes of excitatory forcing is unbalance. Due to the non-homogeneous distribution in the radial direction at each axial section of

mass in the rotating element, the displacement of the center of gravity with respect to the geometric center generates a synchronous centrifugal force, or 1X, respect the frequency of rotation of the shaft. The equation that describes the unbalance-related inertia force is as follows:

$$F_U(t)P = mr\Omega^2 e^{j(\Omega t+\delta)} \tag{2.1}$$

where m is the mass unbalanve, r the displacement of the displacement of the rotor mass centerline respect the axis of rotation, δ is the angular location and Ω is the rotation speed of the system. As we can observe, the force magnitude and direction are functions of the axis rotation speed, generating a synchronous perturbation force.

A vibrational torsional forcer can be generated by a change in gear geometry, periodic contact between rotor and stator, an irregularity in torque generated by the electric motor, etc. These torsional vibrations can generate radial vibrations due to the shaft deflection away from the spin axis, thus generating an increase in the system's moment of inertia. [37]

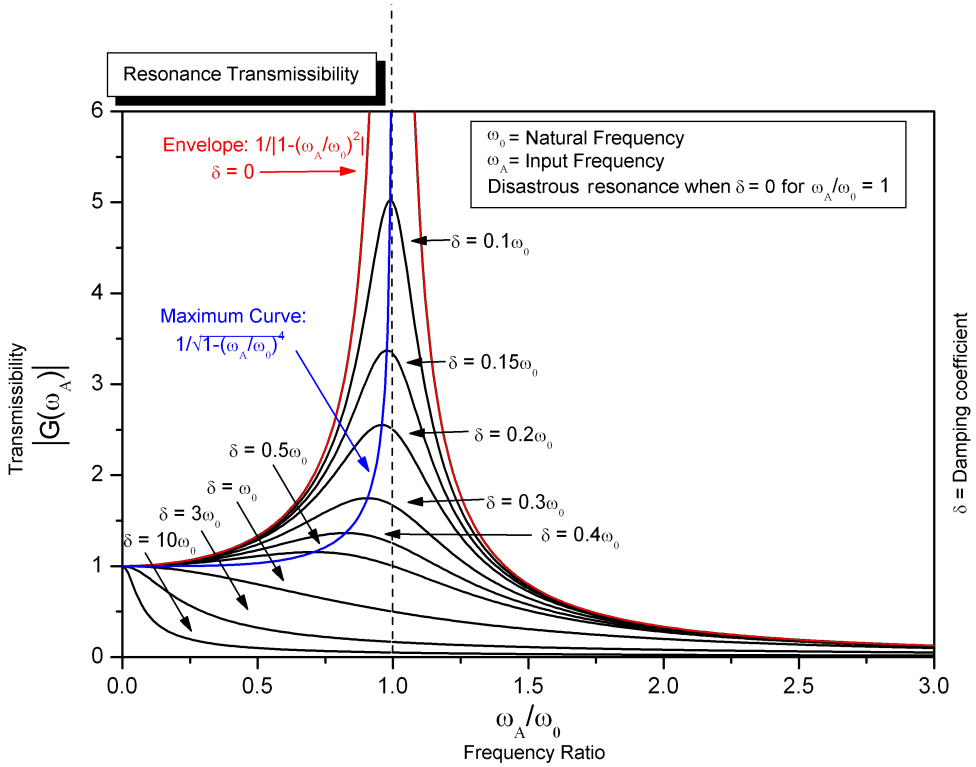


Figure 2.10: Resonance Transmissibility [11]

When the excitation force has a frequency similar to the natural frequency of the mechanical system, the amplitude of the vibrations can increase to a critical value.

This phenomenon is known as *resonance*. The resonance amplitude of oscillation, expressed with the Synchronous Amplifier Factor (SAF), depends on the damping capacity of the structure. In a rotating machine, the unbalance generates a 1X frequency excitation, with the same frequency of the rotor rotation. As the rotor speed approaches the natural frequency of the system, the oscillation amplitude increase. The critical rotor speed is the value that generates a frequency excitation equal to ω_d, reaching the balance resonance. When a machine has an operating point at a higher speed than the critical speed, the system will pass through this risky value during startup or start down processes. Resonance is a dangerous phenomenon for rotating machines, because it generates high mechanical stress, stator-rotor contact and seal wear.

Self-excited Vibration

Self-excited vibrations involve the conversion between an energy source not related to vibration, into vibration energy. The excitation forces that are generated have the same frequency as the natural frequency of the system. This vibration phenomenon is similar to resonance, only the energy source is not connected to the vibration of the system. An example of self-induced vibration is the subsynchronous vibration due to the rub. Under particular conditions, the periodic contact between the rotor blades and the internal casting generates an excitatory pulse at the same natural frequency as the rotor system. At the point of contact, the high tangent friction forces convert the kinetic rotational energy into radial vibration energy. With each new contact, additional energy is released, and it compensates for the vibration energy losses. The self-excited vibration generated by the rub phenomenon has a frequency equal to harmonic submultiples of running speed (ex. 1/3X, 1/2X etc). [37]
One type of rub that presents a self-exited vibration is the "full angular rub", which is one of the most dangerous types of rotor/stator contact. The single-point rub is not a self-excited vibration, but if it is not detected in time, it can turn into a "full angular rub".

2.2.3 Rotating Machine Model Fundamental Steps

All models are always a simplified version of the machine that we want to simulate. It never represents all the features of the system, but it is designed instead to describe specific characteristics of it. A numerical model can simulate certain important features of the real rotor system, but its applications are limited by the assumptions made at the design stage to simplify its structure. Simplifications inevitably lead to errors of approximation.
The modeling of a physical system normally follows the following steps:

- State of assumptions and consequent limits of application of the model
- Definition of the cord system to describe the movement of the system
- Definition of the forces acting on the system, according to the variables of the model (displacements, speeds, acceleration and time)
- Development of the system of differential equations that describe the model behavior, which can be obtained either through the free-body diagram, the theory of virtual works or the theory of Bond Graph.
- Solving the system of equations, thus allowing the coordinates of the system to be expressed as a function of time.
- Validation of the physical model against the real model and correction if the approximation error is excessive

The forces acting on the model can be divided into two categories: *Internal* and *External* Forces. *The Internal Forces* are generated by the interaction between different machine's internal parts. Some examples are the support force of the bearing, forces resulting from shaft deflection, hydrodynamic forces generated by the interaction with the fluid inside the machine. *The External Forces* are applied to the system and generate a perturbation on it. Exist two types of perturbation: static or dynamic perturbation. The static perturbation forces have a constant magnitude and direction, and they produce static deflections or changes in rotor position. Instead dynamic perturbations periodically change magnitude and/or direction, and the rotor system react through vibrations.

An example could be the force generated by the contact between stator and rotor, or the static radial load or any other arbitrary force imposed on the system.

The numerical model used is described in detail in chapter section 4.2.

2.3 General information on preventive maintenance

According to EFNMS (European Federation of National Maintenance Societies) maintenance is defined as: "All actions which have the objective of retaining or restoring an item in or to a state in which it can perform its required function. These include the combination of all technical and corresponding administrative, managerial, and supervision actions". Over time the term maintenance refers to multiple wordings that describe various cost-effective practices to keep equipment operational; these activities take place either before or after a failure.
The maintenance of industrial machinery is based on the following guidelines: [25]

1. Maintain the production equipment and plant utility systems equipment as close to brand new condition as possible and have all equipment ready to start up and run with no unplanned shutdowns.

2. Maintain the production equipment and plant utility systems equipment in the best possible operating condition for the purpose of producing quality manufactured goods while the machines are in service.

3. Complete all maintenance work on a regularly scheduled basis without exceeding the “Point of Diminishing Returns on Investment” for the labor, tools and materials required to perform the work.

The basic types of maintenance falling under MRO (maintenance, repair and overhaul) include: [26]

- Preventive maintenance, also known as PM

- Corrective maintenance where equipment is repaired or replaced after wear, malfunction or break down

- Predictive maintenance, which continuously monitors the system evaluating it against historical trends, to predict failure before it occurs

- Reinforcement

Preventive maintenance (or preventative maintenance) is maintenance which, by making continuous measurements, reduces the likelihood of component failure. Measurements are made on the component while it is still working, so you can intervene when the fault has not yet occurred. In terms of the complexity of this maintenance strategy, it falls between reactive (or run-to-failure) maintenance and predictive maintenance.

In turn, it is possible to identify different categories of PM preventive maintenance:

- Planned preventive maintenance (PPM), more commonly referred to as simply planned maintenance (PM) or scheduled maintenance. This maintenance is carried out at predetermined intervals, which can be either temporal or based on a parameter that describes the use of the machine (e.g. kilometers travelled by a car).
- Condition-based maintenance (CBM). It is maintenance when need arises. CBM maintenance is performed after one or more indicators show that equipment is going to fail or that equipment performance is deteriorating.

The program developed in this thesis project is a support tool for the CBM maintenance of aeroderivative gas turbines. The objective is to prevent malfunctioning from leading to system failure by monitoring engine operating conditions and intervening when the machine is not yet compromised.

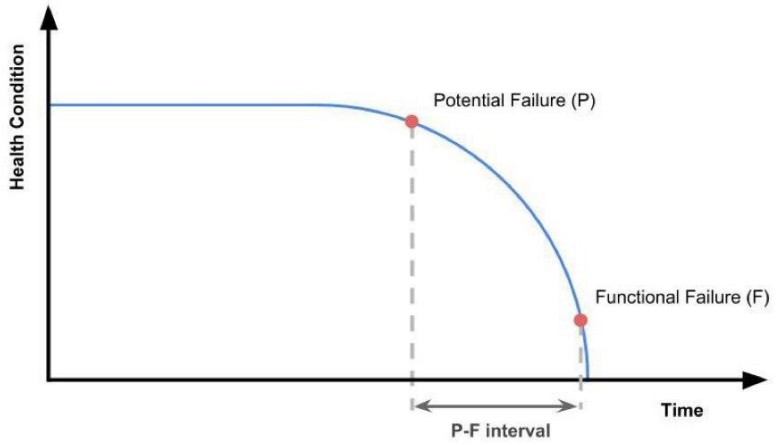


Figure 2.11: The Performance Failure curve [51]

If the gathered data indicates a high probability of reaching a failure, early corrective action can be taken, thus saving lots of maintenance cost and preventing a catastrophic failure. The aim is to optimize operation costs and to minimize downtime times. The idea of CBM is to measure how machine health conditions change in time, so that corrective action is taken before ever reaching a Functional failure event (F). The deterioration of the machine becomes visible in the instrumentation at the Potential Failure (P) 2.11. A predictive maintenance would be able, based on the observation of the machine conditions, to predict when the P point will be reached and how the curve of "Health Condition" will change over time, thus to better optimizing the interventions on the system.

2.3.1 Vibrations applied to maintenance

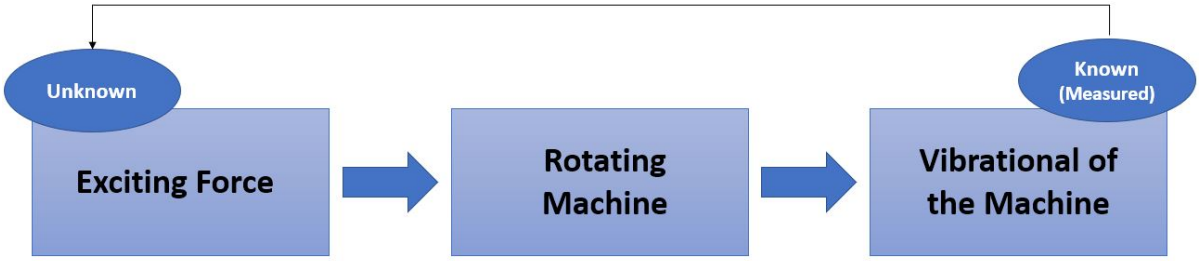


Figure 2.12: Experimental Scheme Rotating Machine

A malfunctioning machine is usually a system that, after a long period of accumulating damage, reaches a breakdown. Most of the machines continue to operate their intended tasks even with an active and developing problem. Proactive machinery management requires to detect the problem as early as possible, in order to schedule the maintenance plan. There are two basic types of measurements to reveal a problem into a mechanic system: [37]

- *Direct Measurements* - A direct control and revelation about the physical propriety of the machine components (E.g.: Stator, rotor, Foundation, etc). Several physical proprieties include vibration and position measurement, rotor speed, and bearing temperature.
- *Indirect Measurement* - Control over the work output parameters of the system. Indirect measurements include processing data, as power, working fluid temperature, pressure, flow, and performance data, such as efficiency. That information has to be correlated with the direct one to diagnose the malfunction.

One of the most important indicators of a malfunction are the vibration of the system. A rotating machine can be considered as a "black box" [2.12] that receives as input the dynamic forces and returns as output vibration of the system, acting as an energy conversion mechanism. To understand how the black box works, we need to understand the relation between the dynamic forces applied on the system and the vibrations that follow them. This would also allow us to deduce, by observing the vibrations of the machine, what forces generated this response, thus understanding the causes of the phenomenon. This will allow us to detect, identify, and correct the potential problem in the rotor system, since we can monitor the vibration of the machine, no directly the dynamic forces than act on it. We can try to guess the contents of the box by shaking it using a technique called perturbation and observing the behavior of the system. The behavior of the black box can also

be simulated by creating a mathematical model of the rotor system.
The vibration is a ratio between Forces and Dynamic Stiffness: a variation of the oscillation of a mechanical system is generated after an alteration of one of those two elements. Dynamic stiffness consists of Direct Dynamic stiffness and Quadrate Dynamic stiffness. Dynamic stiffness is a function of parameters of mass M, spring stiffness K, damping D, lambda λ, and rotor speed Ω. Lambda is a model of fluid circulation between stator and rotor, that reduces the complexity of the viscous fluid behaviour to a single parameter. A change of any one of these parameters changes vibration behavior. The most common exciting vibration force is rotor unbalanced, which can occur after erosion, loss of material, or accumulation of foreign material. Most often, a change in vibration is caused by a change in dynamic stiffness. A variation of mass is unlikely, and a change in lambda produces subtle effects. The more significant vibration variations are caused by lambda and spring stiffness alterations.

The diagnosis of malfunctions is based on the detection of variation in key signal information. In this table is possible to find the most important one to describe the rub phenomena.

Symptomus of the Rub [37]	**Data Plot**
Direct Vibration Amplitude	Orbit, timebase
nX amplitude and phase	Bode, polar, APHT
Frequency	Full spectrum, half spectrum
Position	Average shaft centerline
Orbit and timebase shape	Orbit, timebase

Table 2.1: Data Plot correlated with the relative information of the Rub phenomenon

The effects generated by different malfunctions are very often similar and overlapping in certain aspects. It is, therefore, necessary, to execute a complete analysis, to consider cross-validation with different data sources. Besides, experiments must be performed under different operating conditions: *steady state, transient, slow roll, and stopped.* [37]

- Steady State - It corresponds to a working condition with constant parameters (e.g. rotation speed), and it gives us information about changes in overall vibration levels, frequency content, nX amplitude and phase, position, orbit, and timebase shape during slowing changing, or static, rotor dynamic process conditions.
- *Transient* - It allows us to achieve more information on the dynamic behavior of the rotating machine. Especially It permits to have a correlation of vibrations parameters with the rotation velocity of the rotor. Transient data is

especially important to have information about the variations in the balance resonances speed Ω_{res}.

- *Slow Roll* - This operating condition consists in using the machine at reduced speeds so that the unbalance response can be considered negligible.
- *Stopped* - The system should not vibrate during a non-active condition, so we can measure possible external sources of vibrations. It is also possible to measure, in this state, the absolute coordinates of the system to which all the following dynamic data can be referred.

Operating Conditions	**Description**
Steady state	Constant speed, load can vary
Transient	Startup, shutdown (changing speed)
Slow roll	Low speed, negligible unbalance response
Stopped	Stopped

Table 2.2: Rotating Machine Operating Conditions [37]

The casing vibrations are generated by the transmission of dynamic loads on the bearing supports, which are then transmitted then to the casing through the stiffness and the damping of bearings. The case and the rotor can show different modes of vibration, in-phase, or opposite. The oscillation of the casing is the function of the mass of the structure, the stiffness of the bearings, and of the casing.

A regenerative gas turbine has a low casing mass, compared to the rotor, and relative flexible bearing supports and casing. They also usually have roller bearings that, due to a rigid connection, transmit more vibrations to the casing structure. As a result of the low stiffness, the case could have more vibration modes, and is important to consider them to position the transductors.

2.3.2 Data plot type

To better understand the information on the rotor system contained into the vibration of the shaft is important to apply the best graphic representations correlated to the malfunction that we want to detect.

Timebase Plot

The Timebase plot is the most fundamental graphic representation of the machinery dynamic data. It is the representation plot of a unprocessed single parameter output (e.g. displacement, speed, acceleration) from a single transducer on a very short

time scale (few revolutions). An important use of a timebase plot is to identify change in machine response if sudden event occur or if the machine is rapidly changing speed.

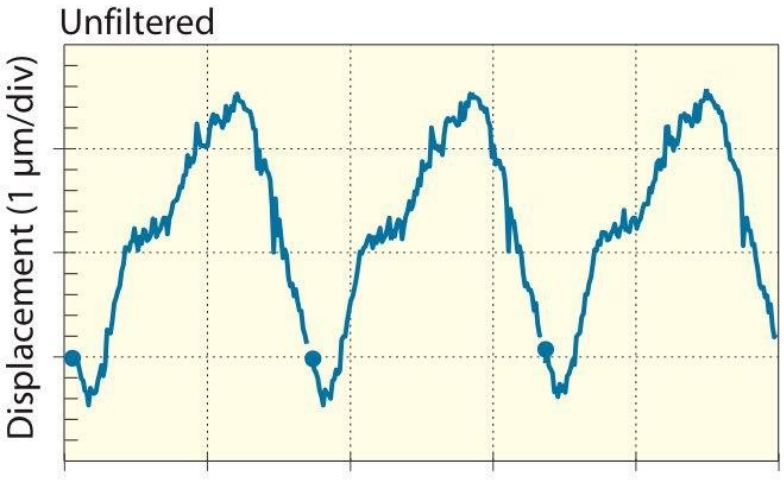


Figure 2.13: Unfiltered timebase plot [37]

Orbit

The orbit is a 2D diagram that allows you to represent the displacement of the center of the axis in a plane. To execute it, it is necessary to use at least 2 transducers, placed at a relative angle of 90°. This diagram can be used to measure the amplitude and phase of the harmonic nX [37]. It is also possible to make several diagrams in different positions to obtain a three-dimensional image of the axis and its deformation [2.14].

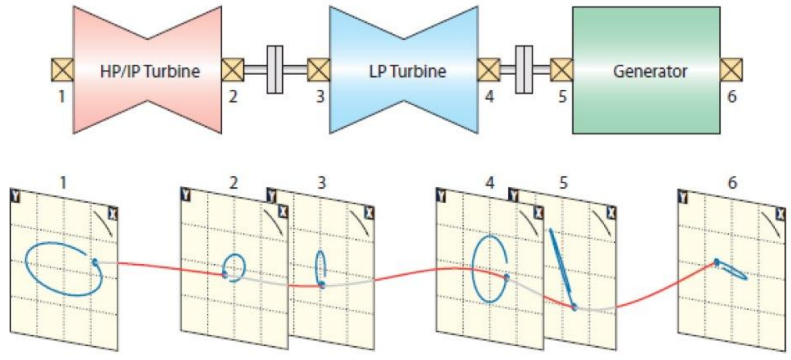


Figure 2.14: Multi-orbit diagram along a steam turbine [37]

Bode and Polar diagrams

The polar and Bode diagrams are mainly used to identify the resonance in the transient periods of the rotating machine. To obtain those graphs it is necessary to filter the information provided by the proximity sensor to its 1X synchronous vibration harmonic component. Generally, the polar diagram is used to identify the position of the point mass to balance the rotor and balance it [37], while the bode diagram is used for the study of resonance. In the Bode diagram, it is possible to observe how the amplitude and phase of the vibration change with the rotation frequency and, with a peak (for amplitude) or a 90° change (for phase), it is possible to notice the presence of a resonance. The polar diagram instead represents the scale of vibration magnitude in the case of the start-up and shutdown of the rotating machine.

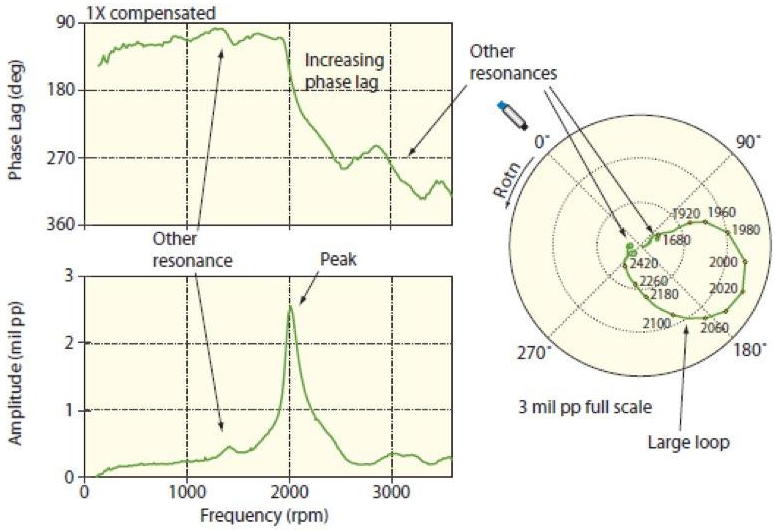


Figure 2.15: Bode and Polar diagrams [37]

Half and Full Spectrum Plots

The machine vibrations can include different frequencies simultaneously. These frequencies are related with the operating conditions of the system (running speed, malfunctions, external vibrations etc.). Half and Full Spectrum plots are important and powerful tool making use to determinate the frequency content of a vibration signal, in order to realize an accurate diagnosis of the mechanical machine. They are applied to identify the frequency components present into the vibration signal and to trend changes on the amplitude of frequency components. The output generated with the application of the Fourier transformation is equivalent to a series of band-pass filters that have been set to integer multiples of the lowest frequency signal f_1. The final signal appears as a series of vertical lines where each one represents a

single frequency and its height is the amplitude of the signal. In this way we lost the phase information of the component signals. For this reason it is not possible to reconstruct the original waveform from the half spectrum plot.

The amplitude scale of the plot can be either linear or logarithmic. With the logarithmic scale can be easily compared amplitudes of very different dimensions, but it is more difficult to quickly discriminate between significant and insignificant vibration components. Instead, by applying the linear scale is easier to find the most significant components.

The frequency scale can be display with hertz (Hz) or Cpm. The hertz scale is useful when comparing machine vibration frequencies to line frequencies, such as an induction motor or steam turbine generator diagnostics. With Cpm is easier to compare frequencies with the running speed of the machine in rpm [37].

Figure 2.16: Half and Full Spectrum Plots [37]

2.3.3 Maintenance sensors for a gas turbine

A *vibration transducer* is a device that converts mechanical motion into an electronic signal. A displacement transducer can detect the displacement or position of an object relative to the sensor location. These sensors are placed as close as possible to the source of vibration information that we want to observe (in the case of a rotating machine it is the rotor). In aeroderivative turbines the accelerometers are positioned on the housing close to the bearings, while in the heavy gas turbines they are positioned in the hydrodynamic bearings (as in the figure below 2.17).

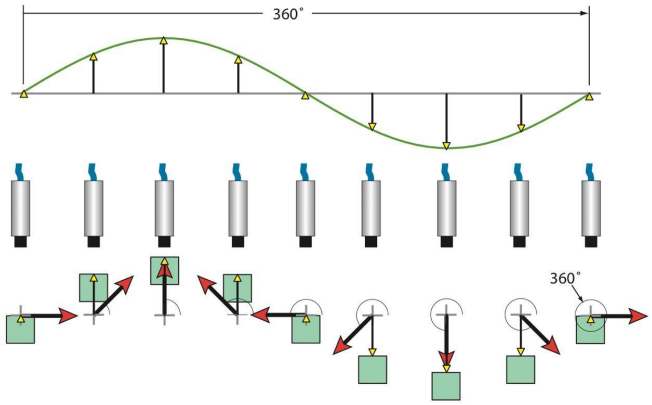


Figure 2.17: The relationship of a displacement vibration signal to the motion of an object [37]

Each sensor is generally able to detect one degree of freedom of the machine. To obtain information of more than one degree of freedom of an object it will be necessary to use more than one sensor.

The types of sensors used in this application are:

Proximity transducer

Also known as Eddy's sensor. The sensor does not need direct contact with the workpiece to perform the detection. It returns the proximity information of the component to the sensor head. A proximity sensor emits an electromagnetic field (Powering a coil at the end of the sensor) or a beam of electromagnetic radiation (infrared, for instance), and looks for changes in the field or return signal [44]. The transducer converts the position of the object with respect to the axis of the sensor into an output voltage, proportional to the gap between the two elements. The variation of the voltage signal intensity represents the relative position of the vibrating object in relation to time. A widespread application of this type of sensors is the machine vibration monitoring to measure displacement between a shaft and its support bearing. This is common in large steam turbines, compressors, and motors that use sleeve-type bearings.

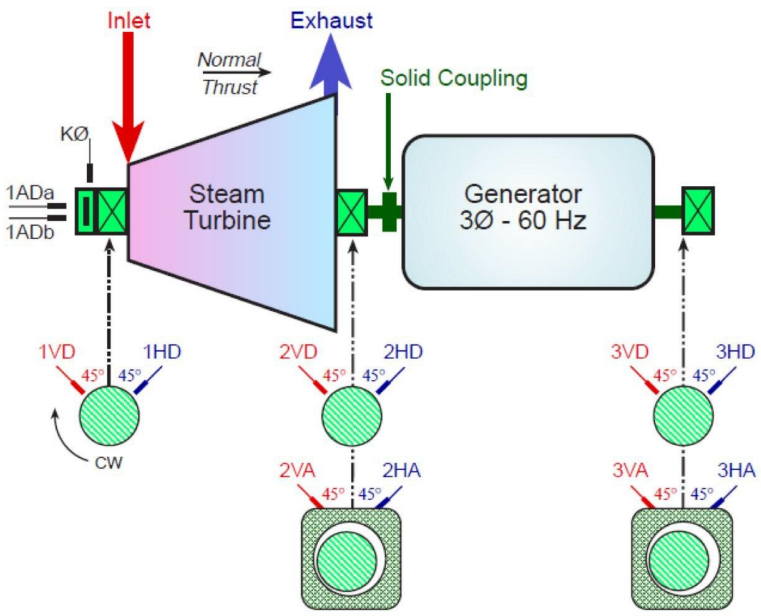


Figure 2.18: Typical configuration of rotating machines with proximity sensors [38]

Speed sensor

Low-frequency sensors that are used to make absolute measurements of the velocity of machine components when the machine is not operating. They are directly inserted into the mechanical structure and they measure the motion of the structure through the contact [45]. The output voltage signal is proportional to the speed of the sensor motion into the magnetic field (Lenz law), generating an output that is a function of the vibration speed of the structure. It is not advised for moving parts or high-frequency vibrations.

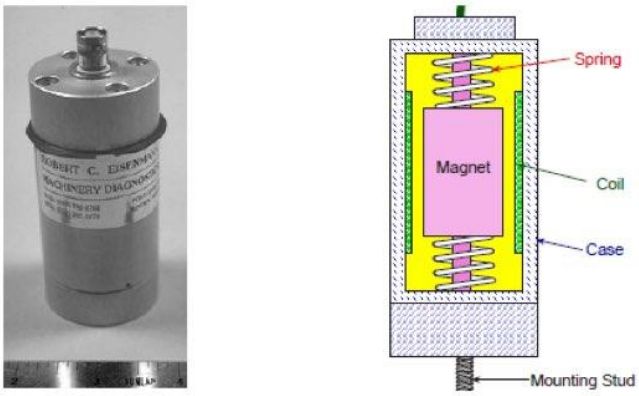


Figure 2.19: Speed sensor [45]

Accelerometer

An accelerometer is a measuring instrument capable of detecting and/or measuring acceleration by calculating the force measured with respect to the mass of the object (force per unit mass). The operation of this sensor can be approximated to a system of 1 degree of freedom. In the design the accelerometer must be designed with an internal natural frequency at least five times higher than the maximum value of the application rank, in order to minimize the inertial effects that can alter its operation and, consequently, the accuracy of the output signal. Knowing that the amplitude of the accelerometer is proportional to the amplitude and frequency of vibration with a quadratic relationship, these sensors are very sensitive to detect vibrations of high frequency and small amplitude [46].

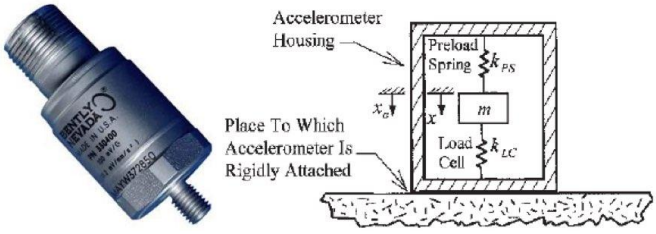


Figure 2.20: Accellerometer [46]

2.4 Different types of Rotating Machine's Malfunctions

The correct behavior of a mechanical system can be altered by the presence of a perturbation forces that alter its operation state. It is important to analyze the different types of external forces to understand their causes, how they impact on the rotating machine and how to prevent them.

2.4.1 Unbalance

The unbalance is the most common cause of vibration inside rotating machines. Unbalance is defined as an uneven distribution of the rotor mass with respect to its center of rotation. The system will respond sinusoidally with the same frequency of rotor rotation. The intensity of the response will be proportional to the residual unbalance of the rotor and the frequency squared.

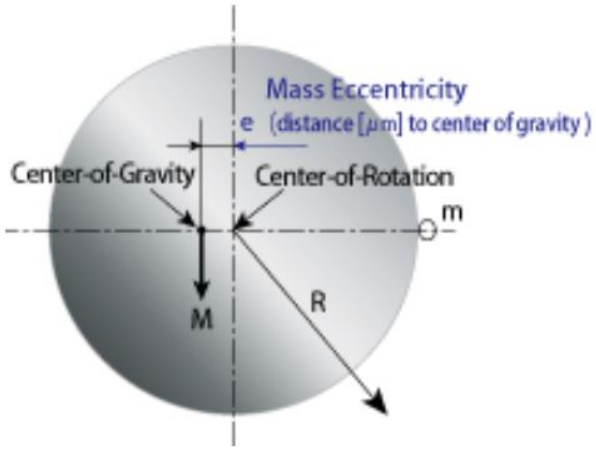


Figure 2.21: Rotor Unbalance [54]

The exciting force generated by this defect has a frequency 1X synchronous to the rotation of the shaft, because, as reported in the chapter on the dynamics of rotating machines, the eccentricity generates a rotating centrifugal inertial force on the center of gravity that rotates with the axis. The unbalanced 1X force acting on the system generates a 1X vibration proportional to Dynamic Stiffness. A 1X vibration generates fatigue loads on the components and, if the intensity of the oscillation is high enough, it generates other secondary malfunctions such as rub or wear in bearings. The unbalance is not the only defect that generates synchronous vibration, so it is important to understand its effects to classify it.

The rotating machine responds to this forcing with non-linear behavior if there is a source of non-linearity in the system. Otherwise the response will be 1X linear. Sources of non-linearity may be:

- The increased rigidity of the bearing fluid-film for an elected eccentricity ratio.
- The onset of secondary malfunction of the rub
- The looseness in the support system

The presence of non-linear behavior generates harmonic vibrations of nX vibration, with n=1,2,3.. etc.

2.4.2 Rub

The rub malfunction consist in the rotor contact with the stationary part of a rotating machine, and the subsequent rubbing on the contact area. It is a serious turbine malfunction that can also lead to a catastrophic machine's failure. The contact between stator and rotor changes the properties of the rotating machine, varying the stiffness of the system, resulting in a variation of the machine's motion. The physical phenomena involved in this event are friction, impacting, variation in system stiffness due to the physical coupling, and also the thermal effects due to energy dissipation. The *friction* depends on the normal force and the condition of the contact surface and generates a tangential force in the opposite direction to the rotation speed. This force is also the main cause of surface wear, which varies its properties. For this reason, this is a transient type phenomenon. Sometimes it can occur a *short-lasting adhesive* rub contacts (with little or no relative movement) transferring rotational energy in rebounding lateral and tangential impacting motion. The rotor/stator contact also varies the *stiffness property* of the system, increasing it, resulting in a non-predicted natural frequency and eigenmode modifications. The physical phenomena of friction and impact are not linear, making the analysis and simulation of this malfunction complex. The problem of rubbing contact is a secondary phenomenon generated, for example, by vibrations due to unbalance, or also due to the displacement of the rotor centerline. Radial rubbing occurs when the rotor displacement exceeds the available clearance [55].

The rub phenomenon can generate rotor bowing, severe wear, local melting or welding of the contact surface, or plastic deformation of the shaft [34].

The rub can occur in the radial direction, axial direction, or both combined, and it also can be a lubricated or not lubricated contact. Most often occurs with an unlubricated one, resulting in a higher friction force and generate sufficient local heating. The axial rub can be caused by a mismatch between the thermal growth rate of the materials that composed the stator and the rotor. During a cold startup, a steam turbine blades expand faster than the casting. [37] The two main common types of the rub are: *Partial Radial Rub* and *Full Angular Rub.*

The first one takes place when the rotor/stator contact happens over a small fraction of the vibration cycle. Partial Rub is the most common type of rub. It can also be divided into other two general categories: *Normal-tight* and *Normal-loose*. The machine rotor normally operates in an unconstrained condition but, when the blades bump against other internal parts of the turbine, it becames contrained, or *tight*. *Normal-tight* is the most common form of partial rub and it increases the rotor system spring stiffness. Instead *Normal-loose* is the manifestation of looseness in the machine. Due to high vibrations, the rotor moves clear of the constrain. The partial rub problem usually generates a temporary, sliding rotor/stator contact. While the rotational speed of the stator is zero, the shaft surface has a high speed. The *dweel time* is the period during which takes place the contact between the casing and rotor. The Partial Rub has a dwell time less than the total vibration period.

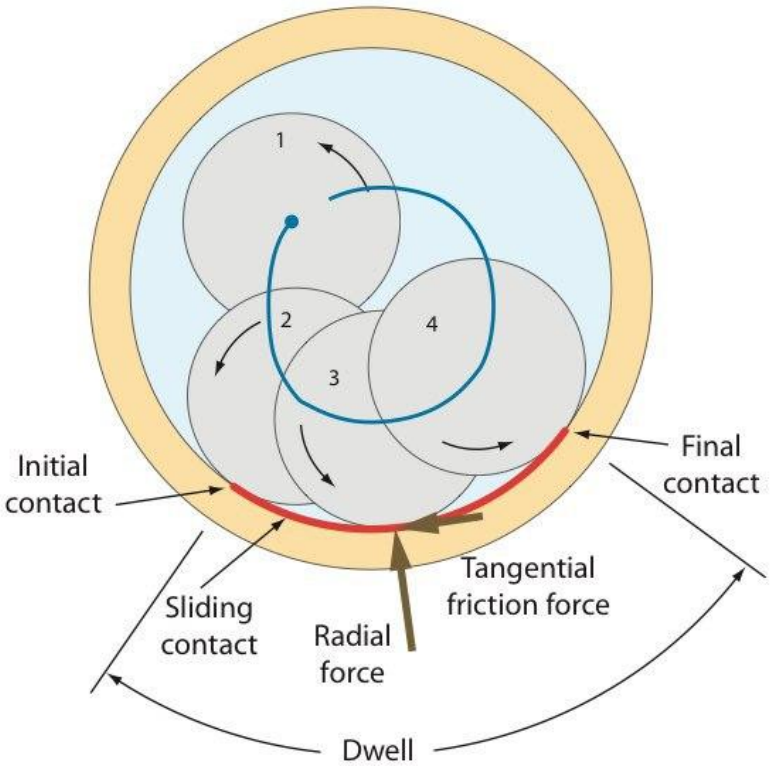


Figure 2.22: Partial Radial Rub [37]

During the period of contact, two forces act on the system: normal and tangent friction force. The second one is proportional to the normal force and to the friction coefficient at the interface. The direction of this force is opposite to the shaft surface velocity. Usually, It is also opposite to the direction of the precession of the center of the rotor. The rotor, during the precession orbit, eventually breaks the contact

and continues in its orbit until the contact recurs. If the contact occurs once per revolution of the rotor, the vibration is 1X. Less often, the rubbing phenomenon one times on each several revolutions, producing subsynchronous vibration.

The *Full Angular Rub* has a dweel time equal to the vibration cycle period. If the precession of the orbit, concentric with the rotor shaft, is large enough, it will exceed the clearance between rotor and stator during all the vibration period, generating the friction. The most common type of Full Angular Rub occurs in Forward direction and, for this reason, the relative velocity of the shaft at the contact point can be quite high. The rotor will lock into a rotor system natural frequency, which has been influenced by the contact stiffness. Rub malfunction introduces non-linear behavior into the rotor system. To prevent this problem, it is important to observe the stationary and transient state. The symptoms of the rub are then explained in detail:

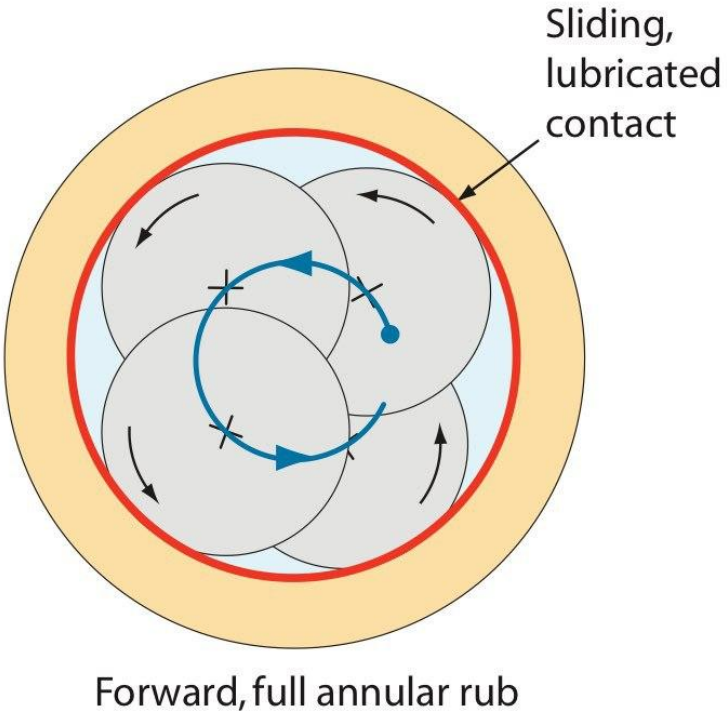


Figure 2.23: Full Radial Rub [37]

Symptomus of the Rub

Changes in 1X Vibration
Abnormal Orbit Shape
Reverse precession components
Harmonics in spectrum
Changes in average shaft centerline position
Wear, damage, loss of efficiency
Thermal bow

Table 2.3: Symptomus of the Rub Malfunction [37]

For a detailed decription of the listed symptoms please refer to chapter 21 of "Fundalmentals of Rotating Machinery Diagnostics" [37]

Chapter 3

MACHINE LEARNING: PROPOSED METHODOLOGY

3.1 Introduction to Machine Learning

Artificial intelligence (AI) refers to all the tools that allow us to simulate human behavior in machines that are programmed to think like humans and mimic our actions. Principally, artificial intelligence programs make it possible to reproduce the classic characteristics of the human mind, such as autonomous learning and problem-solving. Computer science defines AI research as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.[27] Another definition of artificial intelligence is: "a system's ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation." [28]

A famous mathematician and scholar of the 20th century, Alan Turing, wondered if it is possible to define and how a computer intelligent and, consequently, what intelligence is. To solve that problem, he changed its formulation: from whether a machine was intelligent, to whether or not it is possible for machinery to show intelligent behavior. [29] To respond to this dilemma, the mathematician devised a test of a machines' ability, to permit them to exhibit intelligent behavior equivalent to, or indistinguishable from, humans' one. [30] The game he invented is called "imitation game", and is composed of three participants, two humans and a computer. 3.1. One of the two people is the examiner, while the other person and the computer must answer the questions. The examiner cannot see the other two players and he can only communicate with them with written notes. The computer will pass the test if the examiner, based only on the answers, is not able to distinguish between the computer and the other human player.

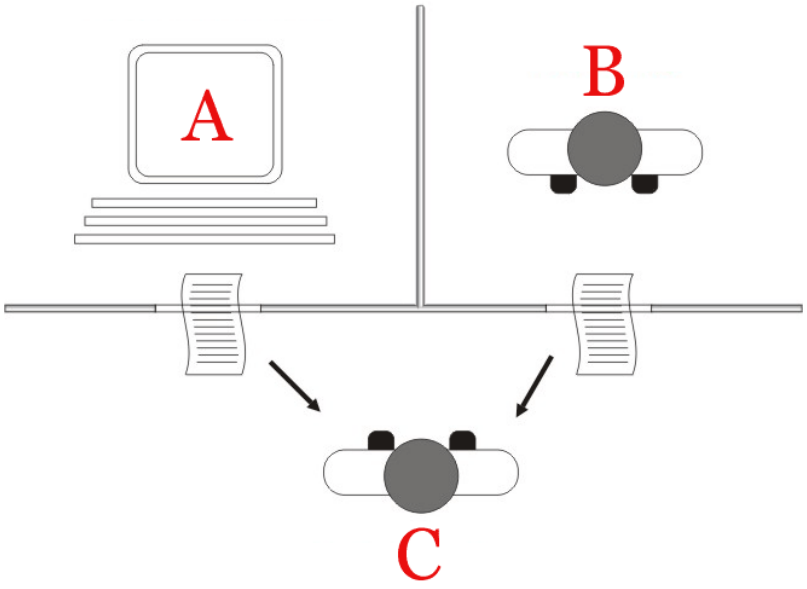


Figure 3.1: Turing test diagram [30]

3.1.1 Machine Learning Structure

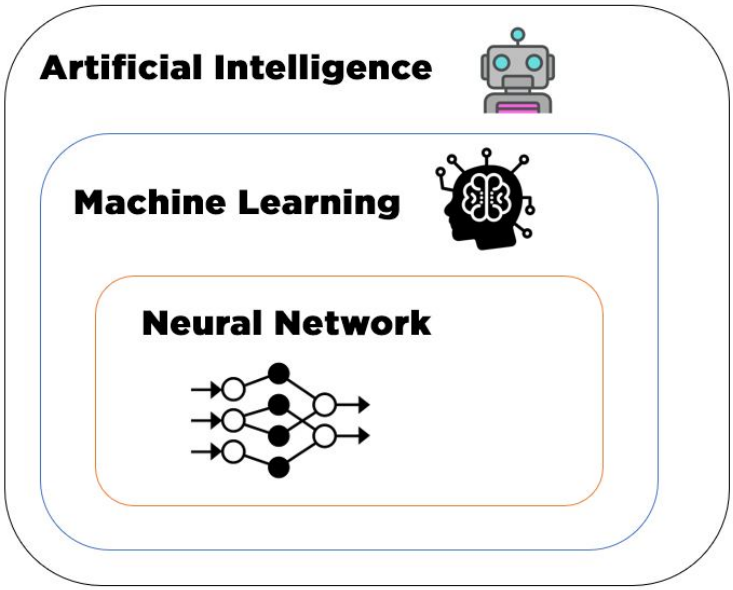


Figure 3.2: AI Structure

Subset of artificial intelligence, Machine Learning is a tool that allows creating algorithms that simulate the human behavior. These tools are based on

mathematical-statistical methods able to learn autonomously from data, updating independently the hyperparameters of the numerical code to improve its performance. In turn, within machine learning, there are different technique solutions that can be adopt depending on the application. Some of these are the decision trees, regression models, classification models, clustering, Support vector machines, etc.

Programming a computer code with Machine Learning changed the programming paradigm. Take for example one of the first machine learning programs developed in the 90s: a spam email filter. [34] Let's imagine that we want to make this program using the classic programming approach.

The first step is to note the patterns present in this type of email. For example, many spam emails contain words like "4U", "free", "credit card" and "amazing". Or other patterns could be the sender's name, the email's body, etc. Then we have to write an algorithm for each of the models listed above to detect. Next, we must test the spam filter and repeat the previous steps until the program is good enough. The program will eventually result as a long list of difficult commands that will have to be constantly updated to adapt the software to new types of spam. For instance, now emails containing "4U" are blocked, but emails with "For U" are not, making the filter ineffective. So, in a classic program, we write the rules with which the program will subsequently process the data.

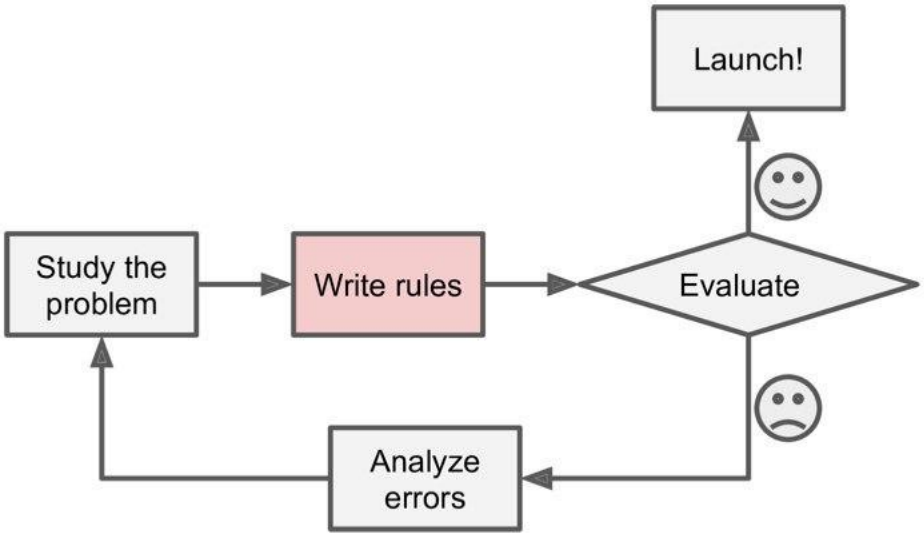


Figure 3.3: The traditional approach [34]

Therefore, we can note that, when the problem to be solved is not trivial, even very complex problems can arise with traditional programming. Let us imagine now to program a software to play Rock/Paper/Scissors recognizing the gesture of the hand through a photo. The code should understand which of the three elements you have chosen by recognizing the shape of the hand. Only each hand is different, with

fingers of different shapes, lengths, different skin colors and also different gestures for the same choice. We can see some sample images in the picture below. [**?**]

Figure 3.4: Rock paper scissors Photos [32]

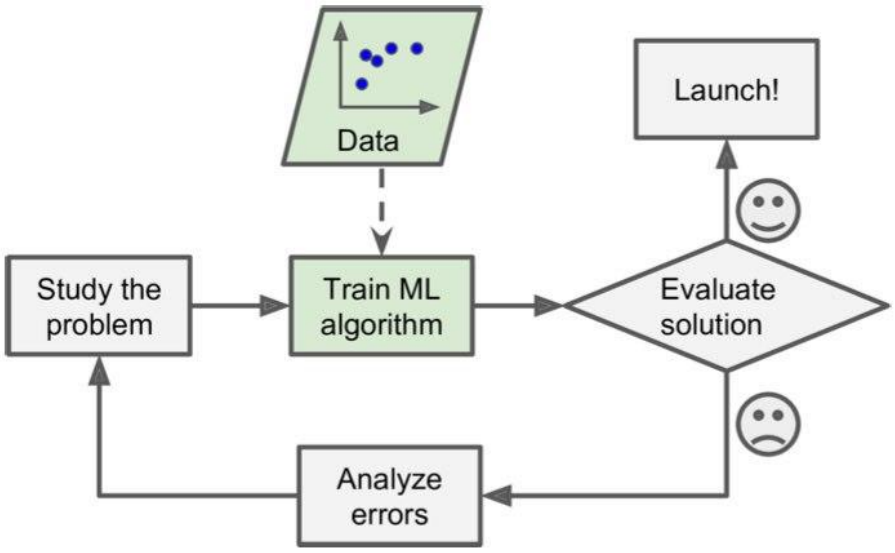


Figure 3.5: The ML approach [34]

Now let's try to solve the task of the spam email filter by imagining programming an ML code. If we wanted to make a spam filter with ML software we would have to provide the program with a large amount of emails with the label "SPAM" or "NOT SPAM" and then, through a mathematical model, the code will be able to automatically to detect the patters needed to classify the emails. So, in an ML code, we do not write the algorithm rules but we provide the program with the

answers, and it will "understand" how to solve it. On the other hand, tasks such as image recognition necessary to solve the "Rock paper scissors problem" cannot be performed effectively with classic programming technique, making machine learning technology the only possible solution for this type of tasks.

(a) Traditional program logic (b) ML program logic

Figure 3.6: Traditional and ML logics

Summarizing the advantages of a Machine Learning program:

- Automatically recognize patters by analyzing the database
- Ability to update the program rules automatically by analyzing new data
- Solve complex problems that cannot be approached using a classic programming logic such as voice and image recognition, for example.

3.1.2 Learning Types

A machine learning program to work needs of a large database of data concerning the problem to be solved. But, according to the structure of the database and based on the task that we have to solve,the most suitable training method can change.

The different learning approaches are:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised Learning

In supervised learning, the data which we use to feed the program contain also the information of the correct solution, called labels. A common task for ML software that uses this type of training is *classification*. The example of the email filter described above is a type of classification supervised learning code: the users tell the program whether an email is a spam or not and the code trains with this information that contains the correct label.

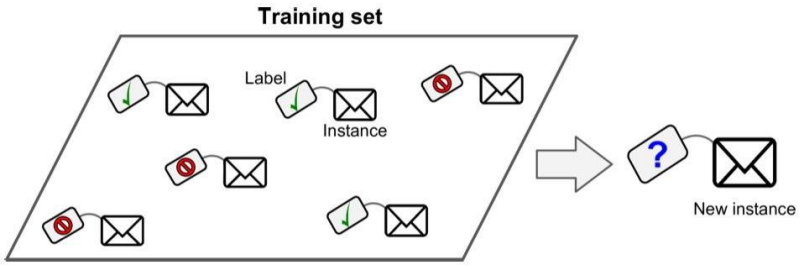


Figure 3.7: Supervised Learning [34]

Some common tasks that need this learning approach are those that have a numerical value as output, called *regression*. An example could be a program that has to predict the real estate evaluations of a house based on a database composed by other property evaluations, correlated with all the useful data (square meters, number of rooms, floors, but also the distance from the sea, proximity to the city center, etc.).

Below there are some examples of supervised learning algorithms:

- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forest
- Neural Network

Unsupervised Learning

The data of a unsupervised learning machine learning code does not have the correct label associated to each instance. Instead the software will learn how to use the data finding autonomously the various categories of the classification, thanks to a mathematical statistical model suitable to solve that problem. Below there are some examples of unsupervised learning algorithms:

- Clustering (K-means, DBSCAN, HCA)
- Visualization and dimensionality reduction (PCA, Kernel PCA, Locally-Linear Embedding LLE)
- Anomaly detection and novelty detection (One-class SVM, Isolation Forest)

Some architectures of the Neural Network type can be unsupervised, for example autoencoders and restricted Boltzmann machines. [34]

Reinforcement Learning

The reinforcement learning adopts a different approach to train a neural network respect the other learning models. In this case the learning system, called the agent, observes "the environment", choose consequently the action, and then, according to the result and consequence of it, it gets a positive (reward) or negative (penalty) feedback. Thanks to that feedback the code will update the hyperparameters to better optimize the network to solve the task. Hence, the next action of the software will consider the feedback obtained previously. The algorithm will have to look for the best strategy, called policy, that allows it to obtain the highest number of rewards. A common application of this type of machine learning software is robotics and related handling. An example of reinforcement learning is the Google AlphaZero algorithm, able to learn the game of chess, shōgi, and go, overcoming the playing power of world champion programs in their respective disciplines, with only a few hours' training. [35]

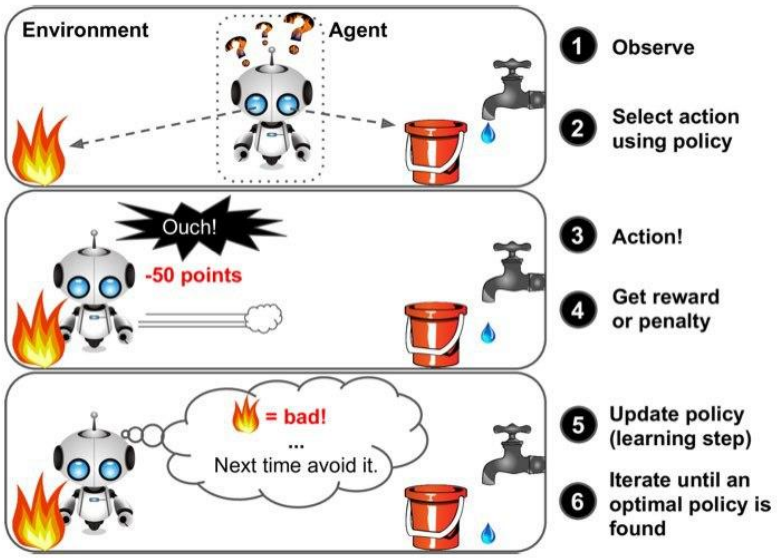


Figure 3.8: Reinforcement Learning [34]

3.1.3 Main Challenges of Machine Learning

The successful design of a Machine Learning algorithm depends on a delicate balance between the mathematical structure of the algorithm and the amount and quality of data that feeds it. The difficulties in building an ML software can come either from the database or from the chosen architecture.

A typical problem is the insufficient dimension of the training database: to correctly learn to solve the task the ML code needs a large amount of data about the problem, which is not always available. To be able to generalize well, it is also crucial that the training dataset is representative of all the cases we want to learn. Obviously, if the training data is full of errors, outliers, and noise it will be more difficult to find the patterns to generalize. It is therefore very important to pay attention to the training database.

Overfitting

A concrete risk of a ML algorithm is to overgeneralize the patterns found by the numerical model. This problem is called *overfitting*: in this scenario the model performs well with training data, while it encounters problems working with validation data, not being able to correctly generalize the problem. In the case of Neural network ML algorithms, the causes that can lead to this problem are:

- Noise in the training database
- A database too small (Which introduces sampling noise [34])
- A network structure not suitable for the problem in question (number of layers, neurons, activation functions)
- Excessive training (e.g. too many epochs)

The possible solutions to solve this problem are therefore:

- The simplification of the model, reducing the parameters
- To gather more training data
- Reduce the noise, e.g. by eliminating outlier instances.
- Increase the **Dropout** of the layers

The *Dropout* is a simple technique of regularization for deep neural networks. Its operation principle consists in randomly "shutting down" different neurons for each training step (including the input neurons, but always excluding the output one). Dropped out neurons are totally ignored during the respective training step, but

they may be active during the next one, while others, who previously contributed to the prediction, will be turned off in subsequent turns. The idea behind this algorithm is that the neural network, forced to work each time with a different part of all its neurons, should better generalize the learning information, making the neurons' hyperparameters then more versatile. The parameter that regulates this optimization is the probability p that the single neuron is temporarily dropped out, called "*Dropout Rate*". A typical dropout value is 50% [34].

Underfitting

An opposite problem to overfitting is *underfitting*. It shows up when the algorithm is too simple compared to the complexity of the problem, and is not able to detect all the patterns useful to solve the task present in the training dataset. The possible solutions are [34]:

- Select a more powerful model, with more parameters
- Feeding better features to the learning algorithm (feature engineering)
- Reducing the constraints on the model

3.2 Deep Neural Networks

One of the most famous technique of machine learning is the Neural Network. The neural network is a programming algorithm developed to simulate the human brain system for Machine learning applications. The ultimate vision of this technology is to create an artificial intelligence capable of simulating the human mind. Behind the Neural Network mathematics theory there is the Rosenblatt's perceptron algorithm, proposed by Frank Rosenblatt in 1958 [31], which laid the foundation for the subsequent development of this technology. At that time it was not possible to development this programming technique, because it requires a large amount of data and considerable computational capabilities, not yet available in the 50th.

Deep learning (also known as deep structured learning) is a class of machine learning algorithms based on artificial neural networks with representation learning, that uses multiple layers to progressively extract higher level features from the raw input. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality.

A *Deep Neural Network* (DNN) is an deep learning artificial neural network composed by multiple layers between the input and output layers. [**?**]. The Rosenblatt's linear perceptron cannot be a universal classifier, while a network with a non-polynomial activation function with only one unbounded width hidden layer can on the other hand so be. Hence, a Deep Neural Networks is theoretically capable of learning any mathematical function with enough data, and some types of neural networks can be defined "Turing Complete". Turing completeness refers to the ability of an algorithm to simulate any other possible algorithm (or Turing machine).

3.2.1 Introduction Neural Networks

The neural network algorithm is inspired by the functioning principles of the neurons in our brain. Neurons are the cells that make up our nervous system. Neurons are interconnected by axons and dendrites, and the region of connection between axons and dendrites is called synapses. The strength of the synapses connections changes in response to different external stimuli. This variation is the basis for the functioning of learning in living beings.

To simulate the biological system, the artificial neural network presents basic computational units, called *neurons*, interconnected through *weights*. Each given input of a neuron is transmitted to the next interconnected one scaled according to the weight's coefficient presents on that link. An artificial neural network works by propagating the input information through the neurons, elaborating them according to the weight coefficients, until reaching the output neurons, where the information will be the result of the algorithm. The network learning is based on the variation

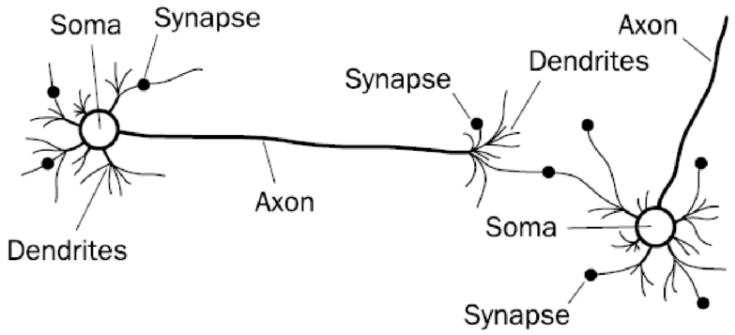


Figure 3.9: Biological Neural Network [33]

of the weight coefficients that process the information.

The database reserved for training allows you to generate positive or negative feedback based on the predictions made by the code and, consequently, improve the interconnections between neurons, increasing or decreasing the weight of these. These corrections on those coefficients aim to minimize the *loss function*, which will evaluate the goodness of the algorithm result by comparing it with the label information.

The most important skill of this statistical mathematical method of machine learning is the possibility, once trained the network with the training data, to make predictions on other different data that previously the neural network has not seen before. Let us imagine that we want to create a software able to classify images of dogs and cats. After training the code with a large number of figures of these animals, the program will be able to find patterns that will allow him to identify and distinguish a new cat that the code has never seen before. This ability to accurately execute functions on unseen input by training it with a finished database of information is referred to as *model generalization*. Thanks to it the ML software is able to generalize its learning from seen training data to unseen examples.

Deep learning is the best machine learning algorithm solution when is available a large amount of data and high computational capacity , as shown in the figure3.10.

3.2.2 Deep Neural Network Applications

Since the 2010s, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training deep neural networks that contain many layers of non-linear hidden units and a very large output layer. [53] The areas of application are therefore boundless, and that make this tool of incredible interest in various areas of scientific research.

The ability of these software to automatically recognize patterns within the database and learn how to use them allows to use neural networks in recognition applications such as Image Recognition or Automatic speech recognition.

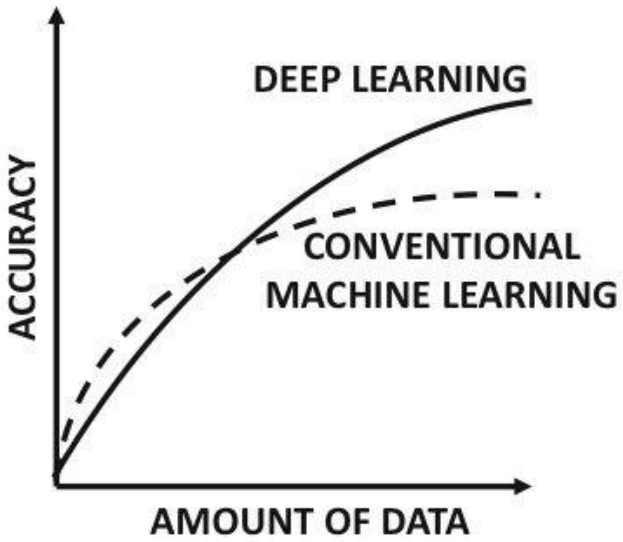


Figure 3.10: Illustrative comparison of typical ML with a DL code [34]

Target marketing is a clustering task that involves market segmentation, where we divide the market into distinct groups of customers with different behavior[56].

In the financial world neural networks have been applied successfully to predict financial indicators, like derivative securities pricing and hedging, futures price forecasting, exchange rate forecasting, and stock performance. Traditionally, statistical techniques have driven the software. These days, however, neural networks are the underlying technologies driving decision making.

3.2.3 Single Computational Layer: The Perceptron

To clearly understand the structure on the neural network it useful to start with the easiest possible version: the Perceptron 3.11 [40]. A neural network can have multiple layers where each one presents many neurons. Different units of the various layers can present many types of interconnection: for example, all the neurons of two consecutive layers can be totally interconnected, or each neuron can present a link with only one group of the elements of the next row. In the simplest scenario, we have only two-layer: the input and the output layer. The input layer is not included in a neural network layer count. If a neural network has more than one layer, with others between the input and the output one called *hidden layers*, it is called *Deep Learning Neural Network*.

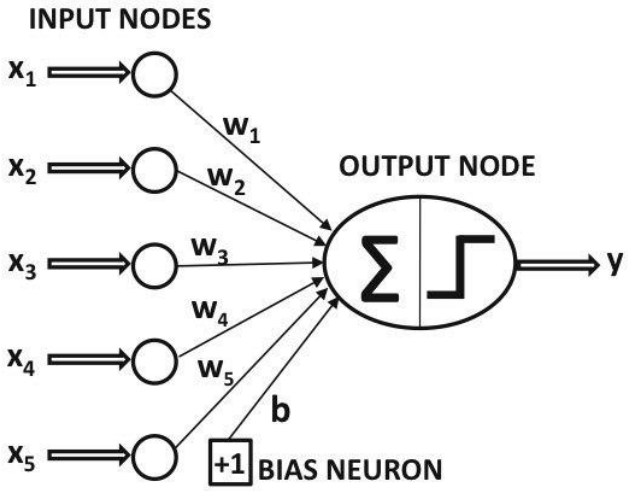


Figure 3.11: The basic architecture of Perceptron with bias [39]

$$\hat{y} = sign(\overline{W} \cdot \overline{X} + b) = sign(\sum_{j=1}^{d} w_j \cdot x_j + b) \tag{3.1}$$

The **Perceptron** is composed by a single input layer, with multiple neurons, and a single neuron output layer. Each train instance has the form $(\overline{X}, y)$, where the input layer contains d nodes that transmit the d features $\overline{X} = [x_1...x_d]$ through edges of weight $\overline{W} = [w_1...w_d]$ to an output node. The y contain the *observed value* $y \in \{-1, +1\}$ of the binary value class, while $\hat{y}$ is the value predicted by the network respect to the instance input $\overline{X}$. The observed value is the correct label value associated with each instance, and the goal of the program is to correctly predict it with new data. The activation of the output neuron is the result of the *sign* activation function applied to the result of the linear system $\overline{W} \cdot \overline{X} + b = \sum_{j=1}^{d} w_j \cdot x_j + b$. The error of the prediction is $E(\overline{X}) = y - \hat{y}$, and it can assume

the discreet values $\{-2,0,+2\}$. If the E error does not turn out to be 0 for a given instance, so the prediction made by the net will turn out to be wrong, it will be necessary to tune, through the decreasing gradient, the value of the weight coefficients, or the bias, to improve the algorithm.
When there is an invariant part into the prediction, it is necessary to use *bias neuron* b in the network. The bias allows you to shift the activation function by adding a constant to the input. Bias in Neural Networks can be thought of as analogous to the role of a constant in a linear function, whereby the line is effectively transposed by the constant value.
The goal of the perceptron algorithm is to minimize the error in prediction, even if a formal optimization formulation was not presented at that time. The basic perceptron algorithm can be considered as a stochastic gradient-descent method, which implicitly minimizes the square error of the prediction.

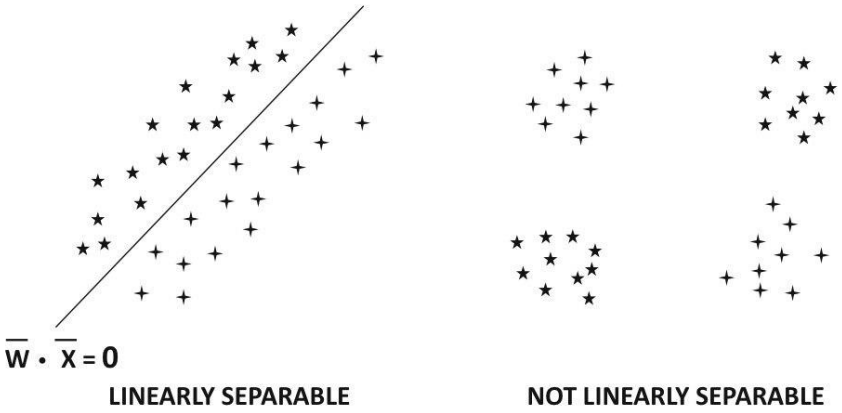


Figure 3.12: Linearly separable data and non-linearly separable data [39]

The perceptron is a linear model, where it defines a linear hyperplane $\overline{W} \cdot \overline{X} = 0$. This algorithm can classify linearly separable data, while it is not guaranteed that the solution converges when the data is not linearly separable. This shows a big limitation of this first neural network solution and can be solved by using more complex networks architectures.

3.2.4 Description of Deep Learning Neural Network Structure

Deep learning is a class of machine learning algorithms that is composed by multiple layers to progressively extract higher level features from the input data. The adjective "deep" in deep learning comes from the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier,

and then that a network with a nonpolynomial activation function with one hidden layer of unbounded width can on the other hand so be. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions.

When all the neurons in a layer are connected to every neuron in the previous layer, it is called a *fully connected layer* or a *dense layer*.

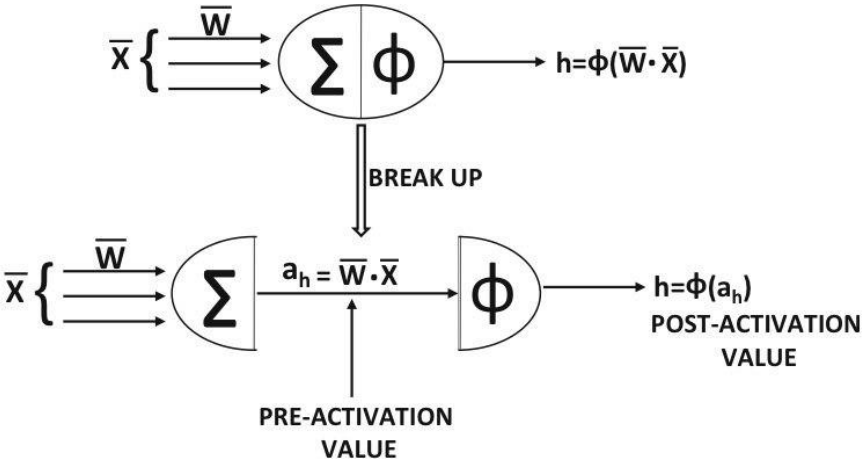


Figure 3.13: Pre-activation and Post-activation values within a neuron [39]

Each neuron compute two different functions in the single node 3.13: the first one is the linear system $\overline{W} \cdot \overline{X} + b$, which allows you to cast the *pre-activation value* a_h . Then the appropriate activation function Φ is applied to a_h , which allows to calculate the output of the neuron, the *post-activation value*.

3.2.5 Activation Function

The choice of the activation function of the various layers that make up the network is one of the most critical parts of the ML code architecture design. One of the main factors to take into account is the type of target you want to predict: if the nameplate is the probability that the instance falls into one of two different options of a binary class, a suitable activation function could be *sigmoid*. Instead, if the target is a real variable, you can opt for the simplest activation function on the last layer, the *identity*. The choice of the appropriate activation also allows the network to solve problems of a non-linear nature. Some types of non-linear activation functions are *sign*, *sigmoid*, or *hyperbolic tangent*. We use the symbol Φ as a notation to refer to activation functions. It is possible to notice that most activation functions saturate beyond a certain abscissae value, so the activation does

not grow over a high absolute value. These functions are suitable for classification problems, and the last layer must adopt one of them.

The non-linear activation functions, combined with a multilayer network architecture, allows to create more powerful combinations of different activation functions. In the last layer that generates the output, *squashing functions* are typically used, as they map the result value from an arbitrary range to bounded one. If the network only uses linear activation functions, even if it has more than one hidden layer, it is not able to solve nonlinear problems, and it has the same capacity of a single linear network layer.

Identity Function

The simplest activation function is the *identity*, which allow you to solve **only** linear problems, making the output simply a linear combination of the activation values of the previous layer.

$$\Phi(v) = v \tag{3.2}$$

Can also be used when the target is discreet, and a smoothed surrogate loss function needs to be set up.

Sign Function

$$\Phi(v) = sign(v) = \begin{cases} -1 & if \quad v < 0 \\ 0 & if \quad v = 0 \\ +1 & if \quad v > 0 \end{cases} \tag{3.3}$$

The sign is a non-differentiable function, with step-like discontinuity point in the origin. Furthermore, the sign function takes on constant values over the totally of the domain (except in 0, where there is the jump), and therefore the exact gradient takes on zero at all differentiable points. The results in a staircase-like loss function, which is not suitable for a gradient-descent.

Sigmoid Function

$$\Phi(v) = \frac{1}{1 + e^{-v}} \tag{3.4}$$

The output of the sigmoid activation function is between 0 and 1, which is helpful in performing computations that should be interpreted as probabilities. Furthermore, it is also helpful for probabilistic outputs and constructing loss functions derived from maximum-likelihood models. [39] It is a function that can solve problems that present a non-linearity.

Tanh Function

$$\Phi(v) = \frac{e^{2v} - 1}{e^{2v} + 1} \tag{3.5}$$

The hyperbolic tangent has a form like the sigmoid activation function; in particular, the two functions are related to the subsequent relationship:

$$tanh(v) = 2 \cdot sigmoid(2v) - 1 \tag{3.6}$$

Since it has a output values between [-1;1], it is preferable to sigmoid when the output you want to obtain can also take negative values. Furthermore, its mean-centering and its wider gradient also make it easier to train than sigmoid. It is a function that can solve problems that present a non-linearity.

ReLU

$$\Phi(v) = max\,\{v,0\} \tag{3.7}$$

The ReLU function is a piecewise linear activation function. This activation function has replaced, in modern architectures, the sigmoid and soft tanh activation function, thanks to their better training predisposition in multi-layer neural networks. [39]

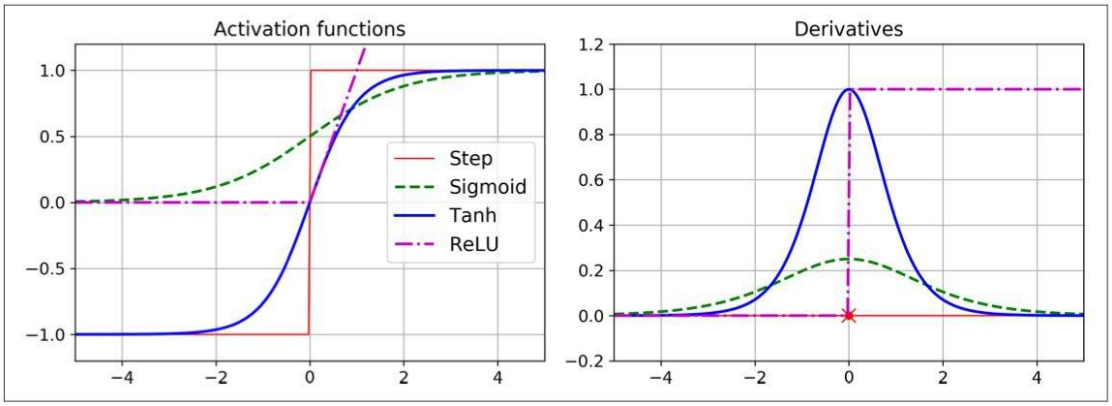


Figure 3.14: Activation functions and derivatives [34]

3.2.6 Back propagation learning and Gradient descent

The basis of neural network learning is the minimization of the *cost function* C, or *loss function*, defined, for this problem, with the least-squares function:

$$Minimze_{\,W}\, C = \sum_{(X,y)\in D} (y - \hat{y})^2 \tag{3.8}$$

With D representing the dataset.

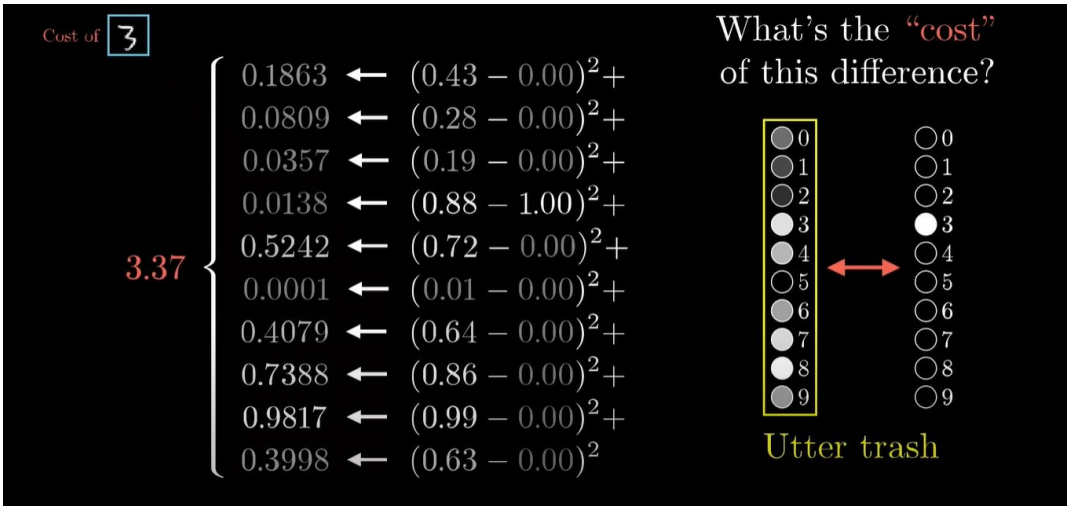


Figure 3.15: Cost calculation of instance relative at a image of "3" [42]

To achieve this, it is necessary to fine-tune the activation values of the output of the last layer of the network. However, this cannot be done directly. The variables that we can adjust to improve our code are all the weight coefficients and all the biases present in our neural network architecture. In the case of a multilayer network, the problem of loss is a complex function of all the weight coefficients and bias present in the previous layers. The *backpropagation algorithm* is used to calculate the gradient of the composition function. The backpropagation algorithm takes advantage of the chain rule of differential calculus, and it calculates the gradient by adding the local gradients of the error function generated over the various paths from a node to the output. [39] The algorithm consists of two parts:

- *Forward Phase* - In this phase the network processes, with the current weight coefficients and biases, the input data through all layers until it obtains the prediction. Subsequently the result is compared with the label of the relative instance to calculate the cost function value. The derivative of the loss function will now be used by tracing the network in the opposite direction to adjust and tone all the algorithm parameters 3.15.

- *Backward Phase* - In this phase the algorithm adjusts all the parameters of the network by using the chain rule of differential calculus, updating the coefficients with the C gradient. In this case the algorithm starts from the last layer, the output.

During the training, the network is fed several times with the whole training

database: each iteration is called *epoch*. The number of epochs chosen is an important parameter of the construction of the neural network: a small number of iterations can lead to a state of underfitting, while an excessive number of epochs can instead lead to the problem of overfitting. Let us now analyze in detail the mathematics that allows the learning of the network. The following explanation is inspired by the Youtube video "Backpropagation calculus | Deep learning, chapter 4" by Grant Sanderson [42], from which the following images are taken.

Let us imagine analyzing the last layer of a neural network that allows to recognize handwritten numbers. The images are composed by 748 pixels, so the first layer has the same number of neurons, while the last output layer is composed by 10 units, equal to the number of possible classifications (from 0 to 9). In the following figure 3.16 we observe the results of one instance, that is the "two" handwritten present in the upper left corner. We can observe that, with this instance, we want to improve the activation result of the output neuron corresponding to the number 2 prediction value, while bringing the others activations values to 0.

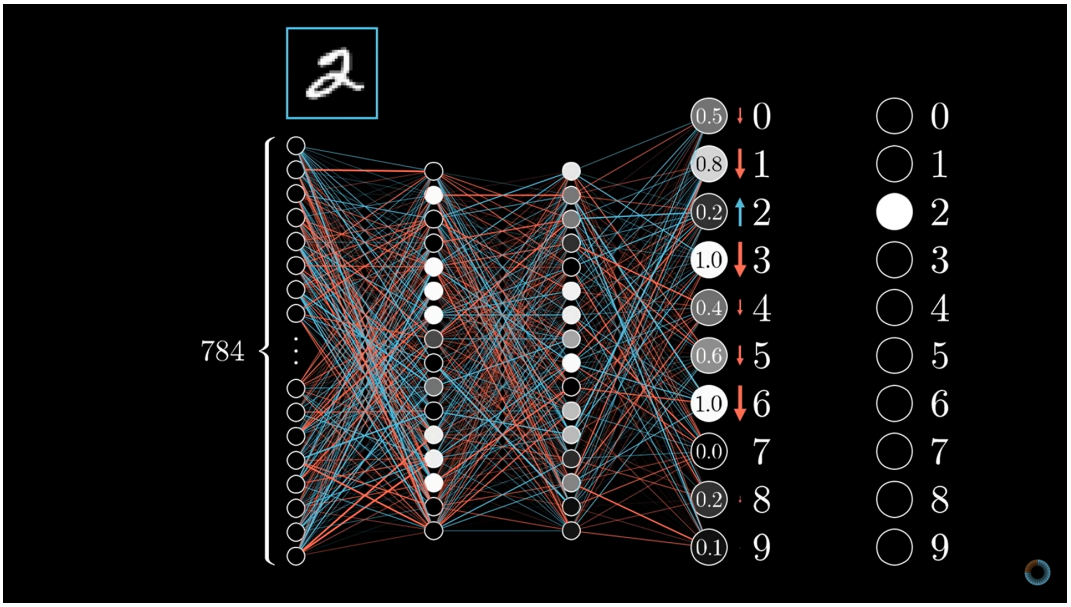


Figure 3.16: Improved output activation [42]

Let us now focus on the variables that influence the activation of neuron "2", with the aim of increasing the score to one: to improve the output of the program we can either increase the weight of the links, increase the bias or increase the activation of the neurons that make up the previous layer. To carry out this last operation, in turn, it is necessary to act on the coefficients that regulate the activations of each single neuron connected to our output unit.

Let's see how the effect obtained by increasing the w_j weight is closely related

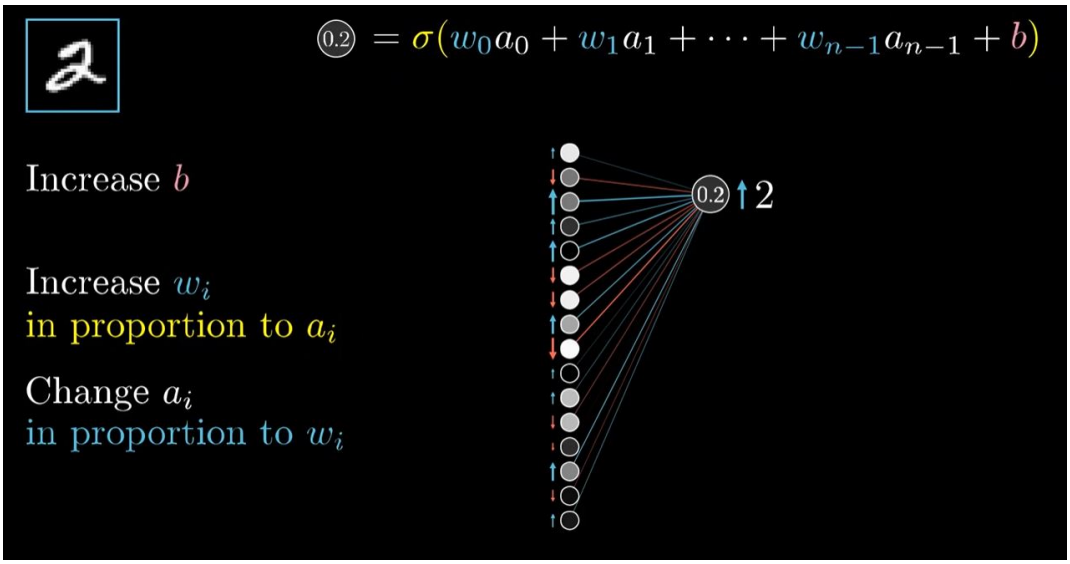


Figure 3.17: Improved output activation [42]

to the a_i activation value of the corresponding neuron and vice versa, since the two numbers are multiplied before being used by the activation function. So w_i must be increased proportionally to a_j, and the same for activation. We can already see the close correlation between the activation of the previous neuron and the strength of the link connecting it to the neuron of the next layer. Weight coefficients can also take on negative values (red link) leading to act in the opposite direction of activation value.

Specifically considering to adjust a_i activation value of the relative neuron 3.18. This is not only connected to the output of the 2, but also to all the other neurons that regulate the classification of the other numbers. Going backwards trying to improve the predictions of these other outputs, there will be many corrections to the activation values, and the back-propagation algorithm has to consider all of them to act next on the previous layer. In the end, by summing all the variations calculated on each neural network coefficient, using all the training data in the D dataset, we will get the value of the C cost function gradient that will improve the network prediction 3.19. For computational reasons each step of the gradient will not include the entire database, but only a reduced part of training data called *minibatch*, whose size is a parameter to choose in the construction of the architecture of the net.

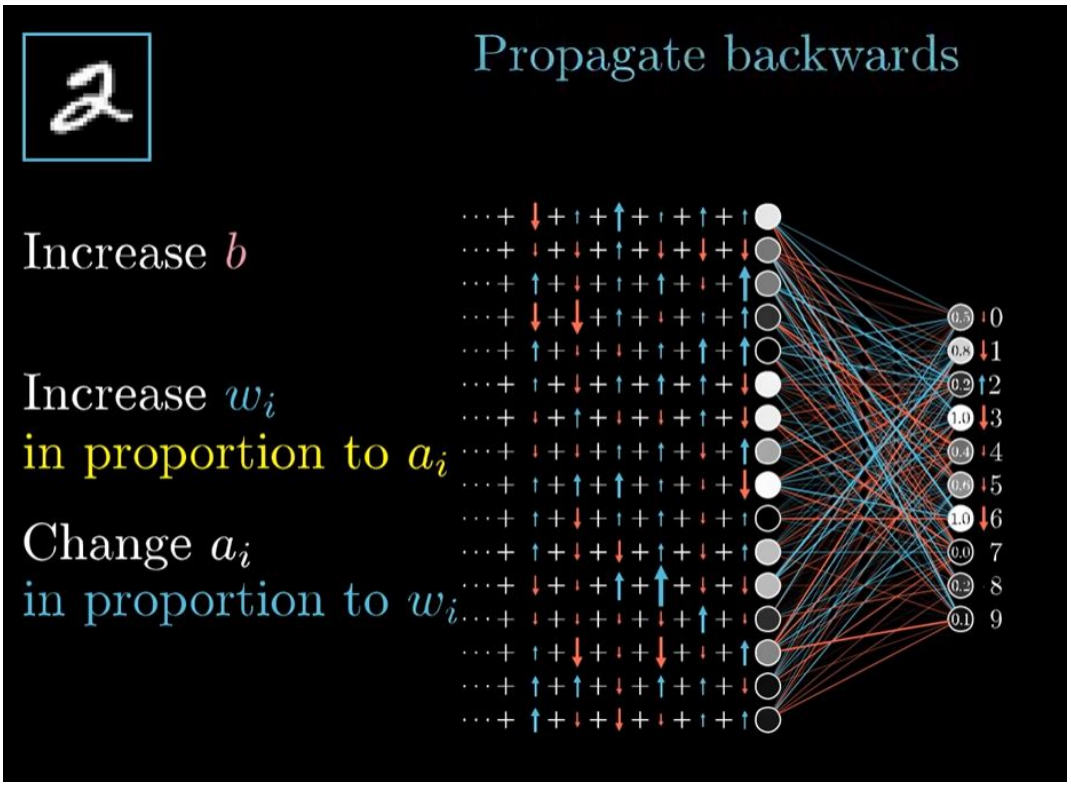


Figure 3.18: Improved output activation [42]

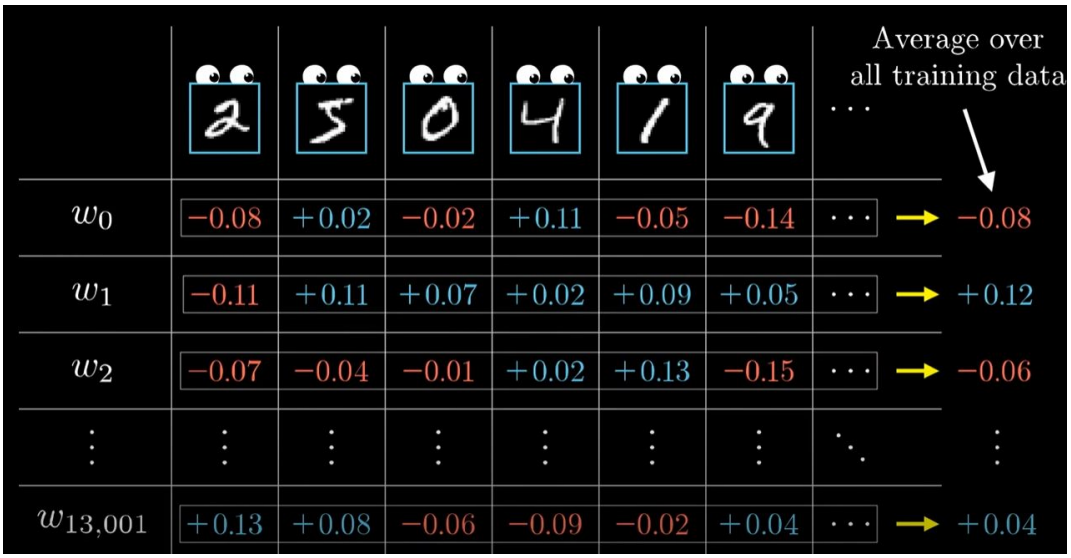


Figure 3.19: Loss function gradient calculation [42]

Analyzing now the equations that regulate and control the algorithm, let's consider the connection between only 2 neurons a^L and a^{L-1} (L indicates the layer, it

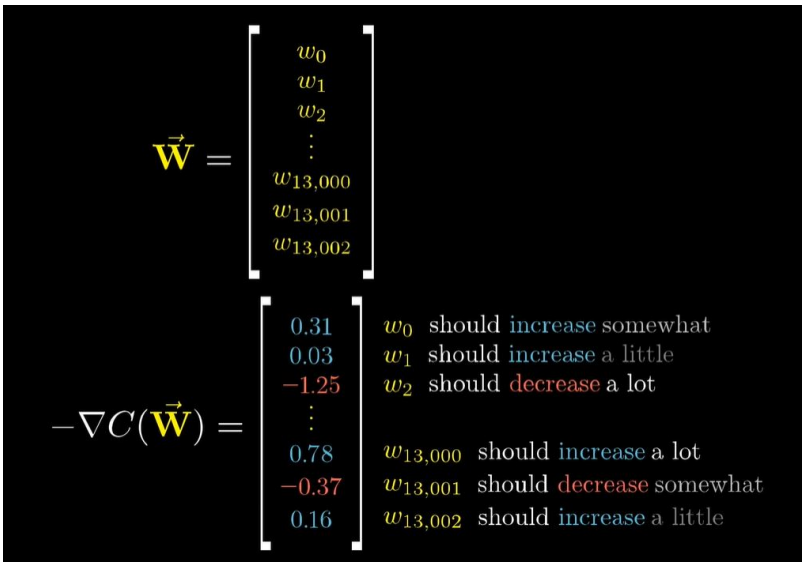


Figure 3.20: Loss function gradient calculation [42]

is not an exponent). The layer L is the network output, and y is the correct label associated with the instance. The value of cost function related is:

$$C_0(w_1, b_1, ..., w_L, b_L) = (a^{(L)} - y)^2 \tag{3.9}$$

$$\begin{aligned} z^{(L)} &= w^{(L)} a^{(L-1)} + b^{(L)} \\ a^{(L)} &= \sigma(z^{(L)}) \end{aligned} \tag{3.10}$$

Where σ is a generic activation function and $z^{(L)}$ is the pre-activation value.

Let's study the dependency of C_0 output on its variables, analyzing the partial derivatives with $w^{(L)}$, $b^{(L)}$ and $a^{(L-1)}$.

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{3.11}$$

The partial derivative of the least-squares cost function from the activation applies:

$$\frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y) \tag{3.12}$$

The derivative relationship between a and z depends on the activation function chosen. We can here observe here the importance of the activation function in the learning process the neural network: choosing the sign function, the derivative

is worth 0 in all the differentiable domain, canceling the product and then the backpropagation.

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)}) \tag{3.13}$$

Let us now analyze the last partial derivative that relates the pre-activation value to the weight of the link.

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)} \tag{3.14}$$

The last derivate shows that the correlation between a small nudge of the weight and the influences on the last layer pre-activation depends on how strong the link with the previous neuron is. This derivate shows the idea that "neurons that fire together wire together", that is the connection with the biological theory of neuron.

The end result is:

$$\frac{\partial C_0}{\partial w^{(L)}} = a^{(L-1)} \cdot 2(a^{(L)} - y) \cdot \sigma'(z^{(L)}) \tag{3.15}$$

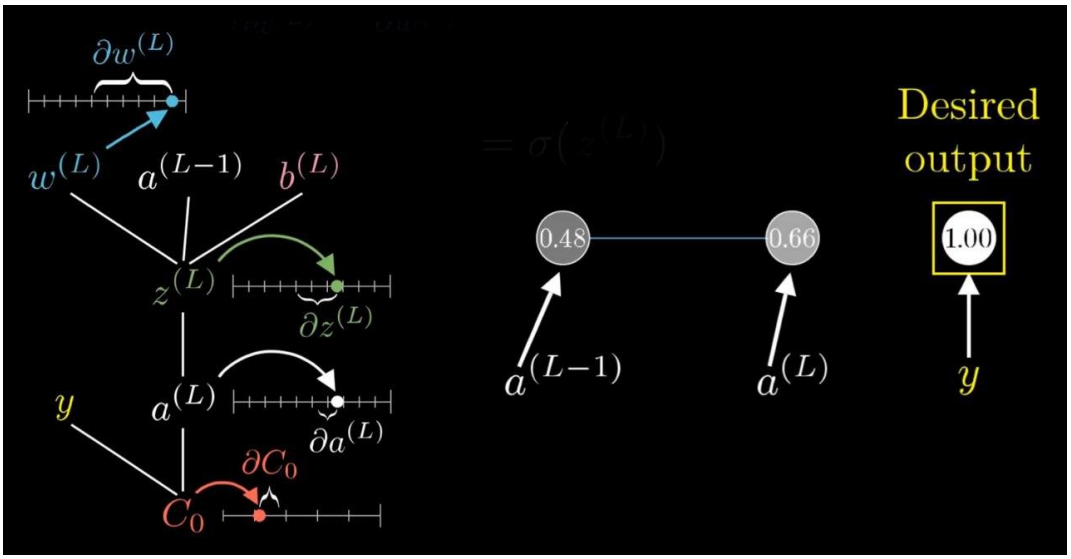


Figure 3.21: Loss function gradient calculation [42]

The derivative of the cost function calculated in this way refers only to a specific training example, since the full cost function involve requires averaging the results

with all training database.

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{1}{n} \sum_{k=0}^{n} \frac{\partial C_k}{\partial w^{(L)}} \tag{3.16}$$

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \cdot \\ \cdot \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix} \tag{3.17}$$

With n number of training examples in the D dataset, and ∂C_k result of the loss function related to the k instance.

In the same way we can calculate the partial derivative of the cost function with respect to the bias variable.

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{3.18}$$

With:

$$\frac{\partial z^{(L)}}{\partial b^{(L)}} = 1 \tag{3.19}$$

The end result is:

$$\frac{\partial C_0}{\partial w^{(L)}} = 1 \cdot 2(a^{(L)} - y) \cdot \sigma'(z^{(L)}) \tag{3.20}$$

To understand the backpropagation idea we have to analyze how sensitive is the cost function to the activation of the previous layer

$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = \tag{3.21}$$

$$= w^{(L)} \cdot 2(a^{(L)} - y) \cdot \sigma'(z^{(L)})$$

With:

$$\frac{\partial z^{(L)}}{\partial a^{(L-1)}} = w^{(L)} \tag{3.22}$$

Although we can not directly affect the $L-1$ neuron activation, we can iterate the same chain rule backward again, until to reach the initial layer.

Generalizing now the model just built in case there are more neurons on the layers that make up the architecture of the network.

Let's see how the previously written formulas remain almost unchanged, simply with a different notation. Let's refer with k to a specific neuron of previous layer and with j to the next layer 3.22.

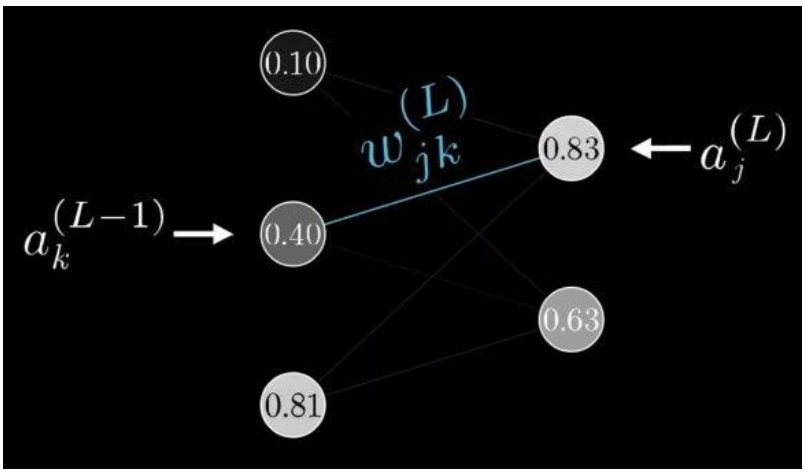


Figure 3.22: Generalization of the Model when there are more neurons [42]

$$C_0 = \sum_{j=0}^{n_L - 1} (a_j^{(L)} - y_j)^2 \tag{3.23}$$

$$z_j^{(L)} = ... + w_{jk}^{(L)}\, a_k^{(L-1)} + ... + b_{jk}^{(L)} \tag{3.24}$$

$$a_j^{(L)} = \sigma(z_j^{(L)}) \tag{3.25}$$

$$\frac{\partial C_0}{\partial w_{jk}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial w^{(L)}{}_{jk}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} \tag{3.26}$$

What changes is the partial derivative with respect to the activation 3.27 of a neuron of the previous layer, which is found to influence the cost function by acting on more neurons of the next layer to which it is connected.

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} \tag{3.27}$$

3.2.7 Data Structure - Trial, Test and Validation Dataset, mini-batches

The organization of the databases used to feed and evaluate the training of the architecture of a neural network is of crucial importance in the design of the machine learning software. The original Dataset has to be divided into three subgroups: *Training, Test, and Validation Dataset*. The data of those three groups have to be homogeneous and it has to contain an adequate and representative number of each type of data to well generalize the problem. It is important to distinguish the database needed for the realization of the network from the one used for the prediction and evaluation of the final neural network code. The latter database may also contain data selected in different ways with different information. Let us analyse the function of each individual group. [50]

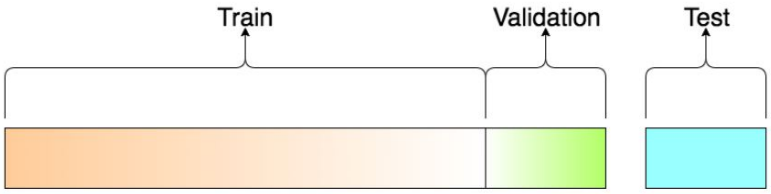


Figure 3.23: Trial, Test and Validation Dataset proportions [50]

The *Training Dataset* is the actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

The *Validation Dataset* is the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. Instead in the design of the neural network validation data are used to fine-tune the model hyperparameters. Hence the validation set results are applied to update higher level hyperparameters. So, the validation set affects a model, but only indirectly. The validation set is also known as the Dev set or the Development set. This makes sense since this dataset helps during the "development" stage of the model. It is common to get slightly lower performance on the test set than on the validation ser, because the hyperparameters are tuned on the validation set, not the test set.

The *Test Dataset* is the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets). The test set is generally what is used to evaluate competing models. The validation set is released initially along with

the training set and the test set result of the model decides the best architecture. Many times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face.

The proportion between the dimensions of these three datasets depends, first, on the total number of samples in your data and second, on the actual model you are training. For example, models with few hyperparameters need a small validation dataset, while large neural networks need a lot of validation data.

The use of the validation dataset is optional, but it allows us to identify possible problems while learning the code. The *learning curve* represents the accuracy and the loss of both the training set and the validation set with the corresponding Epoque in abscissa. With this plot it is possible to identify if the program is generalizing too much training data, leading to the problem of overfitting. To have a correct training of the network, the results of the validation data have to follow the same trend as the training data, with slightly lower values.

3.3 Classification

As mentioned above, the most common task of supervised learning is regression (predict a value) and classification (predict a class). Analyzing the classification in detail, let us focus on a type of binary classifier, which can then be used together with others for the realization of more complex algorithms.

3.3.1 Confusion Matrix

To build a ranking algorithm it is of fundamental importance to measure its performance. One of the most important tools in this regard is the Matrix confusion 3.24. This matrix allows to measure the performance of a binary classifier. It is used after achieving the prediction results with the neural network trained.

The rows of the matrix represent the *actual class* contained in the label, while the columns represent the *predicted class*. Let's consider the deep learning program object of this thesis: the code has to predict whether the instance corresponds to one where the rub is present, or where it is not.

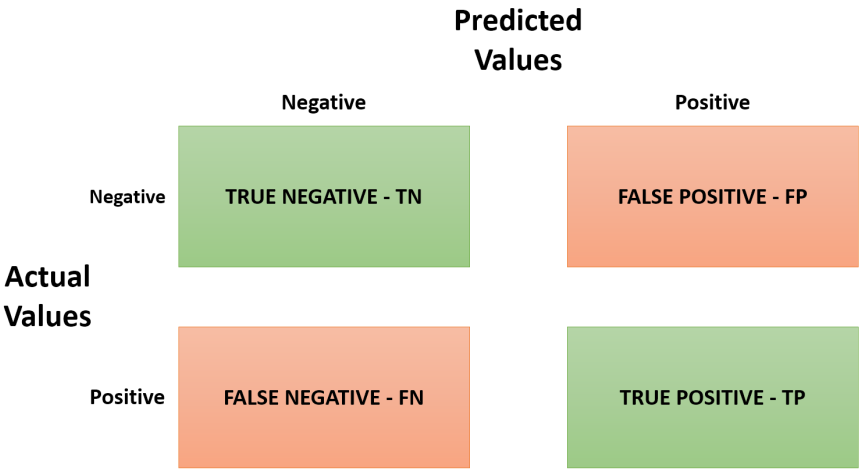


Figure 3.24: Confusion Matrix

If the neural network reveals, through the information contained in the sample, that there is the rub but in reality it is not present, we are faced with a *False Positive FP*. While when it detects that there is no rub, but actually in the gas turbine there is contact between stator and rotor, we are faced with a *False Negative FN*. Both FP and FN are wrong classifications, but they can have different relevances: in an

application where a possible malfunction is very dangerous, detecting a not present risk is much less serious than ignoring it when it is present. We can therefore see this other utility of the confusion matrix in the design of the program itself, in the calibration of the prediction. The *True Positives* and the *True Negatives* are both situations when the ML algorithm guesses the correct instance classification.

3.3.2 Precision and Recall

Once the matrix is complete, we need to analyze the results to understand how the algorithm is working. the two most important parameters to calculate are *Precision* and *Recall*.

The Precision parameter allows you to highlight the accuracy of the prediction. Once the positive classification has been made, the accuracy allows to know the probability to have a correct prediction, namely if it corresponds to the current positive labeling.

$$Precision = \frac{TP}{TP + FP} \tag{3.28}$$

Precision alone, however, is not enough to understand the efficacy of machine learning code. It may be that this parameter has a very high score, even close to 100%, but we cannot know how many positive data the program is ignoring.

$$Recall = \frac{TP}{TP + FN} \tag{3.29}$$

With $Total\ Positive\ Data\ =\ TP + FN$

The Recall is a is a parameter that allows to know the percentage of correct samples have been "seen" by the program and how many have not. In fact, the sum of *True Positive* and *Fake Negative* returns the total number of positive elements contained in the database,

However, these two parameters are related: if you want to improve the accuracy, the percentage of recall will drop. This also applies backwards. When designing an algorithm, you may prefer recall or precision. Let's take an example of a software, a children's video filter. We want the program to block as many dangerous videos (high precision) as possible, even if this means rejecting videos that did not pose a risk to children (low recall).

The parameter that allows these two coefficients to be correlated is the *threshold*. Analyzing the situation of the neural network for a binary classification, on the last layer there will be a neuron, which will produce as output the score related to the classification of that instance. The threshold allows to determine the limit beyond which the prediction will be positive.

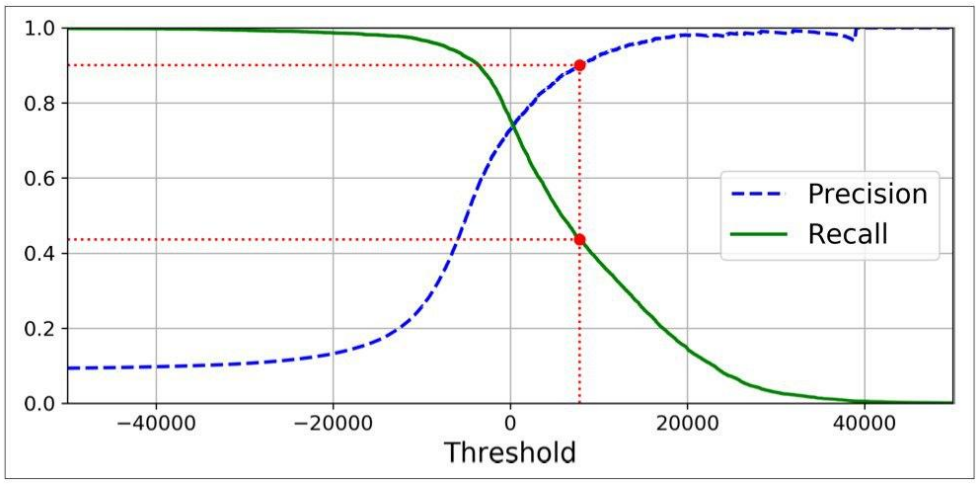


Figure 3.25: Threshold VS Precision/Recall [34]

The dependence of precision and recall on the threshold is the basis of the *automatic optimization algorithm* present in the program of this master thesis project. This code varies the threshold per step, increasing it by one step when $Recall > Precision$ and decreasing it when $Recall < Precision$. This cycle is repeated until the relative error falls below an arbitrary threshold. It is necessary to optimize the step and the output threshold according to the model in question.

Another way to select a good precision/recall tradeoff is to represent the two coefficients in a graph and see how they are related.

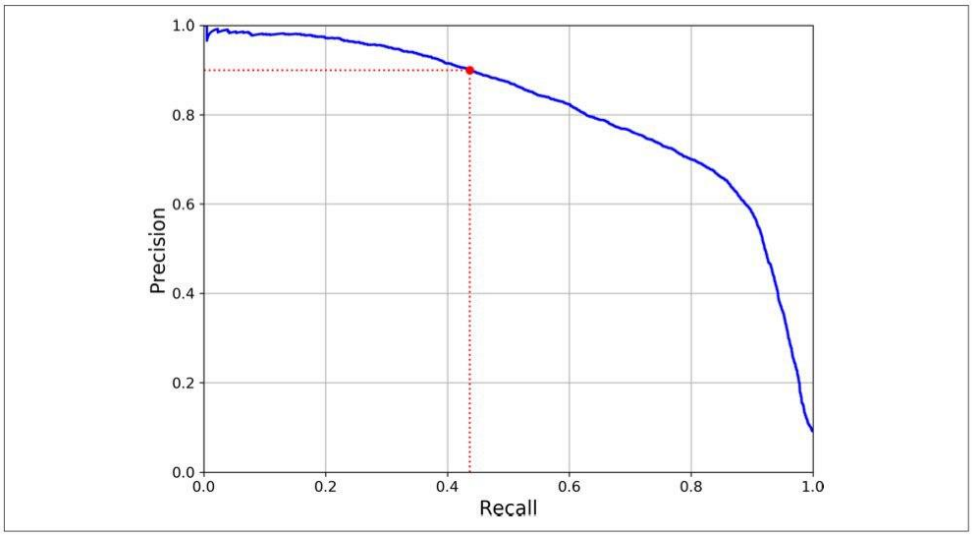


Figure 3.26: Precision/Recall Curve [34]

Chapter 4

EXPERIMENTAL VALIDATION

4.1 Laboratory experiment

4.1.1 Rotating experimental Machine

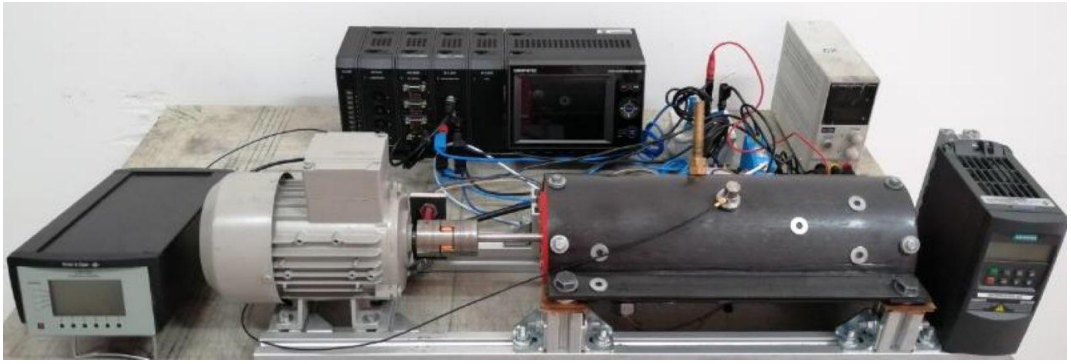


Figure 4.1: Aeroderivative Turbine Model

The experimental model 4.1, designed and built by the mechanical engineering division of ETSII-UPM, presents a 0.55 kW asynchronous electric motor, model Siemens 1LA7073-2AA10. Two metal discs are mounted on the shaft with studs to simulate the rotor of the aeroderivative gas turbine. The maximum shaft rotation speed is 96 Hz, reached with a linear variation of the speed. The axle is made of AISI 4140 steel while the two discs are perforated because 10mm screws will be used to balance the rotating system and then simulate the unbalance. The shaft is supported by two SKF 6000 ball bearings which, in turn, are connected to the crankcase with flanges. The stator element is made of two carbon fiber half-cylinders connected with bolts. Accelerometers located on the outside part of the casing are used to measure the vibrations induced on the structure by the oscillation of the metal axis. The model also has 2 proximity sensors offset by an

angle of 60??ř degrees to draw the precession trajectory of the geometric center, whose signal, however, will not be used in the machine learning program.

The electrical voltage signal generated by the accelerometers passes through a "Bruel & Kjaer 2692C" signal conditioner before being processed by the "GL7000" data acquisition system.

Figure 4.2: Accelerometers Bruel & Kjare (Left) and accelerometer positions in the test (Right)

The accelerometers used are two piezoelectric sensors of the brand "Bruel Kjaer". 4.2. The first is the 4371 with a frequency range from 0.1 to 12600 Hz and a sensitivity of 100 mV/ms^{-2}, while the second is the 4397 with a frequency range from 0.1 to 25000Hz and a sensitivity of 10 mV/ms^{-2}. The installation is carried out by means of washers connected to the casing, The two accelerometers are used together in each pair of positions P1-P2, P3-P4 and P5-P6.

In the table below there are the main data related to the experimental rotating machine

Length of shaft	0.5	m
Radius of shaft	$5 \cdot 10^{-3}$	m
External radius of each disk	$40 \cdot 10^{-3}$	m
Thickness of each disk	$25 \cdot 10^{-3}$	m
Weight of each disk	0.94	kg
Weight of coupling	0.14	kg
Weight of each bearing and bearing support	0.4	kg
Length of casing	0.37	m
Midplane radius of casing shell	$65 \cdot 10^{-3}$	m
Thickness of casing shell	$2.4 \cdot 10^{-3}$	m
Distance between coupling and first bearing	0.12	m
Distance between coupling and second bearing	0.47	m
Distance between coupling and first disk	0.31	m
Distance between coupling and second disk	0.34	m

Table 4.1: Weights and dimensions of rotor rig

4.2 Numerical Model of the Rotation Machine

4.2.1 Model description

The synthetic data to train the machine learning algorithm have been generated through a finite element numerical model (developed by engineer Alejandro Silva [47]), that allows to simulate the behavior of a rotating machine in the presence of a rub-type defect. This allows to create data to train the neural network describing the gas turbine rub malfunction, without compromising system integrity conducting destructive experiments.

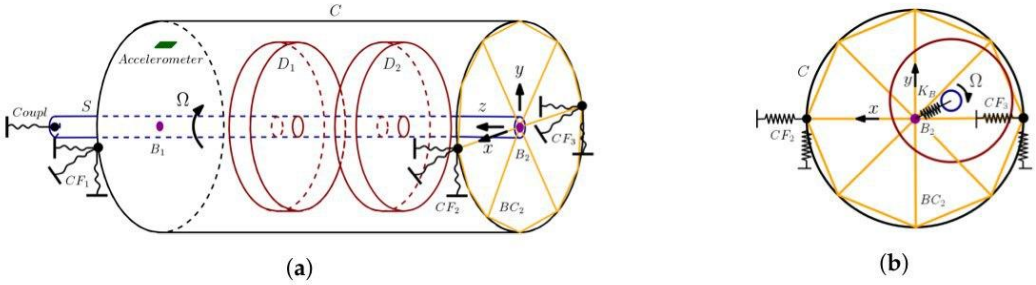


Figure 4.3: The Rotor-flexible casing model [47] (a) perspective view, (b) radial view

The physical elements underlying the numerical model of a rotor-casing system are:

- A flexible, homogeneous, isotropic, linear elastic **shaft** (S, blue)

- Cylindrical rigid solid **disks** (D, red) connected rigidly to the shaft
- A flexible, homogeneous, isotropic, linearly elastic cylindrical shell around the rotor: the model **casing** (C, black).
- Linearly elastic **bearings** (violet). Casing supports to ground, shaft couplings (CF and Coupl, black). Bearing links to casing (BC, yellow)

The shaft and the two discs form the machine rotor, which spins at a constant rotation speed Ω. The rotor is supported by two bearings, B1 and B2, that are radially connected to the machine casing (C, black) modeled as a cylindrical shell. In an areoderivative gas turbine, the casing is attached to solid elements called *frames*, which also house the rotor bearings. The connection between the bearings and the housing is simulated with very stiff bar trusses between bearings and a finite number of homogeneously distributed points in the casing (BC_i, yellow). The sources of excitation of the model can be either the rub or the unbalance: the vibrations generated are then measured by the accelerometers on the stator part (green), which yield casing acceleration versus time. If the basic assumptions hold, the model can be adapted for any material properties, size and number of disks, bearings and supports. The model used to feed the network consists of two disks, two rotor bearing supports on both sides of the casing, an elastic coupling at one end of the shaft and four ground casing supports. The shaft and discs are made of steel, while the housing is made of carbon fiber, and their respective properties have been incorporated into the model.

The model data has been chosen to best simulate the real rotating machine model shown above.

The model has been discretized with the finite element method, considering the tree as an Euler-Bernulli beam and the casing as a Mindlin-Reissner curved shell. The resulting system of differential equations is:

$$\mathbf{M}\mathbf{u}'' + (\mathbf{C} + \Omega\mathbf{G})\mathbf{u}' + \mathbf{K}\mathbf{u} = \mathbf{F}_{\mathbf{U}}(t) + \mathbf{F}_{\mathbf{R}}(\mathbf{u}, \mathbf{u}') \tag{4.1}$$

Rotor			**Bearing**		
Young Modulus	$2 \cdot 10^1 1$	N/m^2	Radial stiffness	$2 \cdot 10^5$	N/m
Density	$7.85 \cdot 10^3$	kg/m^3	Bending stiffness	0	N/m
Shaft length	0.6	m	Mass	0.02	kg
Distance bearings	0.4	m			
Diameter	0.01	m			

Casing			**Coupling**		
Young Modulus	$3.6 \cdot 10^{10}$	N/m^2	Radial stiffness	0	N/m
Density	$1.11 \cdot 10^3$	kg/m^3	Bending stiffness	0	N/m
Poisson's ratio	0.24				
Radio strain/energies	5/6				
Midplane diameter	0.134	m			
Length	0.4	m			
Thickness	0.003	m			

Foundation			**Damping**		
Support stiffness in x	$2 \cdot 10^6$	N/m	Damping ratio 43.1Hz	0.02	
Support stiffness in y	$5 \cdot 10^5$	N/m	Damping ratio 78.4Hz	0.0133	

Rub spindle			**Disk**		
Stiffness coefficient	$2 \cdot 10^5$	N/m	Mass	1.5	kg
Damping coefficient	$1 \cdot 10^2$	Ns/m	Thickness	0.025	m
Friction coefficient	0.3		Diameter	0.1	m
Rotor-casing clearances	$24 \cdot 10^{-6}$				
	$15 \cdot 10^{-6}$				

With:

$\mathbf{M}$	Mass Matrix
$\mathbf{C}$	Viscuous Damping Matrix
$\mathbf{G}$	Gyroscopic Matrix
$\mathbf{K}$	Stiffness Matrix
Ω	Rotational Speed
$\mathbf{u}$	MDOF Vector displacements
F_U	Rotor Unbalance Force
F_M	Shaft misalignment
F_R	Rub Force

Table 4.2: 2nd order SODE Elements

The vector F_R, which describes the action of the external force generated by

the rub, is obtained through the sequential equation:

$$F_R(\mathbf{u}, \mathbf{u}') = (k_R\, d_R + c_R\, d'_R) \cdot H(d_R - \epsilon) \tag{4.2}$$

d_R	Relative deformation between rotor and casing
H	Heavisize function
ϵ	Clearance between rotor and stator

Table 4.3: F_R Elements

For numerical integration over time the implicit method Newmark-β [48] has been chosen. This method calculates the speed and displacement of the $i+1$ step using an implicit finite-element expression.

4.2.2 Numerical Model Simulation

The numerical model is able to simulate the contact between stator and rotor. A rub condition between the rotor and an element attached to the casing affects a shaft node close to one of the disks and one of the casing nodes in the same radial plane, with $k_R = 2.25 \cdot 10^6 \ \ N \cdot m^{-1}$, $c_R = 0$ being no impact damping and $\mu = 0.2$ being steel-to-steel rub. During the simulation aimed at creating data for the machine learning software, the intensity of the contact increases linearly. The force of the Rub increases as the gap between the stator and the rotor decreases, from an initial value of $\epsilon = 50 \cdot 10^{-6} m$ to a final value of $\epsilon = 0 \cdot 10^{-6} m$, corresponding to the maximum contact intensity.

4.2.3 Organization and structure of the CSV numerical model's data achieved

The results of the experiments saved in CSV format. In the file each line corresponds to a single instance, which contains the acceleration data generated by 8 machine rotations. For each simulation there are 3 CSV files: two generated respectively by each of the two accelerometers, while a third one contains the correct classification of the data package (0 = "No Rub", 1 = "Rub").

4.3 Pre-processing of data for neural networks

4.3.1 Synchronous Resampling

An *Synchronous Resampling* is performed on the data signal to obtain a homogeneous number of samples for each turn of the rotor. To achieve this result we have to generate a Hall signal, in order to obtain the sample frequency, the synchronous resampling period and then the change of the domain assuming that the rotating speed is constant. The signal captured by the sensors, in the case below an accelerometer, is a time-dependent voltage signal (time-base plot), with a sample frequency of 5kHz.

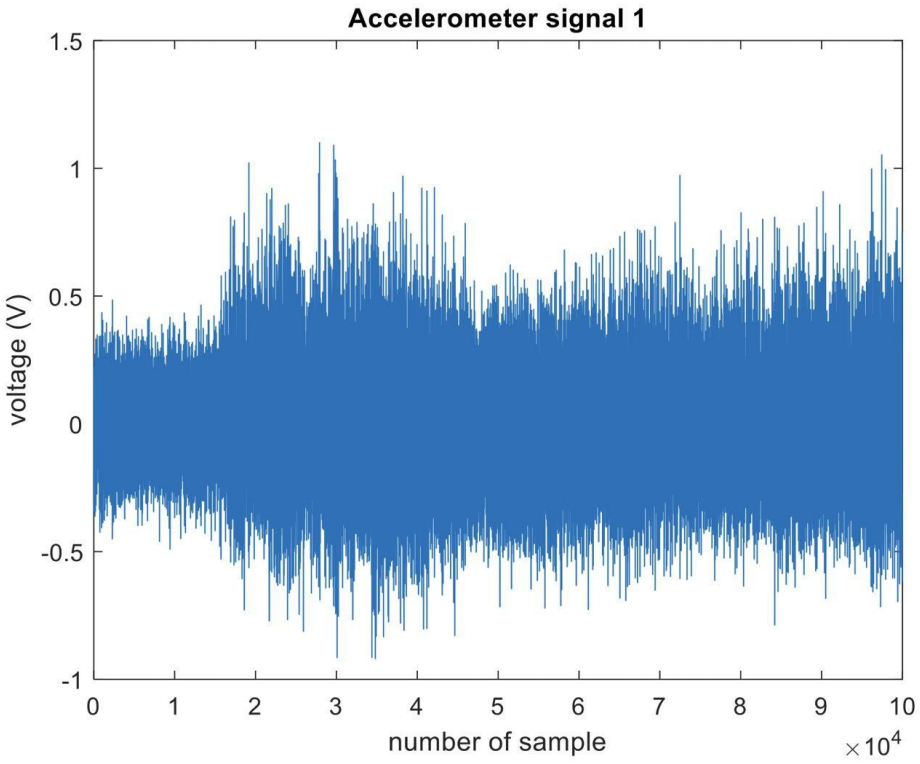


Figure 4.4: Matlab timebase plot of the accelerometer 1 signal of 60s registration

In order to generate a Hall type signal, a Keyphasor sensor is placed into the experimental engine to perform the synchronous resampling operation. The sensor will generate a output of 24 V when it detects a small landmark hole present on the shaft, while the signal will be 0 V for the rest of the round. So, the inverse of the time interval $\Delta t = t_2 - t_1$ between two revelations will be the number of

revolutions per second that the rotor performs.

$$T_s = \frac{2\pi \ radiants}{32 \ samples} \tag{4.3}$$

$$\begin{gathered} T'_s = \frac{t_2 - t_1}{32 \ samples} \\ 2\pi \cdot T'_s \approx \Delta\theta_{cycle} \end{gathered} \tag{4.4}$$

With T'_s corresponding to the period between two consecutive angularly equidistant measurements of two Keyphasor sensor pulses, while T_s is the angular period between two consecutive samples of synchronous data obtained. It is assumed that the speed remains constant within a single round.

With a Synchronous Resampling of 32 samples (16 harmonics) per instance, with a rotation speed of 96 HZ, the output signal will have a frequency of around 1500 Hz.

4.3.2 Feature Scaling

Machine learning algorithms do not work well when they are fed with different types of data with different scales. Having features on a similar scale can help the gradient descent converge more quickly towards the minimum, [43]. The difference in ranges of features will cause different step sizes for each feature. Hence, It is necessary to apply a *scaling feature to the database*. Two are the most common techniques to match the scales of different information sources: *min-max Scaling* and *Standardization* the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Min-max Scaling, also called *normalization*, consists of translating and scaling the numerical values of the data so that they are within a range from 0 to 1.

$$x'_i = \frac{x_i - min(X)}{Max(X) - min(X)} \tag{4.5}$$

With X the set of data of the same nature on which we want to make the min-max Scaling.

Standardization is a feature scaling process, where data is centered and then scaled according to the standard deviation. Unlike min-max scaling, standardization does not have on-board limit values, which can be a problem because many

neural networks expect input values between 0 and 1. However, the biggest advantage of standardization is that it is less affected by outlier data, which in normalization causes a rescaling of the whole dataset.

$$x'_i = \frac{x_i - \mu}{\sigma} \tag{4.6}$$

Where sigma is the standard deviation of the X dataset, while μ is the mean of the feature values.

4.3.3 PCA Dimensionality Reduction

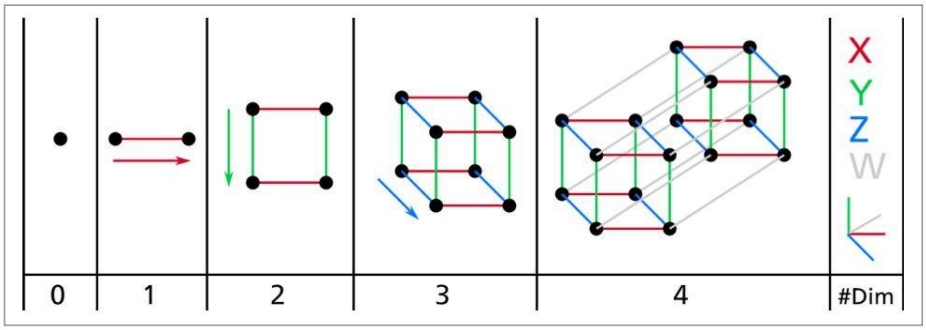


Figure 4.5: Dimensionality Reduction from tesseract to a point[34]

In many Machine learning problems each instance can be composed by a big amount of data features that, in addition to slowing down the learning algorithm, can also create difficulties to achieving the optimal solution. This problem is also called *curse of dimensionality*. To overcome this problem, there are many algorithms to greatly reduce the number of features of each data packet, all this correlated only with a reduced loss of information. Apart of speeding up training, the dimensionality reduction is also extremely useful for data visualization.

Principal Component Analysis (PCA) is the most popular dimensionality reduction algorithm. This mathematical method allows you to find a hyperplane, with a smaller dimension than the number of features of the dataset, that lies closest to the database, and project the data onto it.

The selection of the best subspace is essential to reduce the loss of information that this step brings with it. It is therefore necessary to choose the axis that maximizes the residual variance of the reduced database. The unit vector that defines the i^{th} axis is called the i^{th} *principal component* (PC). The direction of the axes that define the PCs are not stable against a perturbation of the data, but the plane they define will generally remain the same. To find the PCs of the dataset we

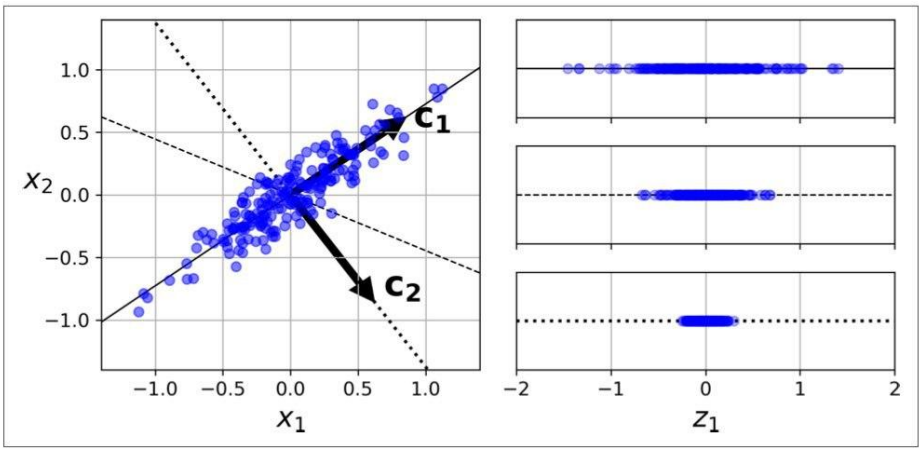


Figure 4.6: PCA reduction [34]

use a standard matrix factorization technique called *Singular Value Decomposition* (SVD), which decomposes the X database matrix into three matrices $U\,\Sigma\,V^T$, where V contains the main components of the hyperplan. The PCA needs the database to be centered around the origin: it is therefore recommended to perform the standardization first, in order to have the dataset ready for the reduction. In order to choose the best compromise between instance dimension size and information, it may be useful to represent the Variance according to the number of reduced features, as in the figure 4.7.

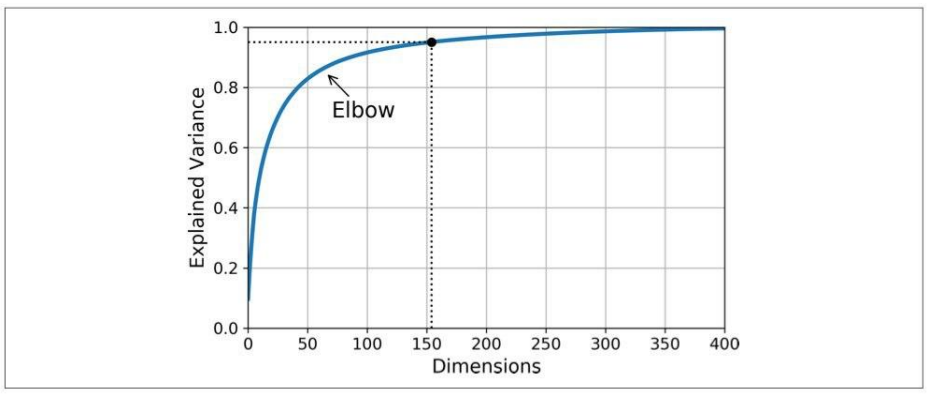


Figure 4.7: Dimensionality Reduction and Explained Variance [34]

The fraction of variance explained by a principal component is the ratio between the variance of that principal component and the total variance.

4.3.4 Mathematical transformations applied to the signal

The transformation is a mathematical operator, generally linear, operating on one function space on another function space. It must be an invertible function. Typically, this type of operator is used to simplify the resolution of a problem.

Let's imagine we want to solve a generic type A problem (e.g. an arithmetic calculation or a differential equation). In order to make the resolution easier to resolve, a transformation can be involved:

- Transform the problem A into B, which is easier to solve
- Solve the problem B
- Subsequently, with the use of the antitransformed, the solution of B is reconverted into the solution of A.

Discrete Fourier Transform (DFT)

The frequency spectrum is used to identify the frequency components (phase and amplitude) present in a complex vibration signal, which amplitude varies for every component of nX (subsynchronous and supersynchronous). When both the function and its Fourier transform are replaced with discretized counterparts, it is called the *Discrete Fourier Transformation* (DFT). A possible tool that to transfer signal information from the time domain to the frequency domain is the FFT (*Fast Fourier Transformation*), that is a very fast algorithm for computing the DFT. The vibration signal can contain the information about the running speed and multiples, line frequency electrical noise, gear mesh frequencies, gear defect frequencies, rolling element bearing frequencies, and vane and blade pass frequencies [37]. In the signal there are also data of excited rotor system natural frequency and also, at certain nX subsynchronous or supersynchronous, information about any malfunction of our interest (rub, flued-inducted instability, compressor rotating stall etc.).

The Fourier transform is an integral mathematical operator, and is a tool, similar to the Fourier series, to carry a signal from the time domain to the frequency domain. While the Fourier series needs the $x(t)$ signal to be periodic over time, the Fourier transform allows you to study any signal, even a non-periodic one.

If $x(t) : \mathbb{R} \to \mathbb{C}$ is a function.

If $\int_{-\infty}^{+\infty} x(t)e^{\mathbf{i}\omega t}\,dt$ exists for every $\omega \in \mathbb{R}$

hence the function $x(t)$ is **transformable** according to Fourier. In this case, the Fourier Transformation Function is:

$$F[x(t)](\omega) = X(\omega) := \int_{-\infty}^{+\infty} x(t)e^{\mathbf{i}\omega t}\, dt, \quad \omega \in \mathbb{R} \tag{4.7}$$

The fourier transform is applied in order to draw the *half spectrum plot*. Since we cannot install 2 proximity sensors at the supports in the aeroderivative gas turbine, we cannot obtain the *full spectrum plot*, which contains much more information about the vibration of the shaft.

The vibration signal is formed by a series of harmonics: the lowest frequency harmonic is called *Fundamental*, while the others have a multiple frequency of the fundamental (the second harmonic has a double frequency, the third triple, etc.). In a typical series, the amplitude decreases rapidly for the highest frequencies. The fundamental frequency of a rotating machine is usually 1X, but it is not excluded that it can also be a subsynchronous or supersynchronous.

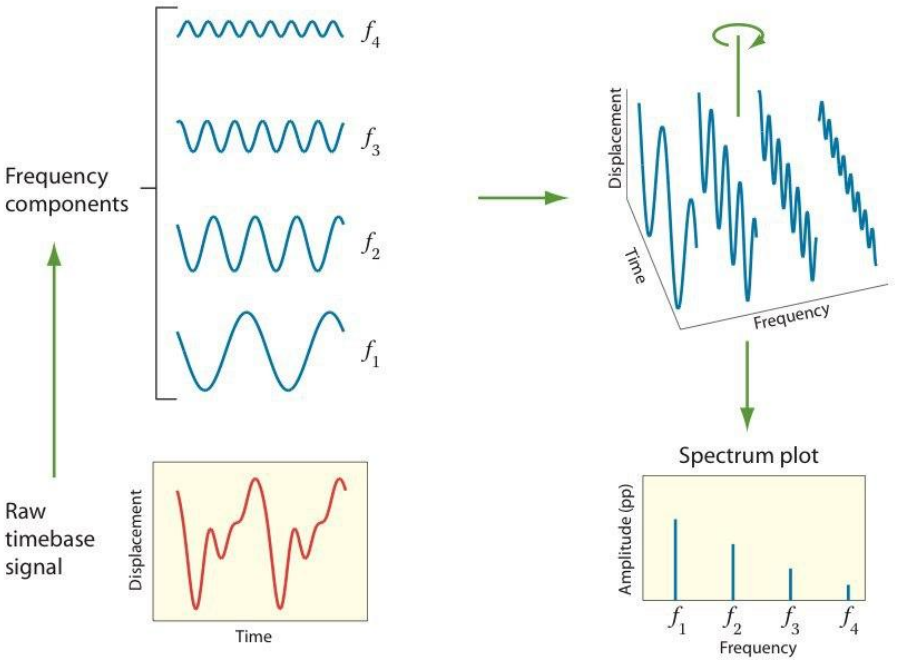


Figure 4.8: Time and Frequency domain display of a complex signal [34]

Let us consider a machine that works at a constant speed. The signal generated by the transducers is complex if it is simply represented with a *timebase plot* (in red). A filtered signal will consist of sine waves with different amplitudes and phases. Fourier allows us to calculate the sine wave *components* (frequency, amplitude and phase). By representing only, the amplitude and frequency of the components we will obtain a two-dimensional discontinuous plot with a series of vertical bands,

where the length represents the peak to peak amplitude 4.8. The phase for each signal is measured with respect to the trigger signal that starts the sampling process at time t_0.

Synchrosqueezed Wavelet transform (CWT)

As the Continuous Fourier Transformation, the CWT *Continuous Wavelet Transformation* is a integral correlation in the time domain of a function $x(t)$ with some basic function. While In Fourier the basic function is $e^{i\omega t}$, in Wavelet transform the basic function is the dilated and translated version of the Ψ function, also called *mother wavelet*

$$W(a,s) = \frac{1}{\sqrt{|s|}} \cdot \int_{-\infty}^{+\infty} x(t) \cdot \Psi^*(\frac{t-a}{s})\, dt \tag{4.8}$$

where a and s are, respectively, translation and scaling factors.

The wavelet transformation is another source of information for the neural network, only it was not possible to implement it in the Neural Network code of the program, due to the size of the generated database with this transformation. It will be later implemented in future versions of the program.

Chapter 5

Deep Learning Program and Results Analysis

5.1 Python's Tool for Deep Learning Neural Network

The Python programming language has been chosen for the creation of the neural network ML code. This is an interpreted, high-level, general-purpose programming language. Python is an excellent choice to deal with machine learning problems, thanks to very powerful packages present on it to solve this type of problem, as shown later. The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which lists some aphorisms such as:[49]

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

The Python distribution used is Anaconda. It is a free and open-source distribution of the Python language optimized for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

Spider has been chosen as anaconda's environment. Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming with the Python language. Spyder integrates with several prominent packages in the scientific Python stack. The packages used in the software are:

- *NumPy* for all the mathematical operations
- *Panda* for data extraction and management
- *TensorFlow's Keras API* for the construction of the neural network architecture
- *Sklearn* library for all the evaluation tools for the results and networks created

5.2 Data Organization

The key objective of the neural network code is the creation of a tool that allows to detect as soon as possible the malfunction of the *single-point partial rub* inside a rotating machine, to build a powerful preventive maintenance apparatus for aeroderivative gas turbines. Hence, the results of the prediction early in the problem, under the most difficult classification working conditions (when the intensity of the rub minimal and the signal disturbed by high background white noise) will be crucial for the performance evaluation of the various network architectures.

The program is a **Binary Classifier** that will evaluate, for each instance (8 rotations), whether or not it has the rub phenomenon.

There are two databases used for construction and evaluation of the neural network architecture: the first one is composed by data created with the finite element numerical model [4.1.1] and the second one from the experimental model data of the rotary machine [4.2]. Both databases have normal operating data (No Rub - Label=0) and rub malfunction data (Rub - Label=1). The program will be trained **only** with the data of the numerical model and then evaluate the classification performance on the experimental model data.

The simulation of the numerical model presents a rub that grows linearly up to a maximum value, and simultaneously a increasing white background noise. The added noise makes possible to obtain distinct synthetic vibration data with different simulations because, otherwise, they would be the same since the parameters that define the simulation do not change.

The data generated by the numerical model will be divided into test, trial and validation, according to the intensity of the rub.

The rub intensity level is calculated based on the absolute value of the average amplitude of the system oscillation $Am_{0_{Pi}}$, recorded by the Pi sensor in a working condition without rub and without background withe noise, where i = 1,2, ..,6. is the position index for each individual sensor.

$$Am_{0_{Pi}} = \frac{\sum_{j=1}^{N} \left| x_{j0}^{(Pi)} \right|}{N^{(Pi)}} \tag{5.1}$$

Where $x_{j0}^{(Pi)}$ is the acceleration value $[g]$ measured by the Pi sensor in the case **without Rub and White Noise**, and $N^{(Pi)}$ is the number of samples present in an instance. Hence, with the same procedure, we can obtain the average oscillation amplitude $Am_{i_{Pi}}$ of each instance i for both $P1$ and $P2$ sensors of the numerical model database.

$$Am_{i_{Pi}} = \frac{\sum_{j=1}^{N} \left| x_{j}^{(Pi)} \right|}{N^{(Pi)}} \tag{5.2}$$

Subsequently, the level of Rub $LoR_{i_{Pi}}$ can thus be calculated as:

$$LoR_{i_{Pi}} = \frac{Am_{i_{Pi}}}{Am_{0_{Pi}}} \tag{5.3}$$

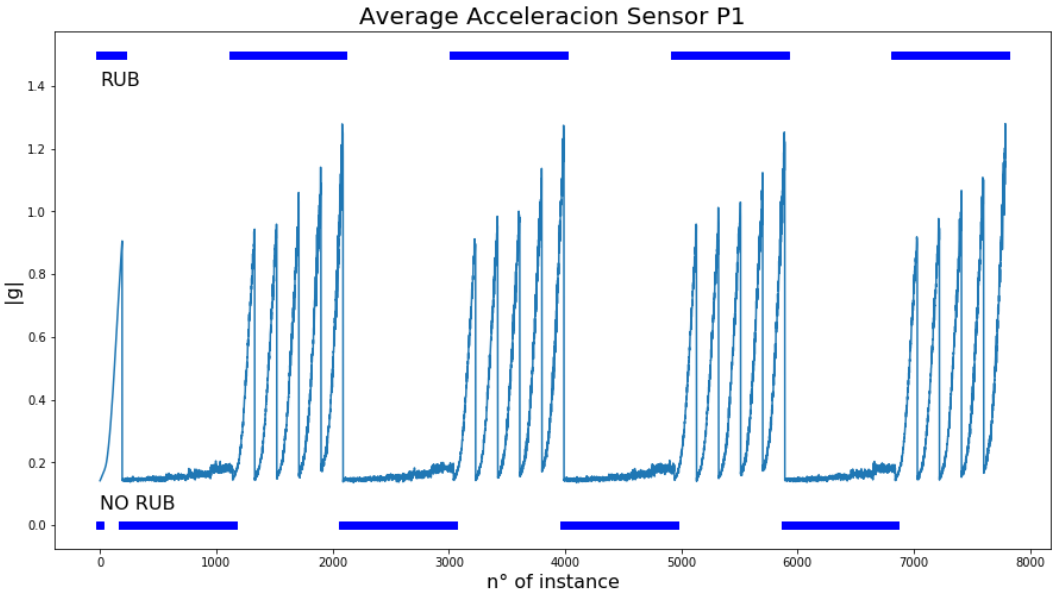


Figure 5.1: Average Acceleration Sensor P1 $Am_{i_{P1}}$

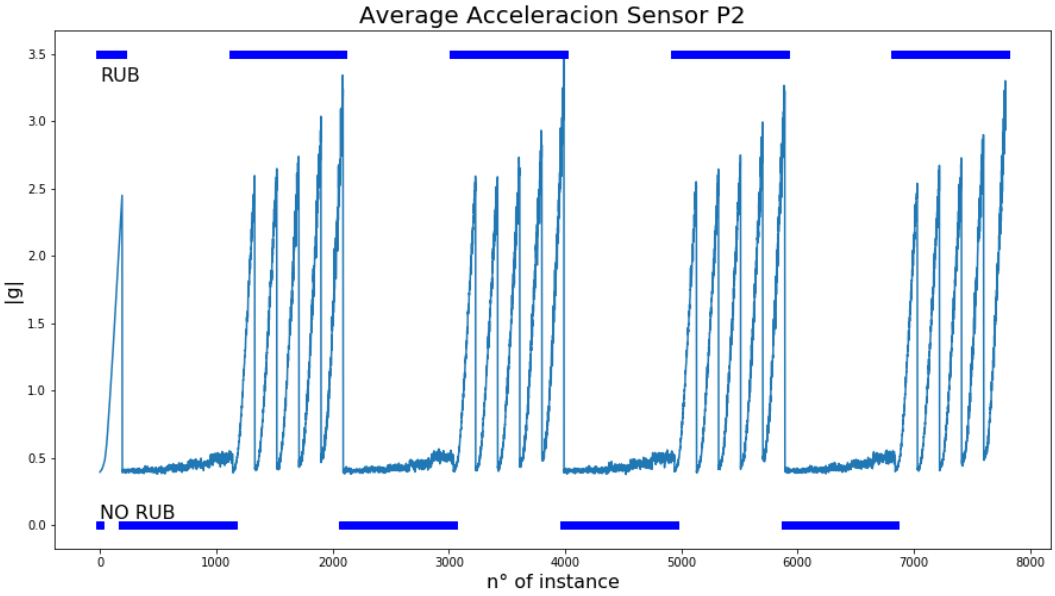


Figure 5.2: Average Acceleration Sensor P2 $Am_{i_{P2}}$

The following graphs (5.1 and 5.2) show the average values of the accelerations simulated at sensors P1 and P2. As we can see, the two sensors detect different acceleration intensities because they are located in two different points of the external casing. To use both sources of information it will therefore be necessary to standardize the data. The blue line indicates the label information, whether that instance refers to a case with rub (1) or without rub (0).

The database is built by attaching the results of different simulations: in the first simulation there is no white background noise and the experiment is performed with an increasing rub level. Then the other four simulations are composed by a first part without rub and with increasing white noise, and the other section with increasing rub with the same growth scale of the first simulation without rub, and variable white noise. These four simulations are identical and are differentiated by the white background noise applied to the signal.

For training the net, only the first three sections with withe noise will be used, while the last experiment will be required to assess at what intensity the code starts to detect the rub, keeping this data hidden from the algorithm.

Subsequently, after setting the two *tresholds*, one for data with rub and another for data without rub, we can split the data for training and performance calculation. Data **Without Rub** with a vibration intensity lower than the corresponding threshold and data **With Rub** above the corresponding threshold will form the training dataset. The remaining data (without rub with high background noise, and with rub with a low contact intensity) will be used for the prediction.

The idea behind this choice is that, by training the program with rub and no rub data easily distinguishable for us, the neural network can find other patterns, different from the ones we used (in our case intensity of oscillation). Thanks to this result the software will be able to execute a correct classification also when the difference is not very noticeable for us. Therefore, the prediction samples will be used to evaluate and compare the performance of different neural network architectures, to choose the best one.

The values of thresholds chosen are:

$Treshold_{No\ Rub}$	$120\%\, of\ Am_{0_{Pi}}$
$Treshold_{Rub}$	$150\%\, of\ Am_{0_{Pi}}$

Table 5.1: Thresholds for Rub and No Rub Data division

The overall size of the database composed of the data provided by the Numerical Model is: 7791 instances with 256 elements each line (32 samples for each shaft rotation cycle). Of these, 3990 instances are of *Rub* and 3801 of No Rub. It is important to verify, when generating a subset of data, that it has a sufficiently representative sample of elements of each type.

With the previously chosen thresholds, we get a *Training Database* of 4497 instances (2262 of Rub, 2235 of No Rub) and a database for the *Prediction Database* of 1484 (778 of Rub, 706 of No Rub). Only test data with white noise shall be used and noise-free data (the first simulation) shall be excluded. The *Training Database* is the sample of data used to fit the model's hyperparameters (weights and biases of the Neural Network). The model sees and learns from this data.

In turn, the training database will be divided into three parts: *Test, Train, and Validation*. All needed for the training and tuning of hyperparameters of the neural network. The database is divided into 70% Train, 20% Test, 10% Validation, since this is a widely used size ratio [34]. The subdivision is random, and it is important that each subset presents a sufficient number of data for each of the two categories.

	%	Total Data	Rub Instances	No Rub Instances
Total Training Data	100	4497	2262	2235
Train Dataset	70	3147	1583	1564
Test Dataset	20	945	467	478
Validation Dataset	10	405	212	193

Table 5.2: Train, Test and Validation Data Distribution

The Fourier Transformation generates complex numbers and, due to the fact that an imaginary number cannot be normalized, the DFT data has to be divided into real and imaginary parts, doubling the number of neurons needed for DFT and increasing the size of the first layer of the network accordingly.

Since the input signal has 256 samples, the Fourier transformation generates 256 harmonics + 1 constants; but these 256 harmonics have 128 symmetrical elements that do not apport any other additional information to the code. To avoid to unnecessarily increasing the size of the first layer, only the first 128 harmonics are considered.

Once the input layer has been built with the appropriate data sources (acceleration and/or DFT), a PCA reduction has been applied in order to reduce the size of the input layer and, consequently, of the entire network, lowering the total number of variables. The resulting size of the reduced PCA samples are different and the dimension is shown later in the description of each model type [5.5].

5.3 Data analysis

A preliminary analysis of the data is essential to the development of a Machine Learning Code to better understand and analyze the information contained in the database.

Let's represent the complete set of all data to see how it behaves between failure and no failure.

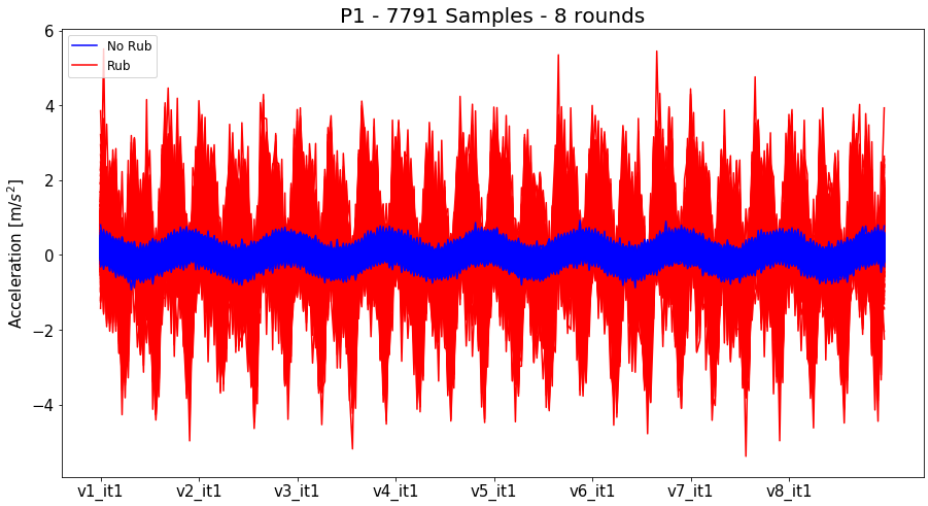


Figure 5.3: Acceleration signal of P1 containing 8 revolutions, with and without rub

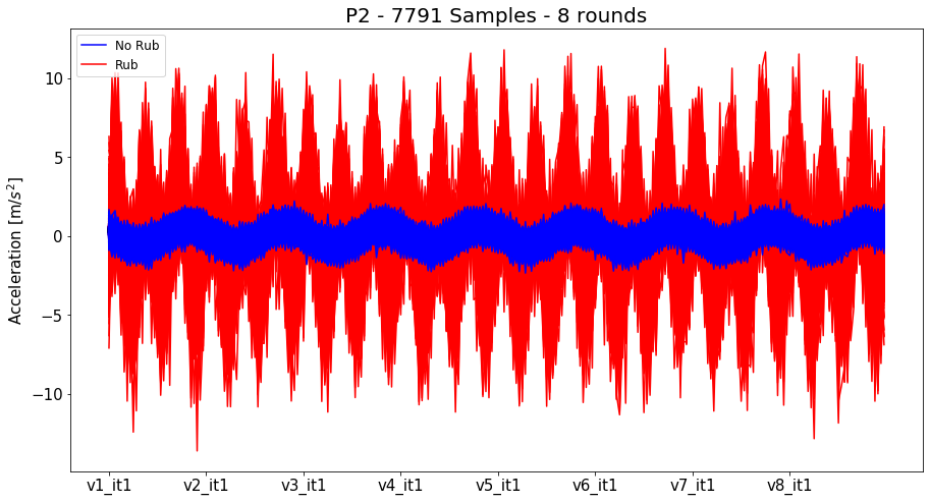


Figure 5.4: Acceleration signal of P2 compared to 8 revolutions, with and without rub

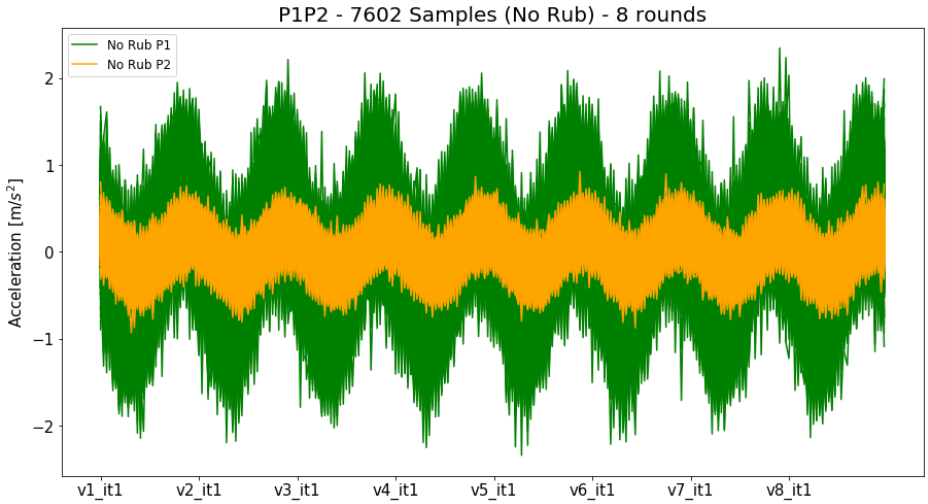


Figure 5.5: Acceleration signal of P1 compared with the P2 signal, without rub

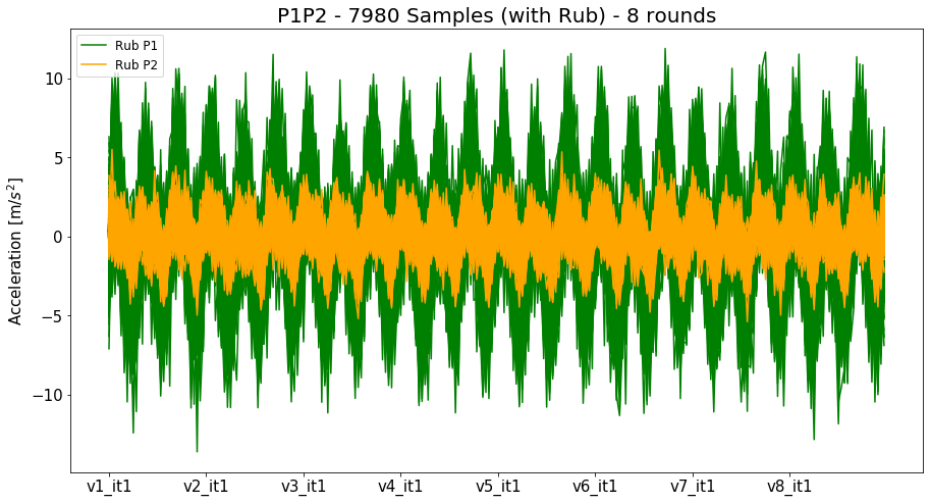


Figure 5.6: Acceleration signal of P1 compared with the P2 signal, with rub

From this representation we can notice that:

- The vibration frequency, in the case without rub, is 1X, with one precession complete rotation for each turn of the machine. Coherently since the exciting force is centrifugal and acts with a frequency equal to the rotation of the shaft.
- The precession vibration, in the case with rub, has a higher frequency, between 2X and 3X.
- From the graph comparing the signals of the two sensors P1 and P2 we can notice that the information contained in the two sensor signals are not

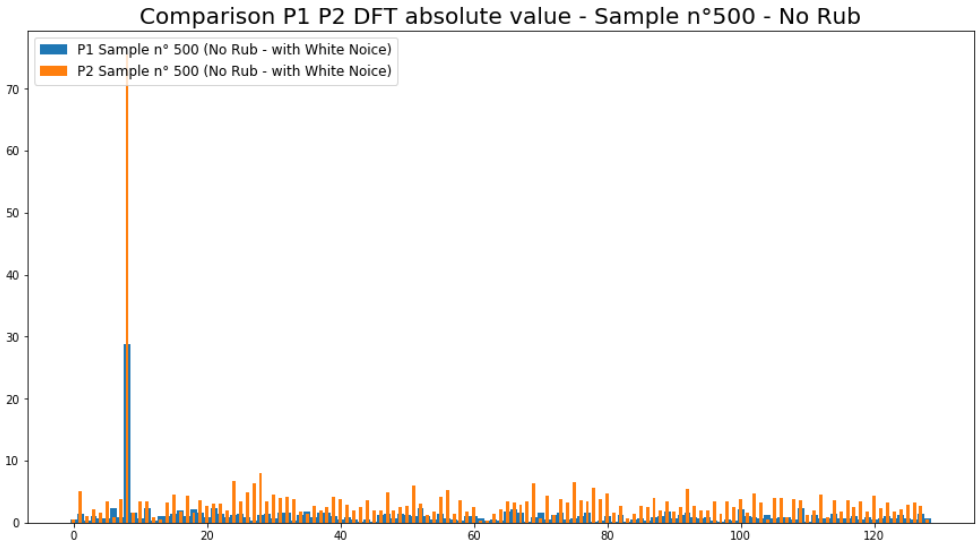


Figure 5.7: DFT transformation of the P1 signal compared with the P2 DFT signal, without rub

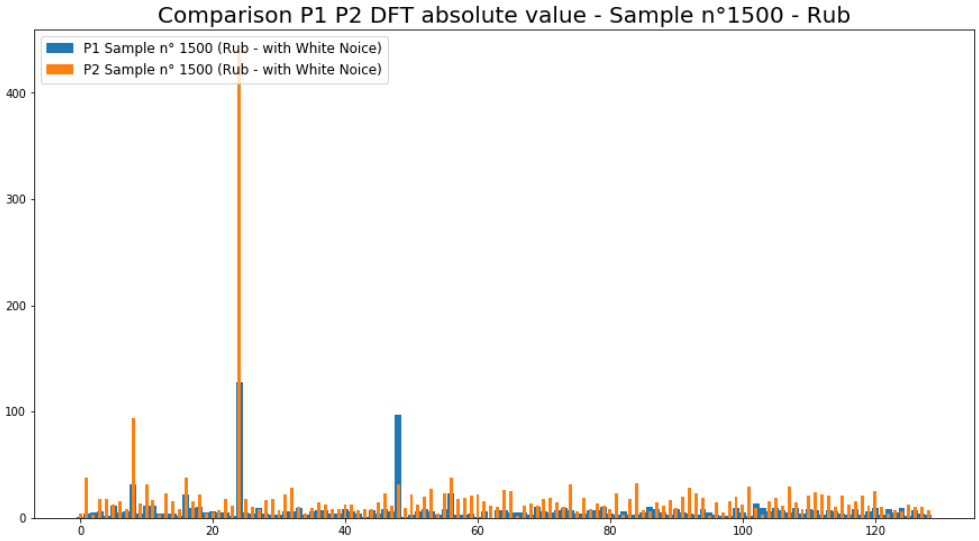


Figure 5.8: DFT transformation of the P1 signal compared with the P2 DFT signal, with rub

superimposable, and both should be used at the same time to generalize the problem. Especially the harmonic information contained into the DFT transformation present different values at the same frequencies.

5.4 Pipeline of the program

To build the best possible neural network software, three different approach of data elaboration to feed the code have been tried:

- Time Domain Acceleration data of P1 and P2
- Frequency Domain Acceleration signal of P1 and P2
- Acceleration and DFT transformation of P1 and P2

For each one of that three solutions many net architectures have been tested to optimize the program to the size of the input data. For future works Wavelet coefficients should be considered as input data and then analyze the prediction performances of the model. That solution required an "online learning approach" for the higher dimension of the Wavelet signal.

Below there is the pipeline of operations performed by the python program

1. Import modules and packages (*keras, sklearn, pickle, pandas, numpy, scipy*)
2. Uploading all data in the database
   (a) Import Sensor Label data
   (b) Import Acceleration data
   (c) Creation and saving DFT data
3. Representation and data Analysis
   (a) Representation the total P1 Failure and No Failure Acceleration Signal
   (b) Representation the total P2 Failure and No Failure Acceleration Signal
   (c) Representation the comparison of the total P1 and P2 Failure Acceleration Signal
   (d) Representation the comparison of the total P1 and P2 No Failure Acceleration Signal
   (e) Representation the comparison of some samples of P1 and P2 No Failure and Failure DFT Samples
4. Dataset Organization and Division
   (a) **Measurement of the level of rub** of all data
   (b) Saving training and prediction data sample positions based on the level of the rub

5. Preprocessing of data
    (a) Dividing DFT data into absolute value and phase
    (b) Data Standardization
6. Creation of the dataset for training
    (a) Creation of the dataset training matrix
    (b) PCA Reduction fit on the training matrix data
    (c) Application of the PCA Reduction on each sample
    (d) Division into train, test and validation dataset
7. Deep Neural Network model architecture design
    (a) **Deep Learning Neural Network Structure**
    (b) Model Compilation (Optimizer, loss and metrics)
    (c) **Training of the model**
    (d) Learning graph of the training
    (e) Model Save
8. Prediction Analysis
    (a) Creation of the dataset prediction matrix
    (b) Application of the same PCA Reduction of training data on the prediction matrix
    (c) **Prediction execution** with the Model Saved
    (d) Threshold automatic optimization
    (e) Plot the comparison of the prediction result score with the labels
    (f) Evaluation of the classification with the **Confusion Matrix**
9. Final Model Evaluation
    (a) Prediction VS Recall plot
    (b) **Detection Threshold Limit Analysis**
    (c) **Prediction on Real Data with the best model of each achieved with each approach**

5.5 Analysis of different Models

5.5.1 Time Domain Acceleration Model

The model is powered only with accelerometer [g] data, without the application of any transformations to the signal. All the layers that composed the network are “Dense" type, which means that the order of the input neuron does not affect the network, as long as the sequence does not vary with different samples. Each sample is composed of 512 data (256 of P1 + 256 of P2).

To evaluate the performance of the network during training, *accuracy* metrics [5.4] were chosen. This parameter consists of the ratio between all correctly classified samples (TP+TN) and the total number of samples in the database.

$$Accuracy = \frac{TP + TN}{Total\ number\ of\ samples} \tag{5.4}$$

The cost function chosen is the *Mean Square Error* [5.5], which is compatible with the learning technique based on the *Stochastic Gradient Descendent (sgd).*

$$MSA = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{n} \tag{5.5}$$

Optimizer	***Stochastic Gradient Descendent (sgd)***
Loss Function	***Mean Squared Error***
Metrics	***Accuracy***

Table 5.3: Model Compiler General Setting

The applied PCA transformation presents an explained variance of 0.97, reducing the sample size to 386 data, with a consequent 25% reduction of the number of neurons on the first layer of the net. This allows to reduce the number of variables (weights and bias) present in the network, reducing the computational burden to train the algorithm. To choose the value of the explained variance it is appropriate to represent the graph 5.9, where it is possible to observe the relationship between the value of the variance and the result dimension of the PCA sample.

In the following graph 5.9 we can observe that, up to a size of 350, corresponds approximately to an E.V. of 95 %, we do not lose a significant amount of information while, for smaller sizes of the PCA sample, the slope of the curve increases with a consequent greater loss.

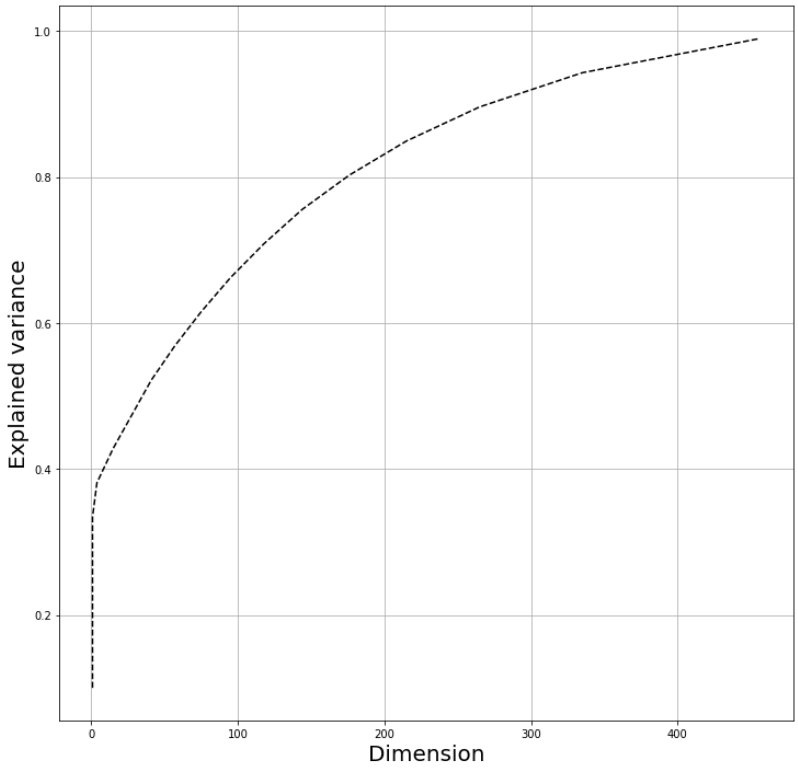


Figure 5.9: Variance Curve PCA reduction- "Time Domain Acceleration"

Model 1 "Time Domain Acceleration"

Network Architecture

The architecture of the Model 1 "Time Domain Acceleration" neural network presents the "sigmoid" function as the activation function of the last layer. Sigmoid and Tanh are also continuous and derivable functions, which makes them perfect for the network learning process. The sigmoid is a suitable solution for the classification task because it saturates for both positive and negative values. The intermediate layers have a reduced number of neurons compared to the size of the input size, and "tanh" as an activation function. The number of epochs has been chosen to guarantee network training, avoiding the problem of overfitting.

Even though the data are homogeneous, each sample has been standardized in all 3 "Time Domain Acceleration" models in order to make it possible to use later also other sources of information.

The dropout rate value selected is 50%, which is a value widely used in neural networks to prevent the problem of overfitting.

	N° Neurons	Activation	Dropout Rate
Layer Input	386	X	0.5
Layer 1	32	tanh	0.5
Layer 2	32	tanh	0.5
Layer 3	32	tanh	0.5
Layer Output	1	sigmoid	X
Number of Epochs	10		
Batch Size	20		
Total Parameters	18561		

Table 5.4: Network Architecture - "Time Domain Acceleration" - Model 1

Training Results

The learning curve 5.10 is useful to assess whether the network is encountering any problems during the training. For example, if the validation curves are not close and do not follow the training curves, it may be that the network is overfitting the training data.

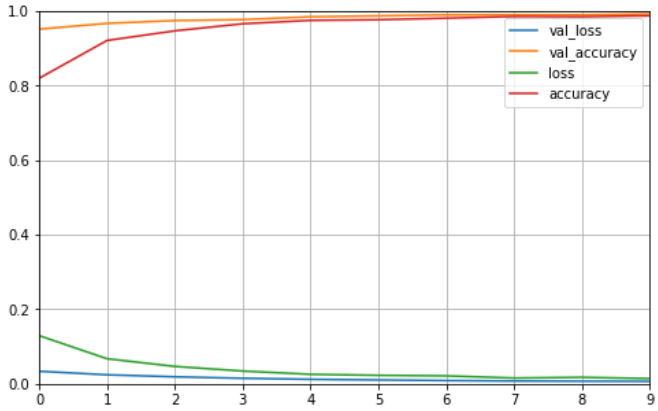


Figure 5.10: Learning Curves - "Time Domain Acceleration" - Model 1

By observing that the results show good accuracy and loss values (over 99% for the accuracy and below the 1% for the loss function), the network has been fully trained.

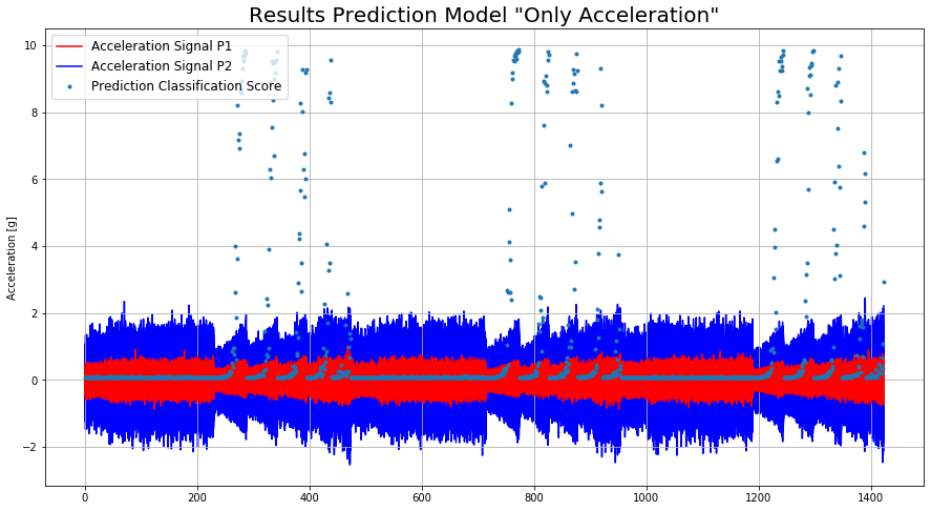
	Loss	Accuracy
Train	0.0142	0.9876
Validation	0.0066	0.9924
Test	0.0054	0.9924

Table 5.5: Training Results - "Time Domain Acceleration" - Model 1

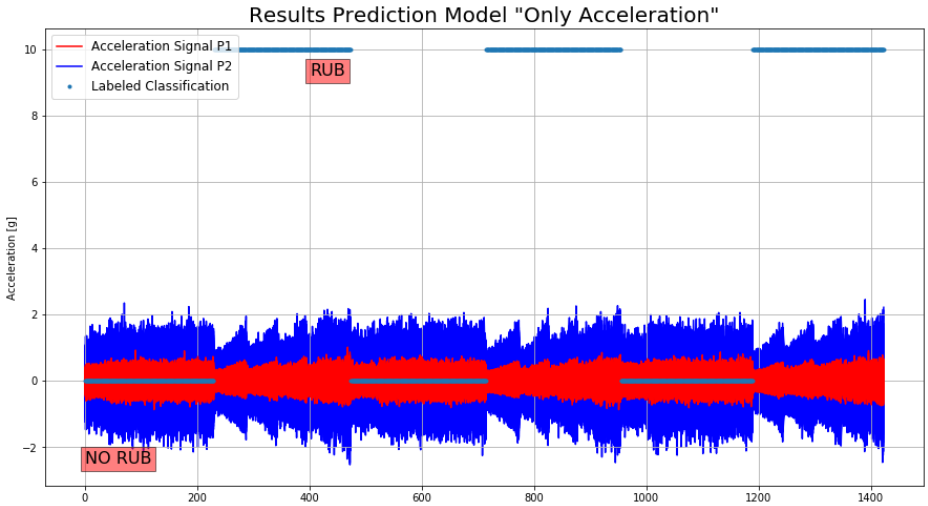
Prediction Results

We can observe in the next two graphs a comparison between the prediction results and the labeled data samples. Let us remember that the prediction database is composed of samples with less clearly distinguishable data ("Rub" instances with a low level of rub and "No Rub" instances with high white noise), in order to evaluate the performance of the model in the most difficult conditions. As we can notice in the plot 5.11, the average intensity of the acceleration recorded by the two sensors is similar for both "No Rub" and "Rub" samples.

The label of the samples is represented with a light blue line (the upper one for samples with Rub and the lower one for samples without Rub), while the prediction scores are light blue points ranging from 0 to 10. It is clearly distinguishable that, beyond a certain rub intensity oscillation, this deep neural network starts to notice the malfunction occurring into the rotating machine.



(a) Prediction Results



(b) Labeled Classifications

Figure 5.11: Prediction and Labeled Classification comparison - "Time Domain Acceleration" - Model 1

In the table below are the results of the classification matrix of this model. With the following attempts we will try to improve these values, modifying the network architecture or acting on the input data. We can already see from the first model that a deep neural network is capable to detect the problem of the rub, showing that this is a feasible option for the maintenance of an areoderivative gas turbine.

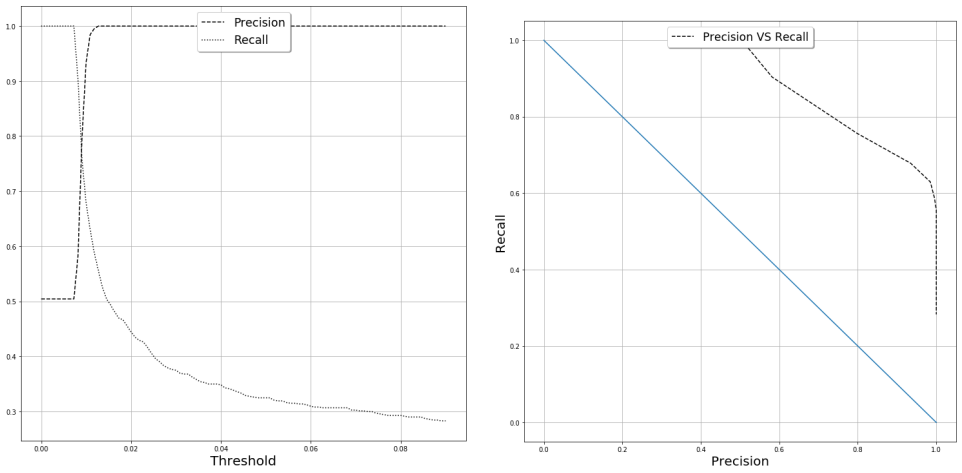
	Predicted NO Rub	Predicted Rub
Label NO Rub	562	144
Label Rub	132	586

Precision	78.21	%
Recall	78.97	%
Accuracy	78.30	%
Especificity	77.62	%

Table 5.6: Confusion Matrix and Results - "Time Domain Acceleration" - Model 1

Model Evaluation

We can understand how sensitive Precision and Recall are to the variation of the threshold observing the plot below 5.12. We will see how each architecture presents different sensitivity and optimal value of threshold that maximize both precision and recall. Accordingly we will evaluate this aspect for each model solution.



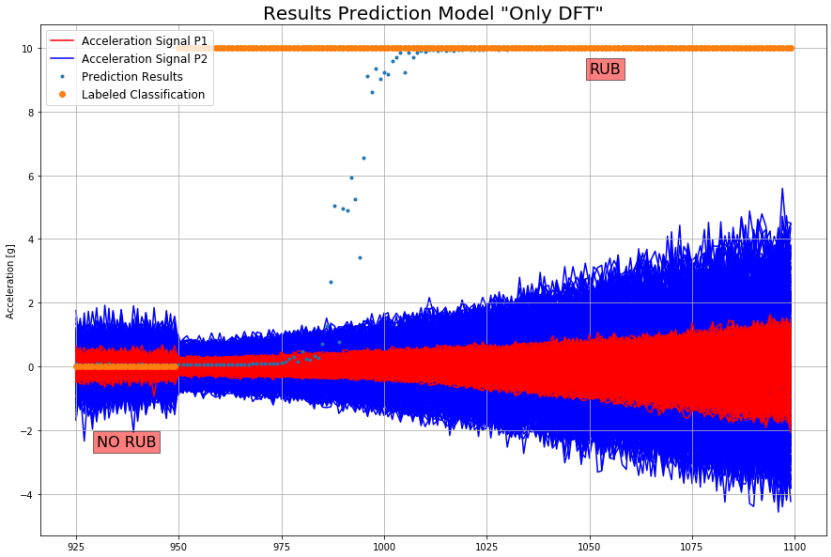
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

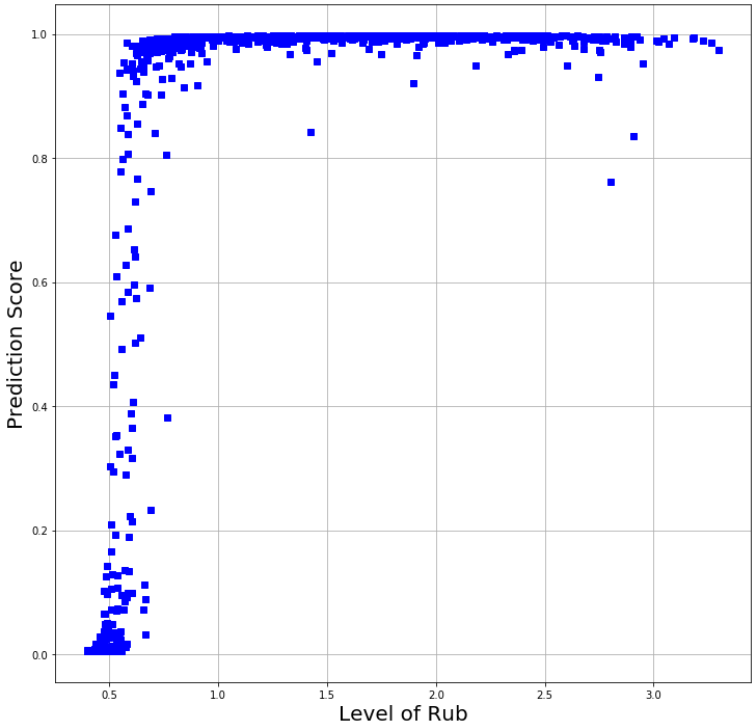
Figure 5.12: Relation between Threshold/Precision/Recall - "Time Domain Acceleration" - Model 1

To better understand when the network starts to see the rub malfunction, the picture below 5.13 (a) is a zoom on the acceleration signal prediction values with both results and label information. We can therefore note that, when the rub starts, the network is not immediately able to detect it. Subsequently, as the intensity increases beyond a threshold value, the network is able to detect it. In the second plot 5.13 (b) the prediction score is correlated to the average oscillation

intensity.



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.13: Level of Rub Prediction Analysis - "Time Domain Acceleration" - Model 1

Model 2 "Time Domain Acceleration"

Network Architecture

The general infrastructure of the model is the same as the previous one, only the number of neurons of the hidden layers has increased. We want to see if, having more parameters, the network is able to better generalize the data by extracting more information.

	N° Neurons	Activation	Dropout Rate
Layer Input	386	X	0.5
Layer 1	300	tanh	0.5
Layer 2	300	tanh	0.5
Layer 3	300	tanh	0.5
Layer Output	1	sigmoid	X

Number of Epochs	20
Batch Size	20
Total Parameters	142.001

Table 5.7: Confusion Matrix and Results - "Time Domain Acceleration" - Model 2

Training Results

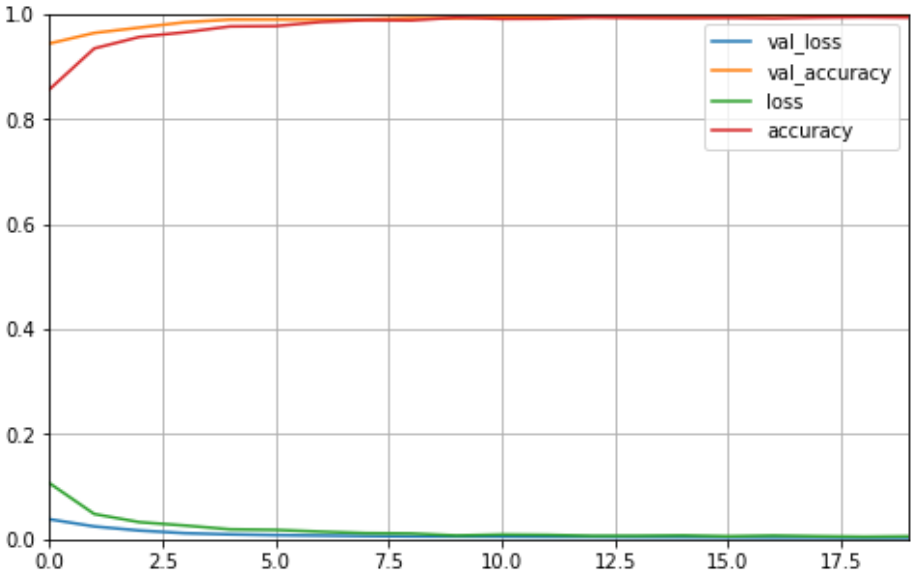
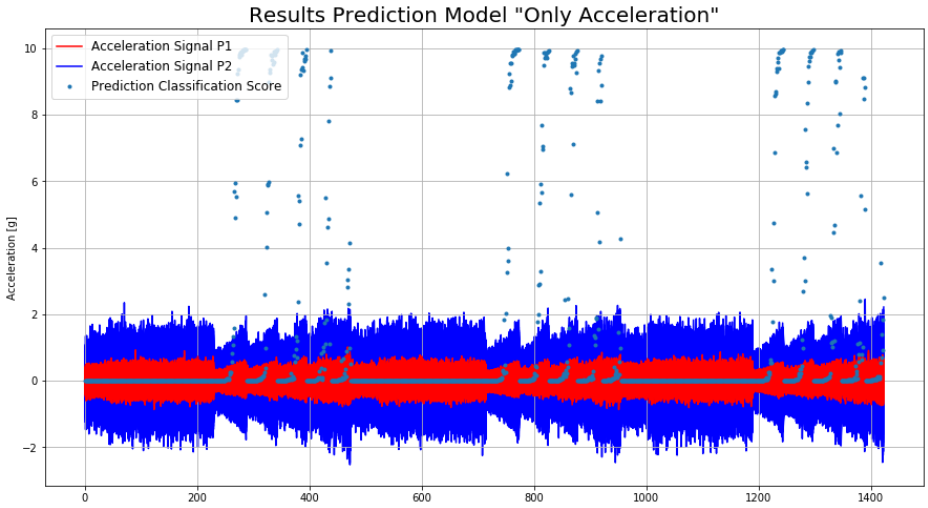


Figure 5.14: Learning Curves - "Time Domain Acceleration" - Model 2

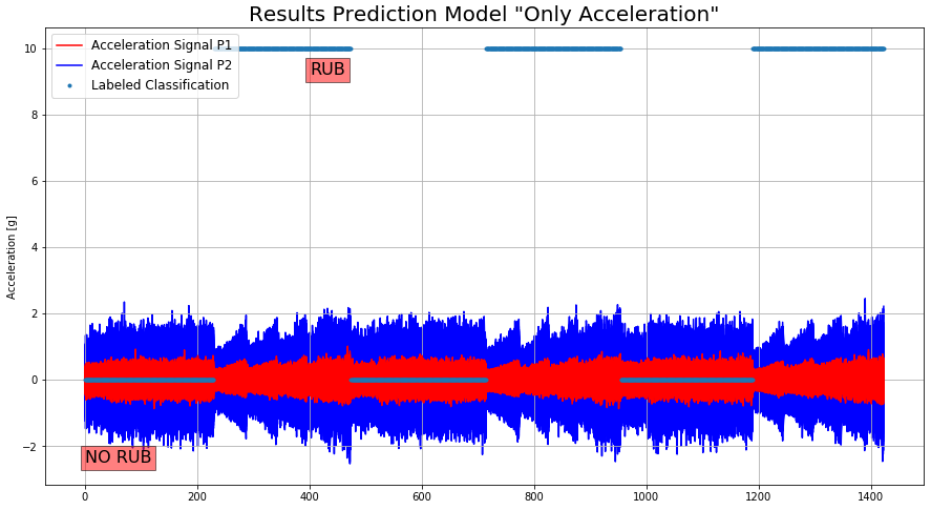
	Loss	Accuracy
Train	0.0036	0.9964
Validation	0.0025	0.9975
Test	0.0022	0.9967

Table 5.8: Training Results - "Time Domain Acceleration" - Model 2

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

Figure 5.15: Prediction and Labeled Classification comparison - "Time Domain Acceleration" - Model 2

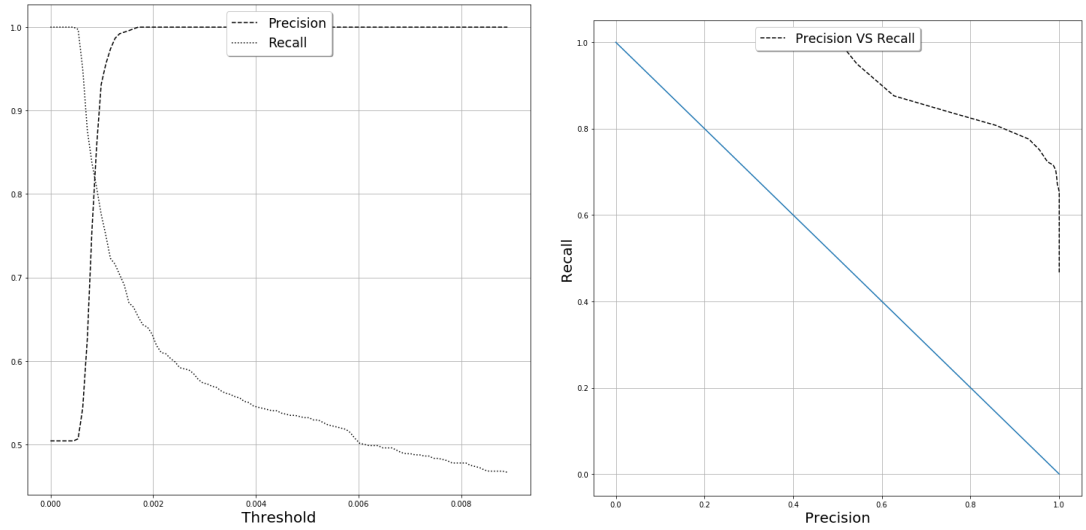
	Predicted NO Rub	Predicted Rub
Label NO Rub	598	108
Label Rub	116	602

Precision	84.78	%
Recall	83.84	%
Accuracy	84.27	%
Especificity	84.70	%

Table 5.9: Confusion Matrix and Results - "Time Domain Acceleration" - Model 2

Looking at the results 5.16, we can see an improvement by a few percentage points, compared to the previous model. Hence, the model 1 does not present enough parameters to describe the model.

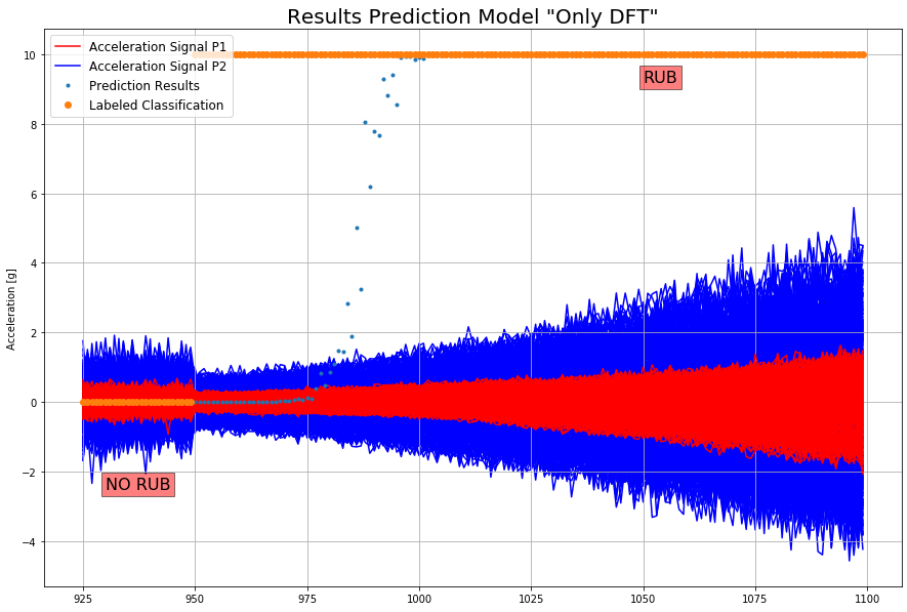
Model Evaluation



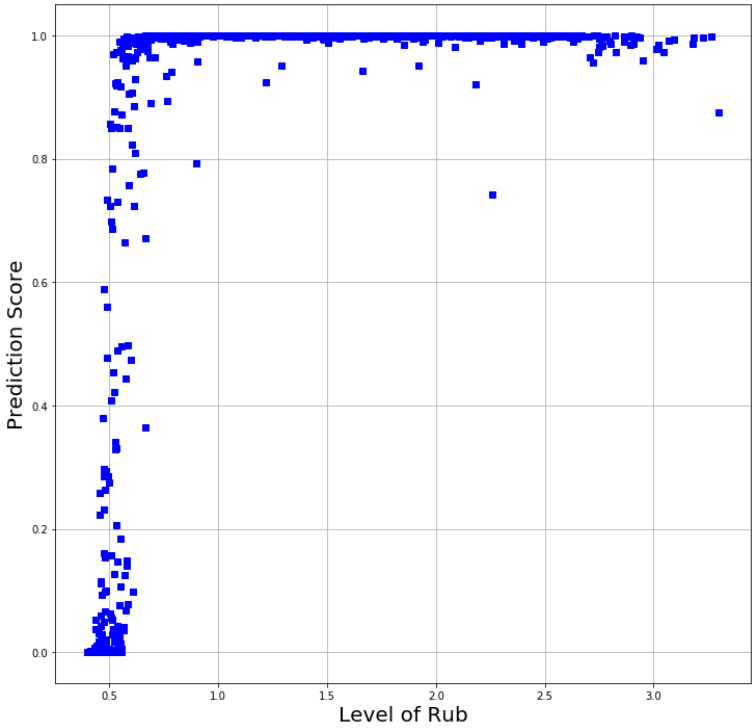
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.16: Relation between Threshold/Precision/Recall - "Time Domain Acceleration" - Model 2



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.17: Level of Rub Prediction Analysis - "Time Domain Acceleration" - Model 2

Model 3 "Time Domain Acceleration"

Network Architecture

In this model 3 we start from the architecture chosen for model 2, only that we use only the "sigmoid" function as activation function, even for the hidden layers.

	N° Neurons	Activation	Dropout Rate
Layer Input	386	X	0.5
Layer 1	300	sigmoid	0.5
Layer 2	300	sigmoid	0.5
Layer 3	300	sigmoid	0.5
Layer Output	1	sigmoid	X

Number of Epochs	20
Batch Size	20
Total Parameters	142.001

Table 5.10: Confusion Matrix and Results - "Time Domain Acceleration" - Model 3

Training Results

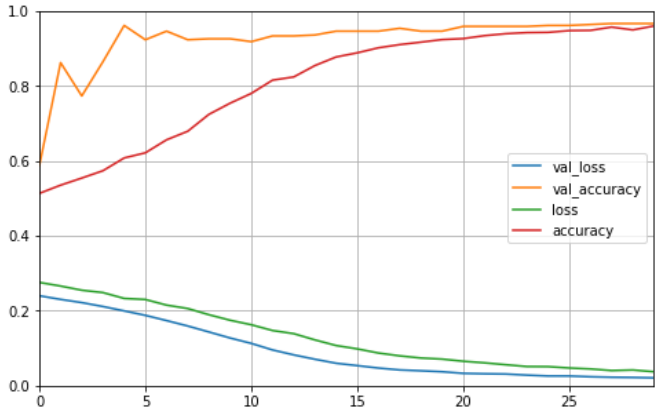


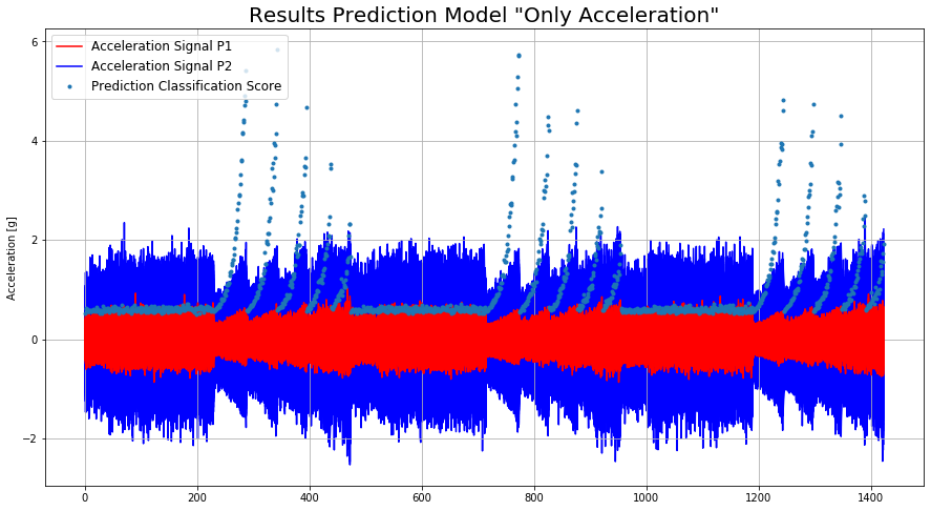
Figure 5.18: Learning Curves - "Time Domain Acceleration" - Model 3

From the learning curves we can clearly observe that this solution leads to less hanging learning curves, as the possibility of saturation can lead to the inactivation of neurons, slowing down the Stochastic Gradient Descendent process. On the other hand neuron saturation reduces the model's sensitivity to small variations in input data,

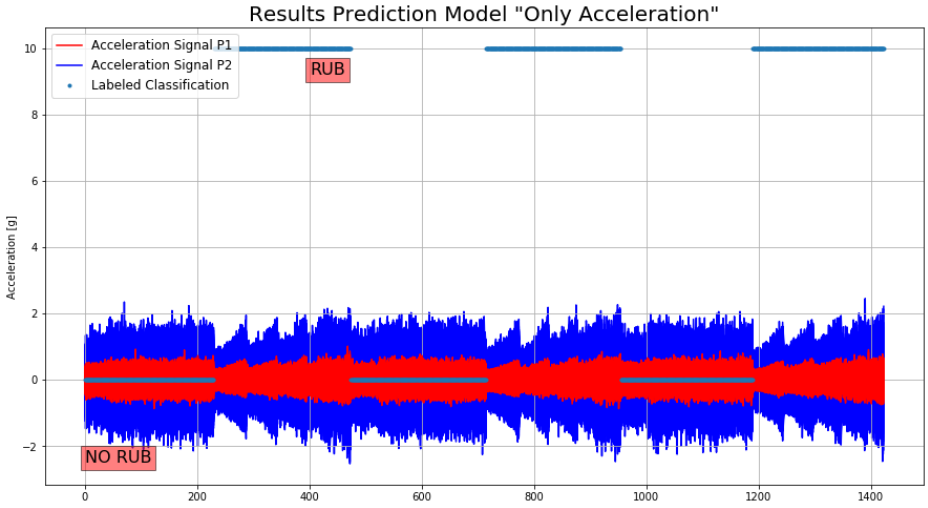
	Loss	Accuracy
Train	0.0372	0.9604
Validation	0.0209	0.9669
Test	0.0193	0.9760

Table 5.11: Training Results - "Time Domain Acceleration" - Model 3

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

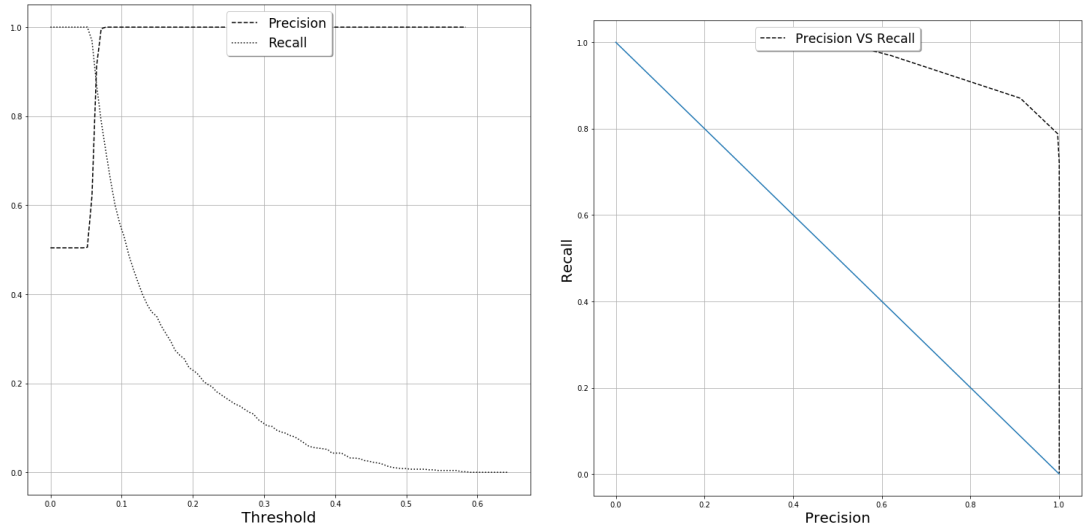
Figure 5.19: Prediction and Labeled Classification comparison - "Time Domain Acceleration" - Model 3

	Predicted NO Rub	Predicted Rub
Label NO Rub	615	91
Label Rub	84	634

Precision	87.44	%
Recall	88.30	%
Accuracy	87.71	%
Especificity	87.11	%

Table 5.12: Confusion Matrix and Results - "Time Domain Acceleration" - Model 3

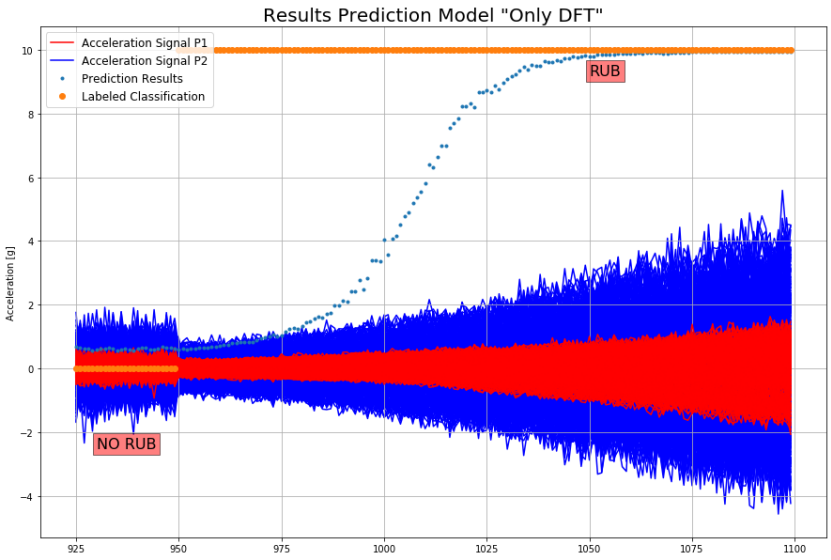
Model Evaluation



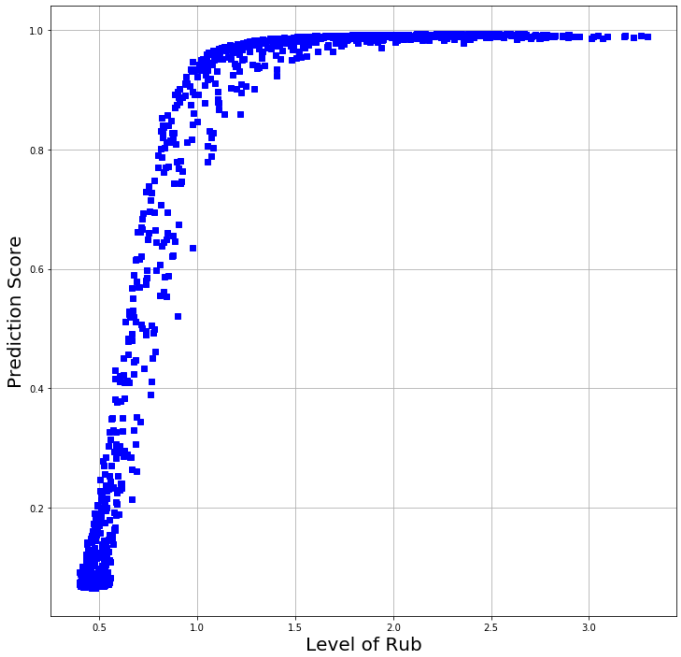
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.20: Relation between Threshold/Precision/Recall - "Time Domain Acceleration" - Model 3



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.21: Level of Rub Prediction Analysis - "Time Domain Acceleration" - Model 3

The application of the sigmoid activation function allows the creation of a program that gradually detects the important patters of the signal, resulting in a smoother prediction curve 5.21.

5.5.2 Frequency Domain Acceleration Model

The following models have been developed using only the accelerometer signal processed with the Discrete Fourier Transform (DFT). All the layers that composed the network are “Dense" type. Each sample is composed of 516 data (129 for the module of P1, 129 for the phase of P1, and the same for P2). The order of the array that composed each sample is $P2\,Phase + P1\,Phase + P2\,module + P1\,module$. It is important to maintain the order of information in the various databases prepared to feed the network.

Optimizer	***Stochastic Gradient Descendent (sgd)***
Loss Function	***Mean Squared Error***
Metrics	***Accuracy***

Table 5.13: Model Compiler General Setting

The applied PCA transformation presents an explained variance of 0.99, reducing the sample size to 439 data. I this case a slightly higher of E.V. value was chosen because the results were more affected by the loss of information. The following models also differ in the standardization techniques used, which have given diverse results.

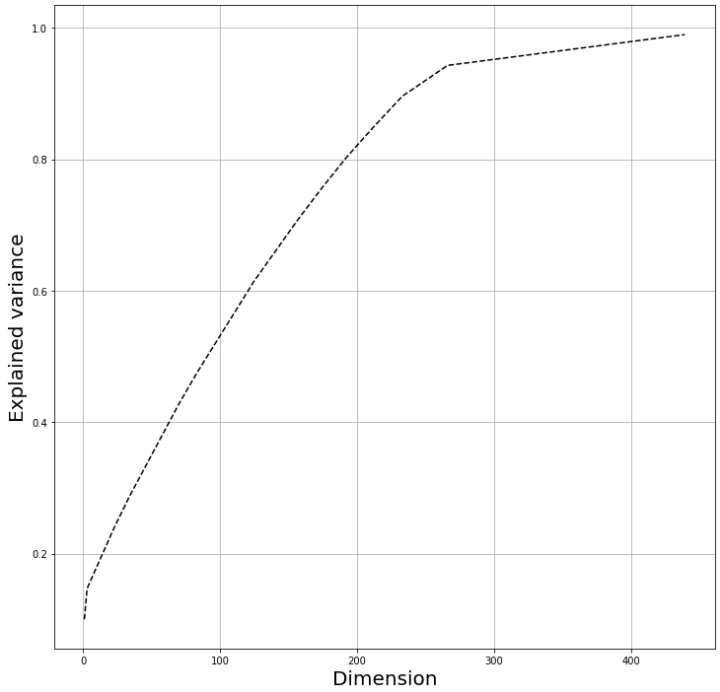


Figure 5.22: Variance Curve PCA reduction- "Frequency Domain Acceleration"

Model 1 "Frequency Domain Acceleration"

Network Architecture

The model 1 "Frequency Domain Acceleration" uses the same architecture as for model 3 "Time Domain Acceleration", which has returned the best results until now.

All the samples in the database are standardised.

	N° Neurons	Activation	Dropout Rate
Layer Input	439	X	0.5
Layer 1	300	sigmoid	0.5
Layer 2	300	sigmoid	0.5
Layer 3	300	sigmoid	0.5
Layer Output	1	sigmoid	X

Number of Epochs	10
Batch Size	50
Total Parameters	310201

Table 5.14: Network Architecture - "Frequency Domain Acceleration" - Model 1

Training Results

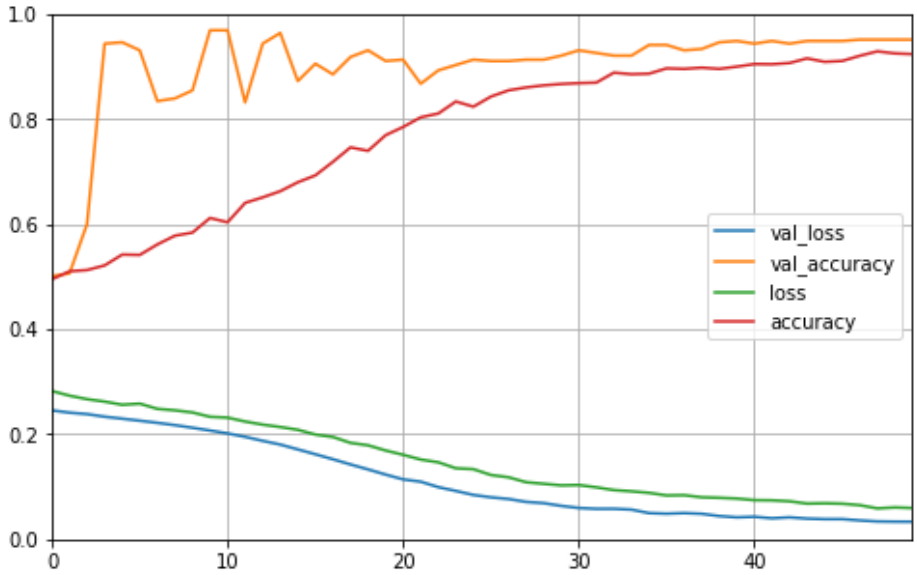
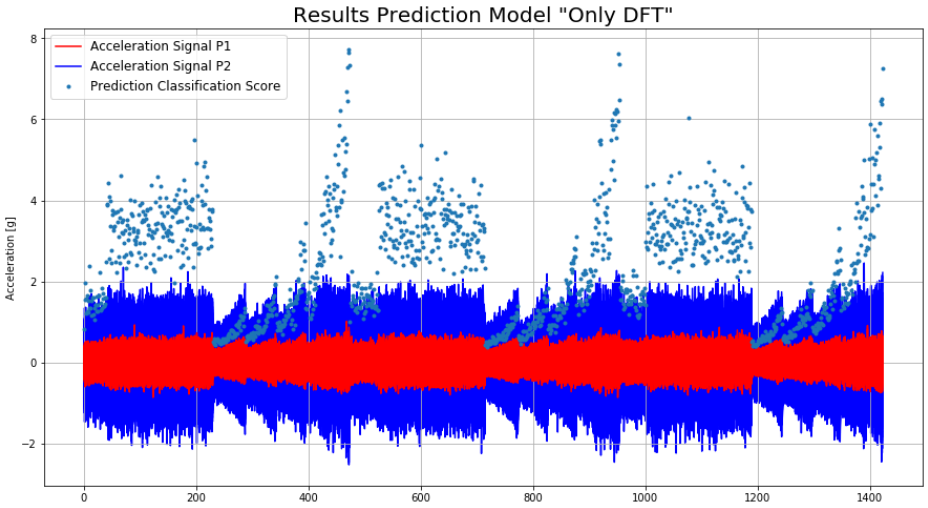


Figure 5.23: Learning Curves - "Frequency Domain Acceleration" - Model 1

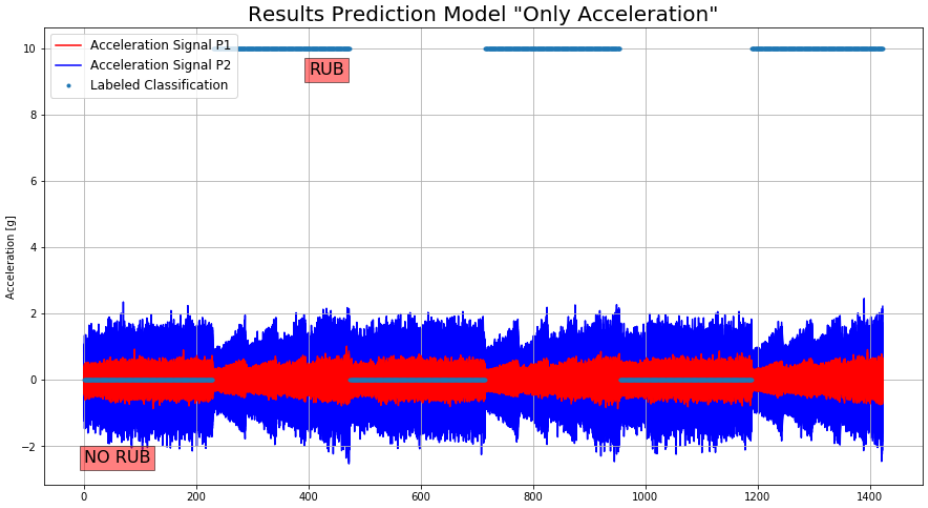
	Loss	Accuracy
Train	0.0594	0.9241
Validation	0.0334	0.9517
Test	0.0346	0.9531

Table 5.15: Training Results - "Frequency Domain Acceleration" - Model 1

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

Figure 5.24: Prediction and Labeled Classification comparison - "Frequency Domain Acceleration" - Model 1

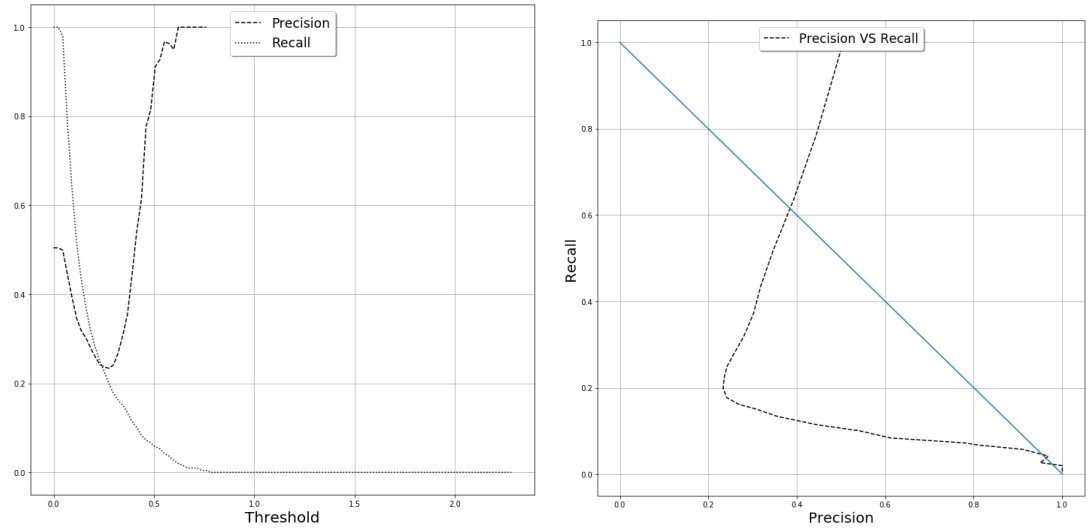
	Predicted NO Rub	Predicted Rub
Label NO Rub	145	561
Label Rub	533	185

Precision	24.79	%
Recall	25.76	%
Accuracy	23.17	%
Especificity	20.53	%

Table 5.16: Confusion Matrix and Results - "Frequency Domain Acceleration" - Model 1

At the moment it is evident from the results of the table 5.16 and the graphs 5.25 that this network is not generalizing the information in the database.

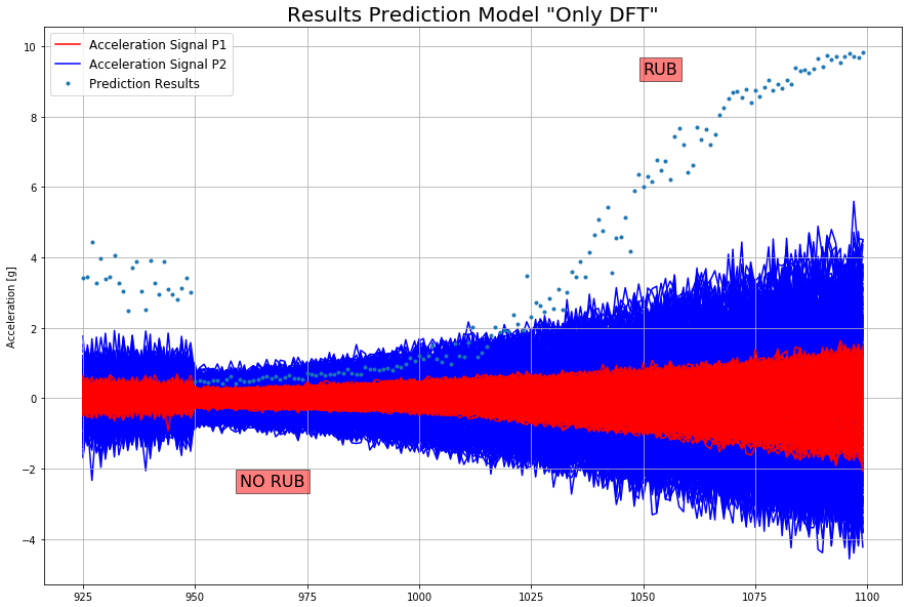
Model Evaluation



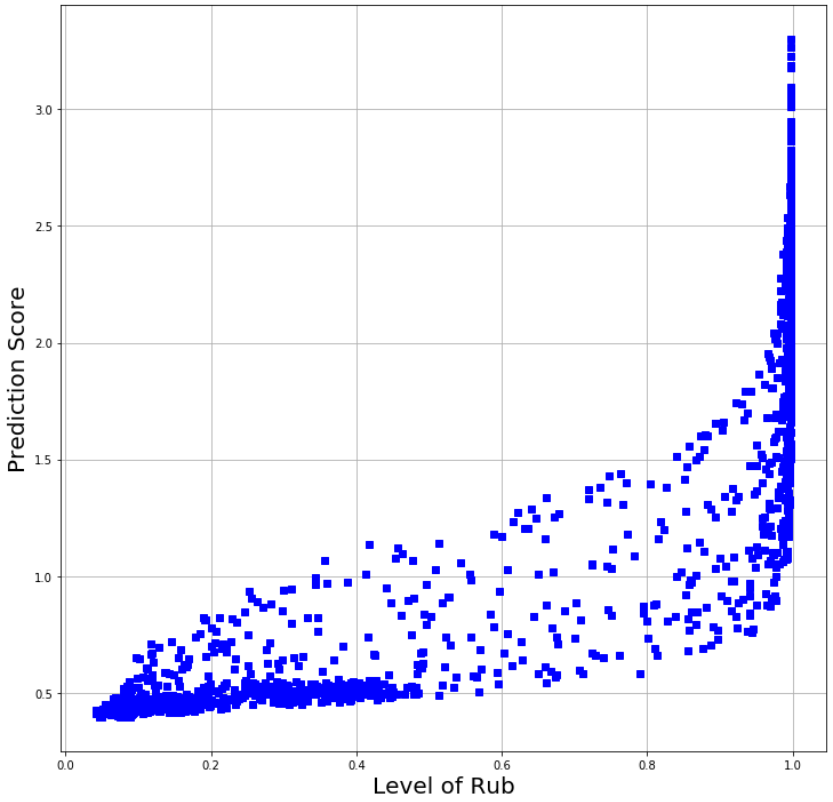
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.25: Relation between Threshold/Precision/Recall - "Frequency Domain Acceleration" - Model 1



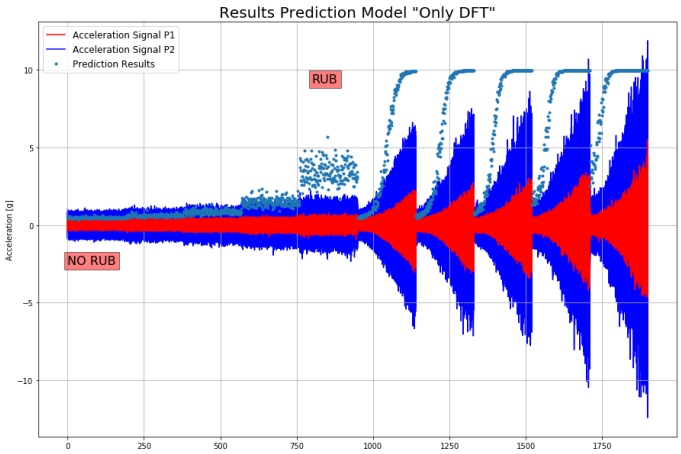
(a) Minimum Rub Level Detection



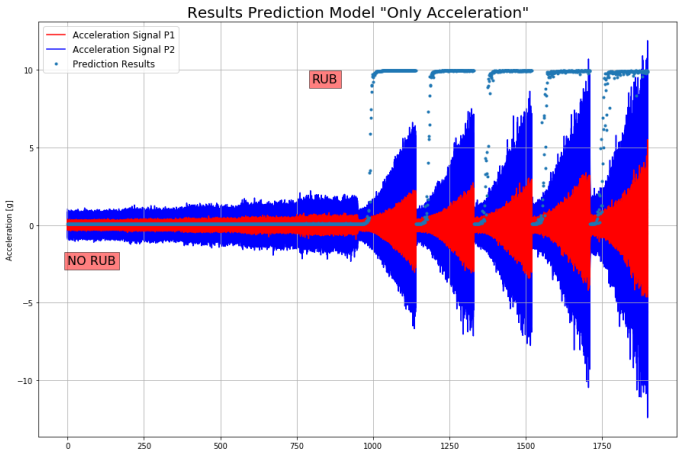
(b) Relation between Prediction Results and Level of Rub

Figure 5.26: Level of Rub Prediction Analysis - "Frequency Domain Acceleration" - Model 1

Although the results of this model are rather disappointing and the prediction is very confusing and uncertain, I would focus to analyze the following graph 5.27, which shows the prediction with respect to the entire dataset for the evaluation of the rub level sensitivity of the model. In fact, here we can observe very clearly how the code is able to detect the different intensities of white noise, unlike model 3 "Time Domain Accelerations". This model, although it returns much better general results, is not able to clearly distinguish the various white noise intensities in the dataset without rub. It is therefore possible to imagine in the future to build a second neural network downstream that, by analyzing the results of several samples in a row, can understand whether the rub is occurring in the turbine, using the prediction results of the first network to filter out the white noise.



(a) Level of Rub Prediction on the total dataset - "Frequency Domain Acceleration" - Model 1



(b) Level of Rub Prediction on the total dataset - "Time Domain Acceleration" - Model 3

Figure 5.27: Comparison between the Level of Rub Prediction on the total dataset

Model 2 "Frequency Domain Acceleration"

Network Architecture

These new model presents an optimize number of neurons on the hidden layers according to the new size of the input. The activation funcions of the hidden layers are again the "tanh".

But the substantial difference between the first and the second "Frequency Domain Acceleration" model is the diverse feature scaling process. In model 2 are performed two featuring scale operation on the database: the first one is a standardization applied to all the data in the database at that frequency range. So all the harmonics at the same frequency are compared, allowing to understand the relative importance of each harmonic. Then a normalization is applied on each sample. The normalization has returned better results than standardization in this case, so we opted for this solution of featuring scales.

	N° Neurons	Activation	Dropout Rate
Layer Input	439	X	0.5
Layer 1	500	tanh	0.5
Layer 2	500	tanh	0.5
Layer 3	500	tanh	0.5
Layer Output	1	sigmoid	X

Number of Epochs	4
Batch Size	20
Total Parameters	310201

Table 5.17: Network Architecture - "Frequency Domain Acceleration" - Model 2

Training Results

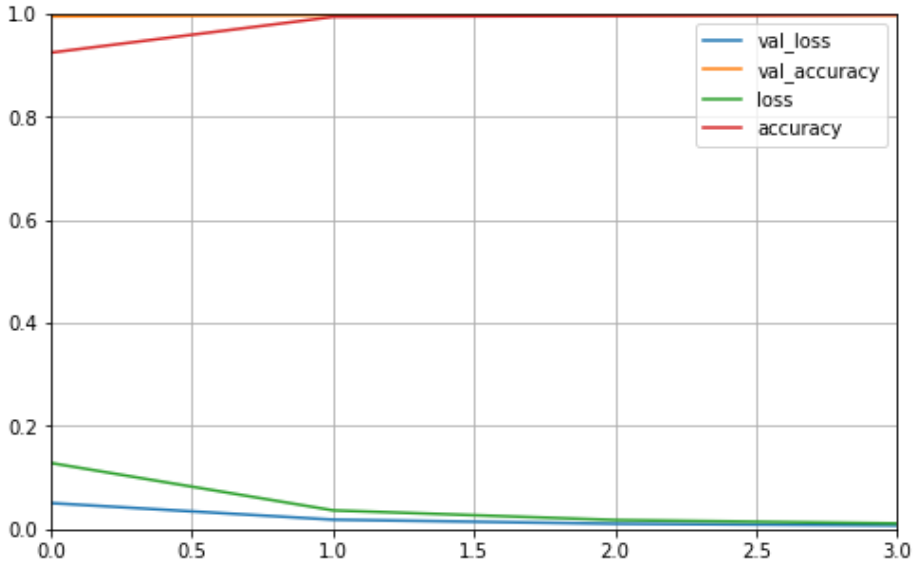
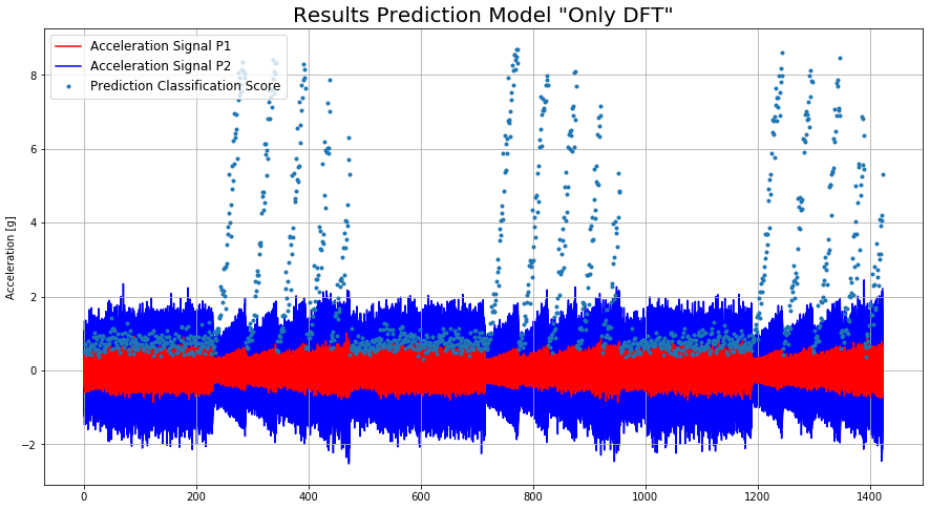


Figure 5.28: Learning Curves - "Frequency Domain Acceleration" - Model 2

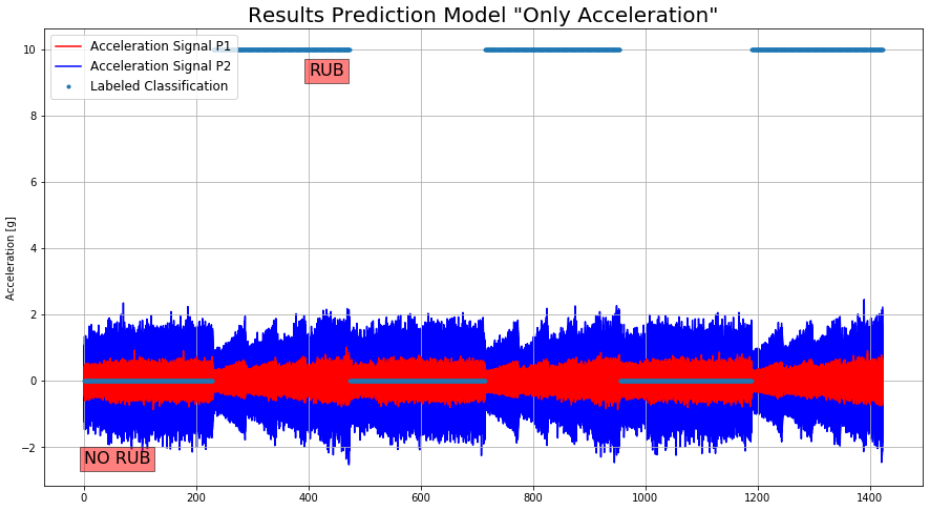
	Loss	Accuracy
Train	0.0110	0.9984
Validation	0.0071	0.9975
Test	0.0071	0.9956

Table 5.18: Training Results - "Frequency Domain Acceleration" - Model 2

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

Figure 5.29: Prediction and Labeled Classification comparison - "Frequency Domain Acceleration" - Model 2

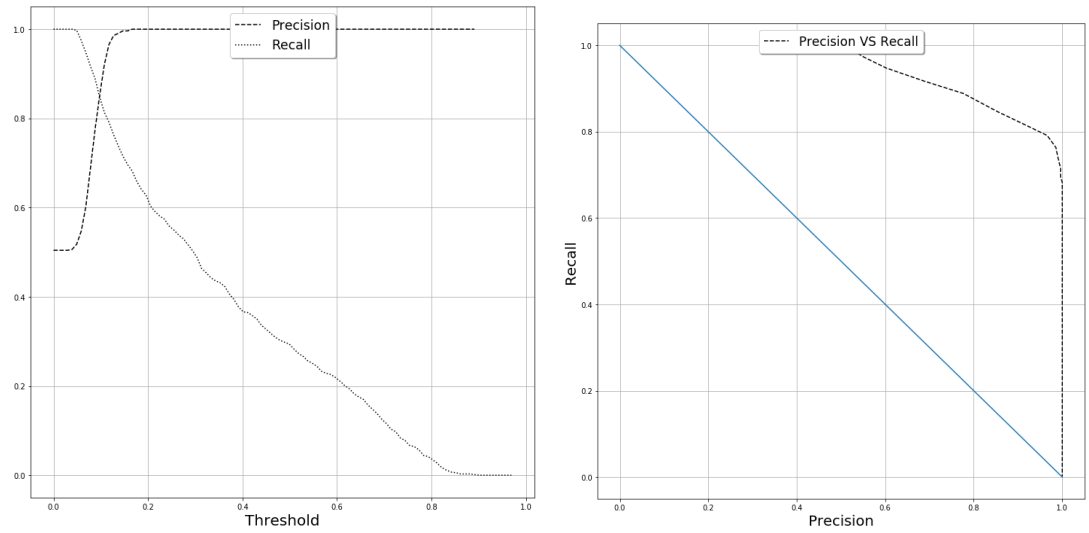
We can see that the results obtained by applying this architecture are clearly superior to the previous model. The values obtained are comparable to the best model of the "Time Domain Acceleration" type. Subsequently it will be necessary to compare the best networks with the experimental data, in order to know which model is the one that best generalises the information of the numerical model data.

	Predicted NO Rub	Predicted Rub
Label NO Rub	595	111
Label Rub	103	615

Precision	84.71	%
Recall	85.65	%
Accuracy	84.97	%
Especificity	84.27	%

Table 5.19: Confusion Matrix and Results - "Frequency Domain Acceleration" - Model 2

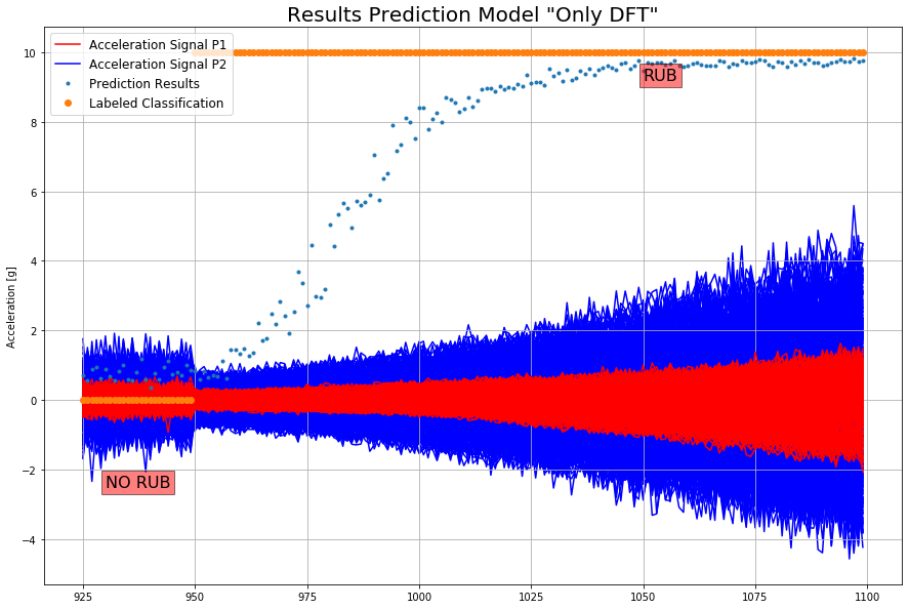
Model Evaluation



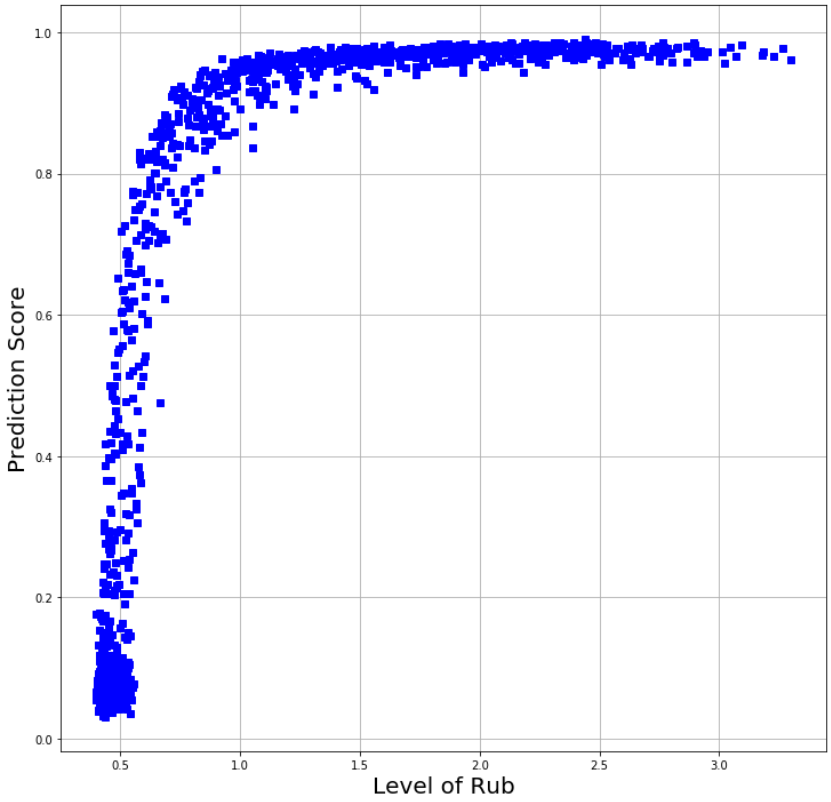
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.30: Relation between Threshold/Precision/Recall - "Frequency Domain Acceleration" - Model 2



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.31: Level of Rub Prediction Analysis - "Frequency Domain Acceleration" - Model 2

Model 3 "Frequency Domain Acceleration"

Network Architecture

The architecture of model 3 is identical to model 2, except for the hidden layers activation functions. Instead of the "tanh" there is the "elu". It is an activation function similar to the "Relu" without the not-derivability in the 0. Since the derivative of "elu" is more pendant than "tanh", it should improve network learning.

	N° Neurons	Activation	Dropout Rate
Layer Input	439	X	0.5
Layer 1	500	elu	0.5
Layer 2	500	elu	0.5
Layer 3	500	elu	0.5
Layer Output	1	sigmoid	X

Number of Epochs	10
Batch Size	50
Total Parameters	721.501

Table 5.20: Network Architecture - "Frequency Domain Acceleration" - Model 3

Training Results

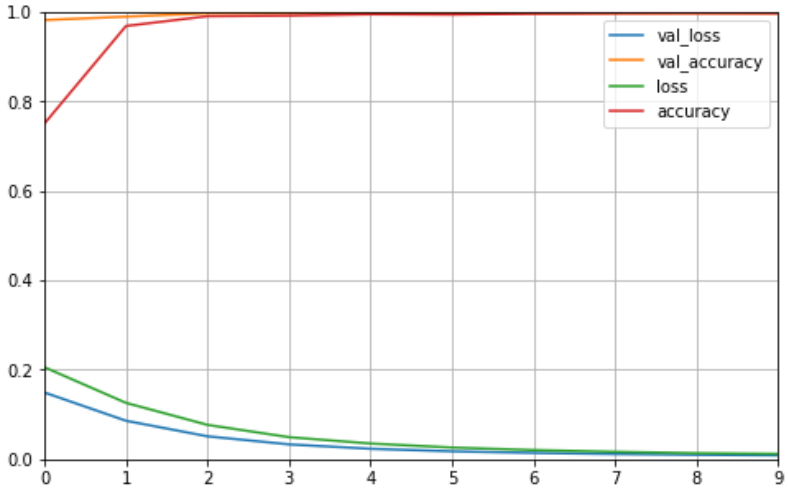
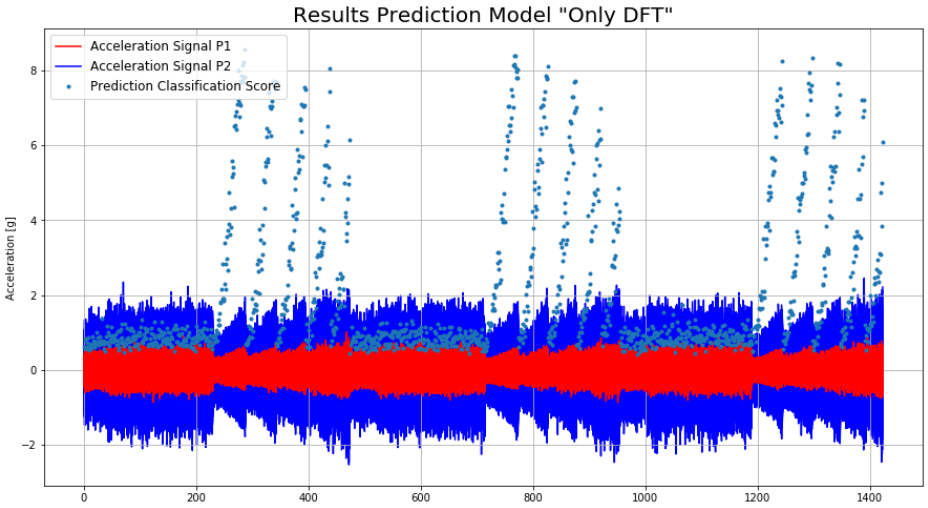


Figure 5.32: Learning Curves - "Frequency Domain Acceleration" - Model 3

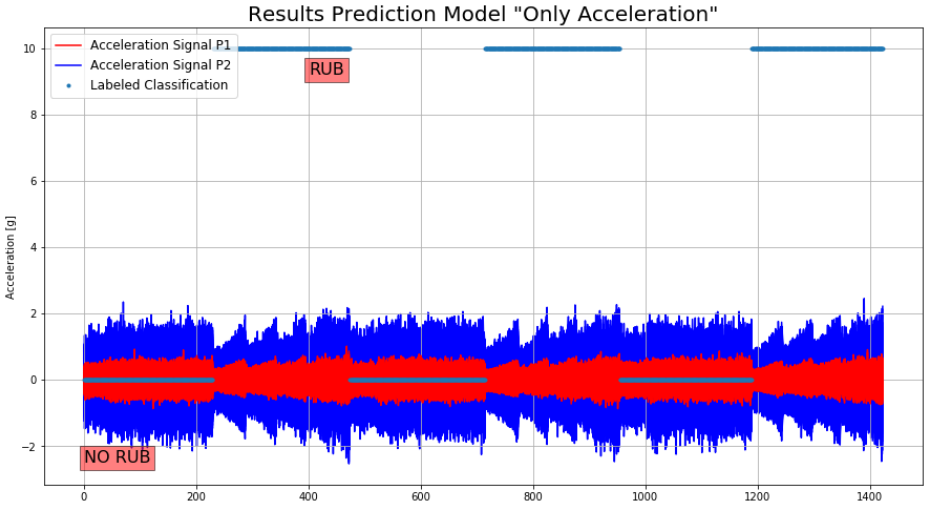
	Loss	Accuracy
Train	0.0119	0.9974
Validation	0.0086	0.9975
Test	0.0082	0.9956

Table 5.21: Training Results - "Frequency Domain Acceleration" - Model 3

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

Figure 5.33: Prediction and Labeled Classification comparison - "Frequency Domain Acceleration" - Model 3

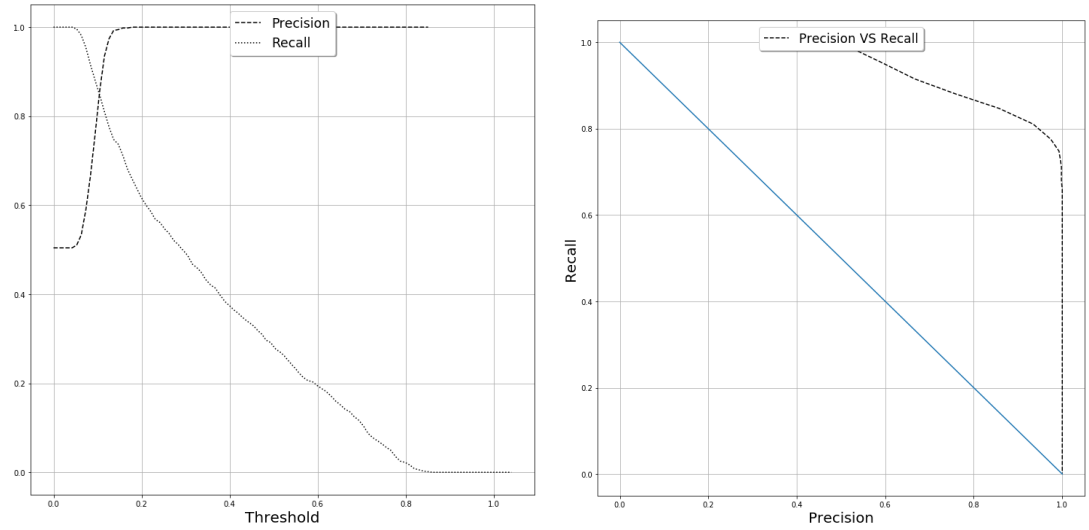
	Predicted NO Rub	Predicted Rub
Label NO Rub	591	115
Label Rub	110	608

Precision	84.09	%
Recall	84.67	%
Accuracy	84.19	%
Especificity	83.71	%

Table 5.22: Confusion Matrix and Results - "Frequency Domain Acceleration" - Model 3

The results of the model 3 confusion matrix are slightly worse than model 2, but also this model returns excellent prediction and recall values.

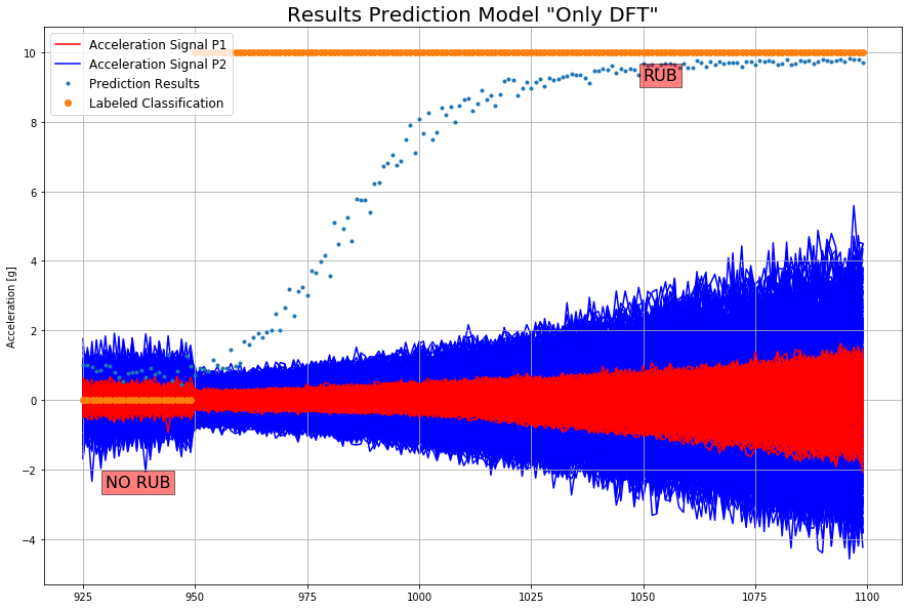
Model Evaluation



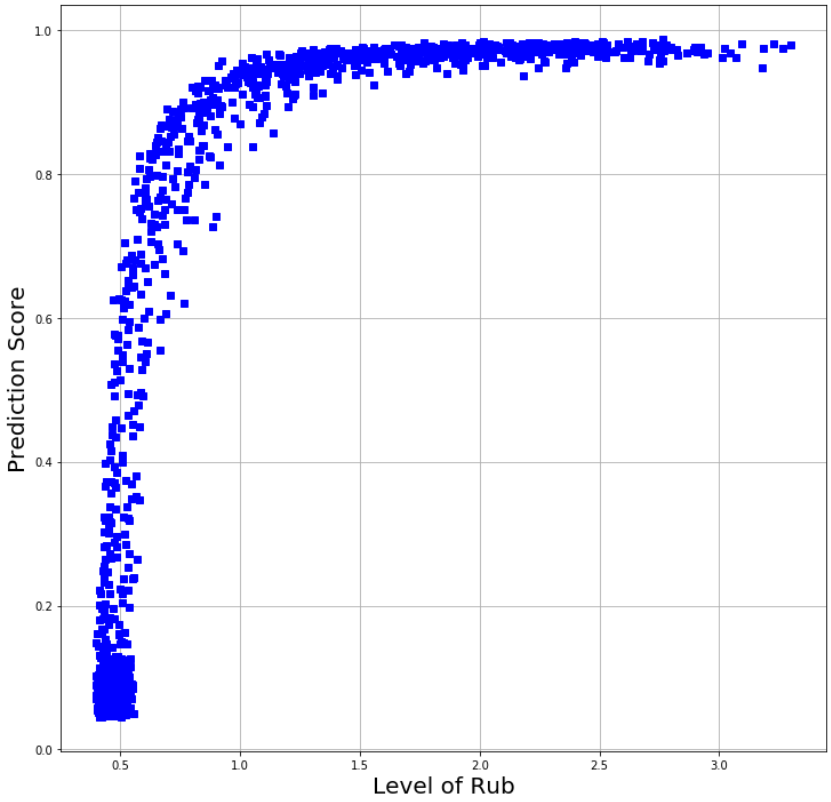
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.34: Relation between Threshold/Precision/Recall - "Frequency Domain Acceleration" - Model 3



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.35: Level of Rub Prediction Analysis - "Frequency Domain Acceleration" - Model 3

5.5.3 Acceleration and DFT Model

Subsequent models are powered by both DFT and accelerometer signal simultaneously. Each instance is composed by both sources, with 1028 data (516 for the DFT and 512 for the accelerometer signal, for P1 and P2). All the layers that composed the network are “Dense" type.

The goal of this type of models is to find a solution that uses both sources of information simultaneously to improve neural network performance by extracting more data from each sample.

Optimizer	***Stochastic Gradient Descendent (sgd)***
Loss Function	***Mean Squared Error***
Metrics	***Accuracy***

Table 5.23: Model Compiler General Setting - "Time and Frequency Domain Acceleration"

The applied PCA transformation presents an explained variance of 0.99, reducing the sample size to 861 data. All models present a standardization of the DFT generated data with the same frequency range and then the scaling feature is performed with the normalization over all the samples.

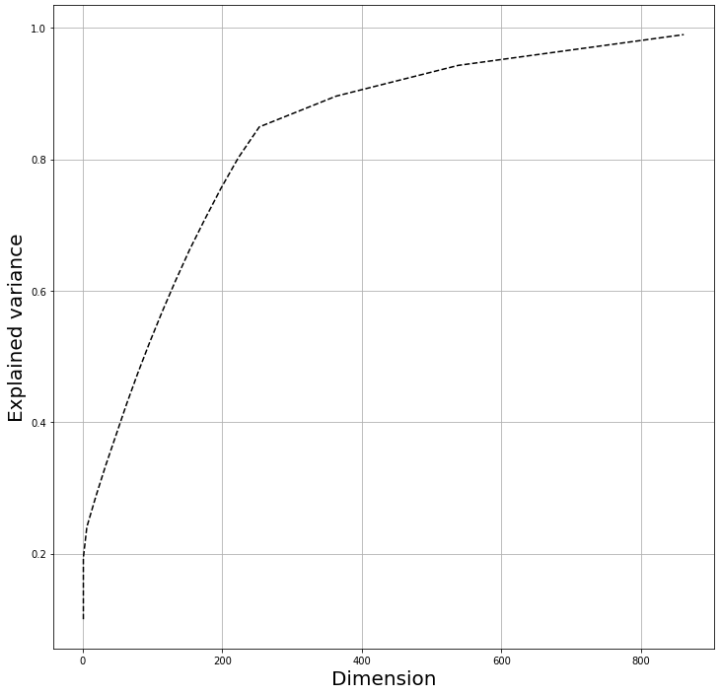


Figure 5.36: Variance Curve PCA reduction- "Time and Frequency Domain Acceleration"

Model 1 "Time and Frequency Domain Acceleration"

Network Architecture

The architecture of the Model 1 "Time and Frequency Domain Acceleration" presents layers optimized respect to the number of neurons of the input layer. The activation function is "tanh" for the hidden layers and "sigmoid" for the last layer, which are the combination that has returned the best results so far.

	N° Neurons	Activation	Dropout Rate
Layer Input	861	X	0.5
Layer 1	1000	tanh	0.5
Layer 2	1000	tanh	0.5
Layer 3	1000	tanh	0.5
Layer 4	1000	tanh	0.5
Layer Output	1	sigmoid	X

Number of Epochs	10
Batch Size	50
Total Parameters	3.866.001

Table 5.24: Network Architecture - "Time and Frequency Domain Acceleration" - Model 1

Training Results

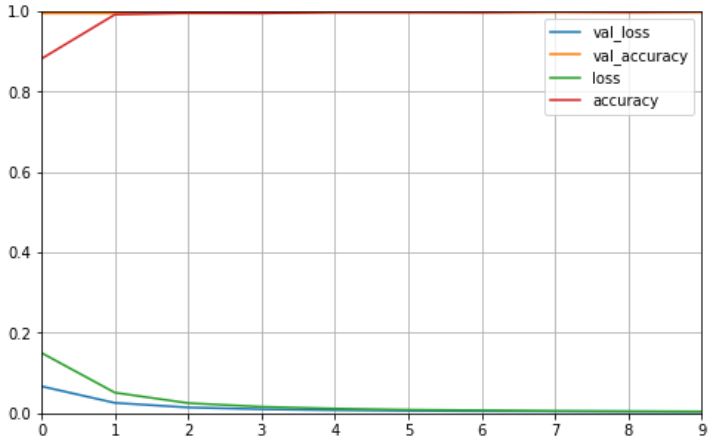
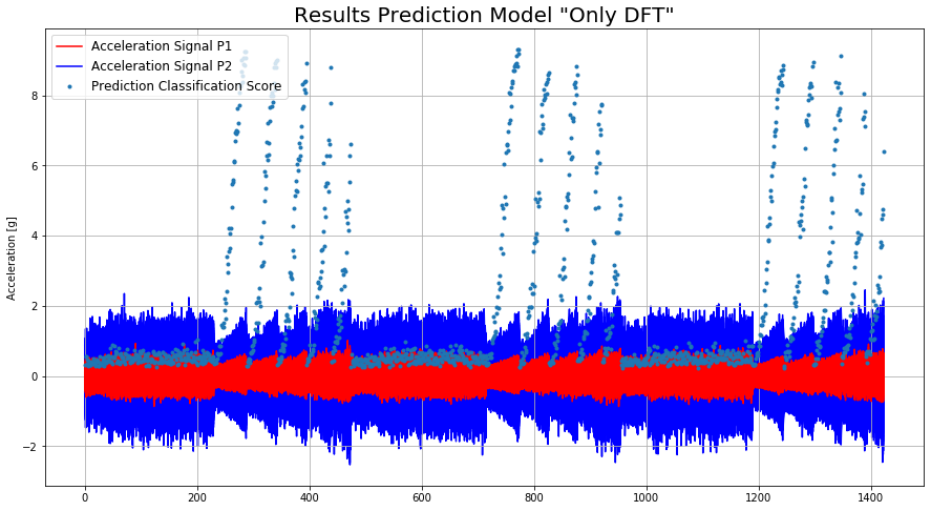


Figure 5.37: Learning Curves - "Time and Frequency Domain Acceleration" - Model 1

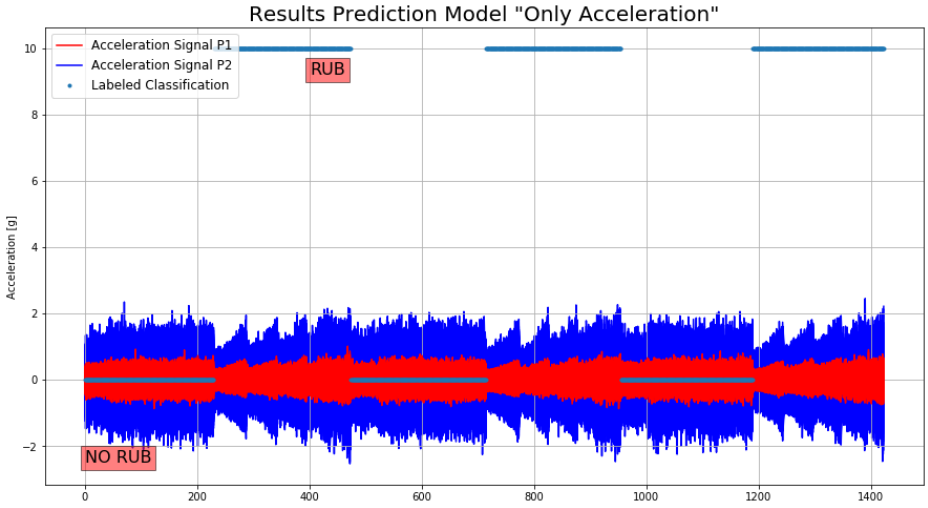
	Loss	Accuracy
Train	0.0043	0.9987
Validation	0.0037	0.9975
Test	0.0030	0.9989

Table 5.25: Training Results - "Acceleration and DFT" - Model 1

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

Figure 5.38: Prediction and Labeled Classification comparison - "Time and Frequency Domain Acceleration" - Model 1

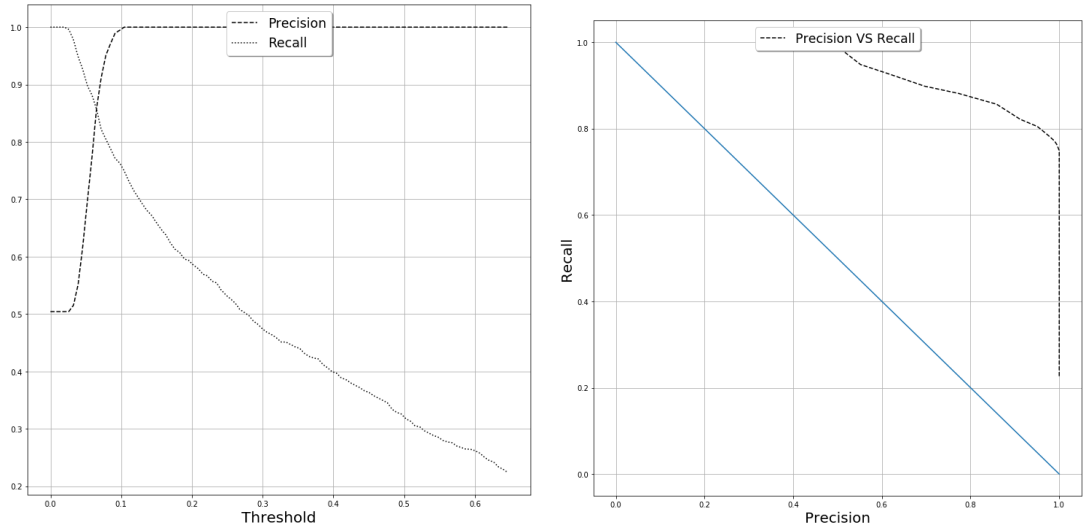
	Predicted NO Rub	Predicted Rub
Label NO Rub	612	94
Label Rub	86	632

Precision	84.96	%
Recall	85.79	%
Accuracy	85.18	%
Especificity	84.56	%

Table 5.26: Confusion Matrix and Results - "Time and Frequency Domain Acceleration" - Model 1

The results obtained with this first model are good but do not exceed the results obtained by models that use the accelerometer signal alone.

Model Evaluation

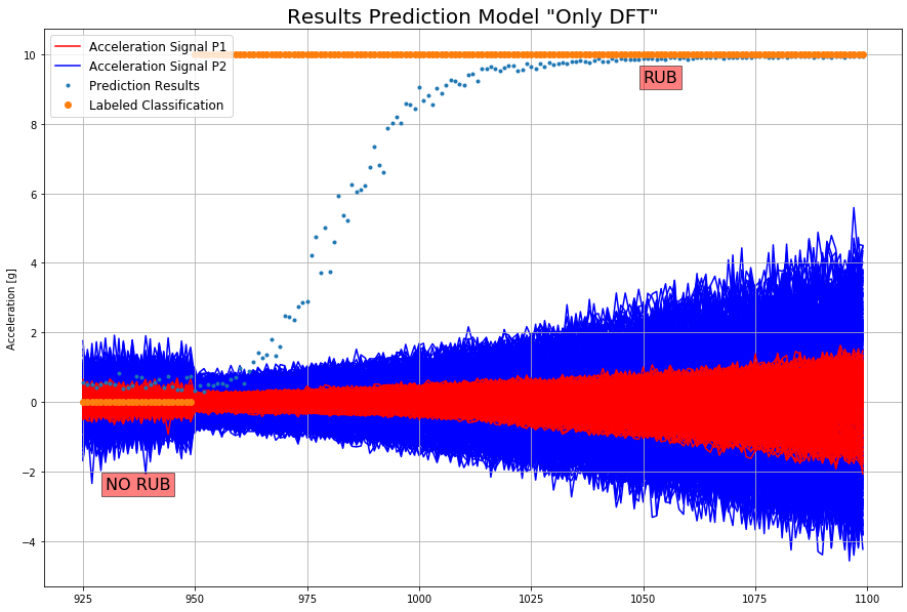


(a) Relation between the Threshold and the Precision/Recall

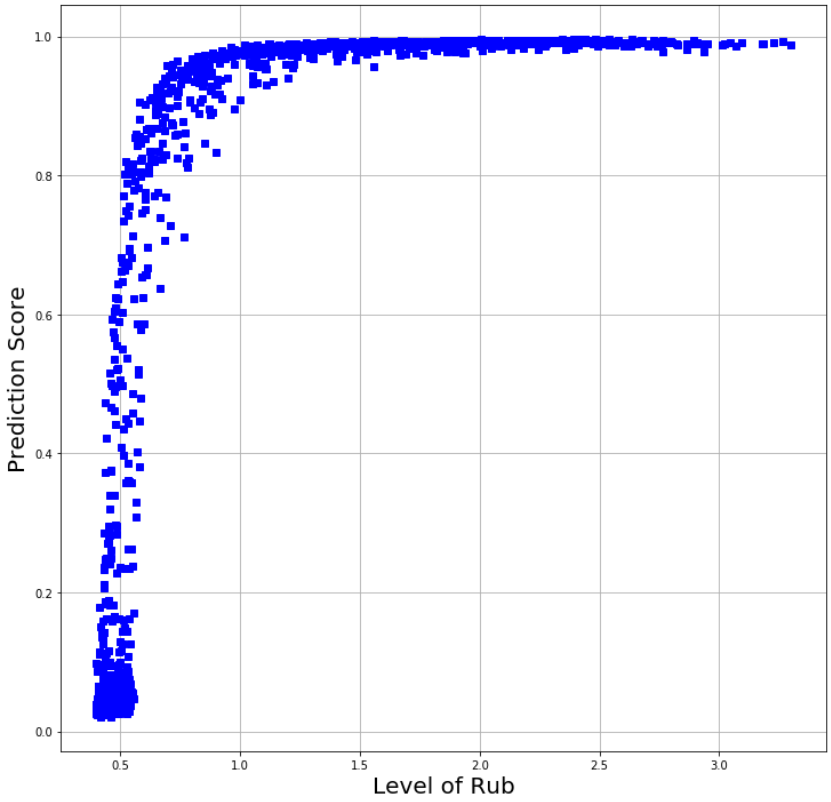
(b) Precision VS Recall

Figure 5.39: Relation between Threshold/Precision/Recall - "Time and Frequency Domain Acceleration" - Model 1

We can note that the trend of the prediction score 5.40 is similar to that obtained for the "Frequency Domain Acceleration" models and not like the "Time Domain Acceleration" models [5.31].



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.40: Level of Rub Prediction Analysis - "Time and Frequency Domain Acceleration" - Model 1

Model 2 "Time and Frequency Domain Acceleration"

Network Architecture

This new model presents more neurons in each layer in order to extract as much information as possible from the database. The activation functions used are "Relu" and "Tanh" for the hidden layers and "sigmoid" for the last layer. The "Relu" should improve the training of the network thanks to a higher slope of the derived function.

	N° Neurons	Activation	Dropout Rate
Layer Input	861	X	0.5
Layer 1	1200	relu	0.5
Layer 2	1200	relu	0.5
Layer 3	1200	relu	0.5
Layer 4	1200	tanh	0.5
Layer Output	1	sigmoid	X

Number of Epochs	5
Batch Size	20
Total Parameters	5.359.201

Table 5.27: Network Architecture - "Time and Frequency Domain Acceleration" - Model 2

Training Results

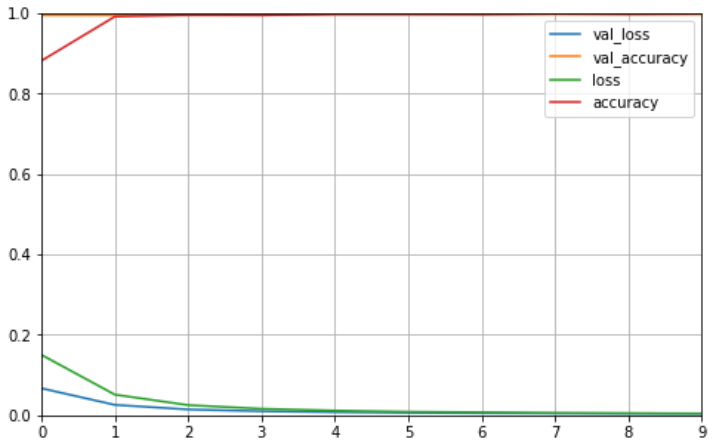
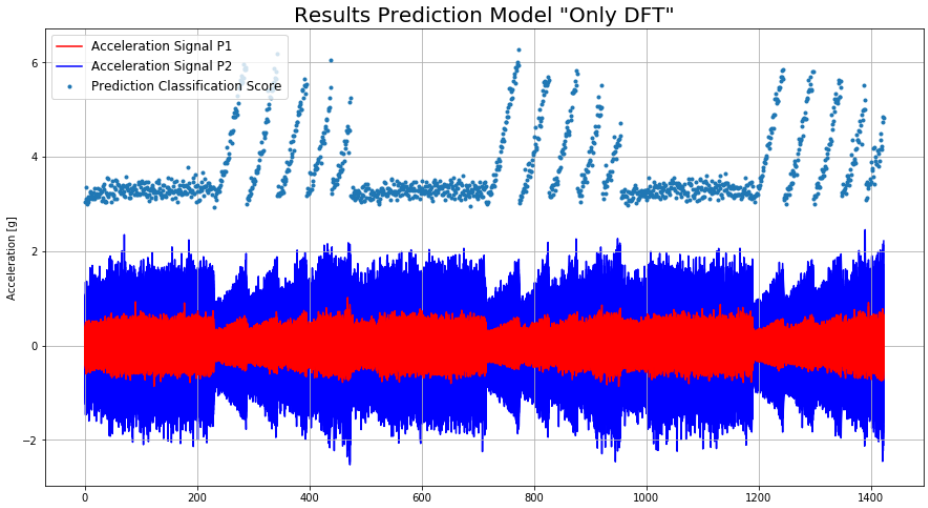


Figure 5.41: Learning Curves - "Time and Frequency Domain Acceleration" - Model 2

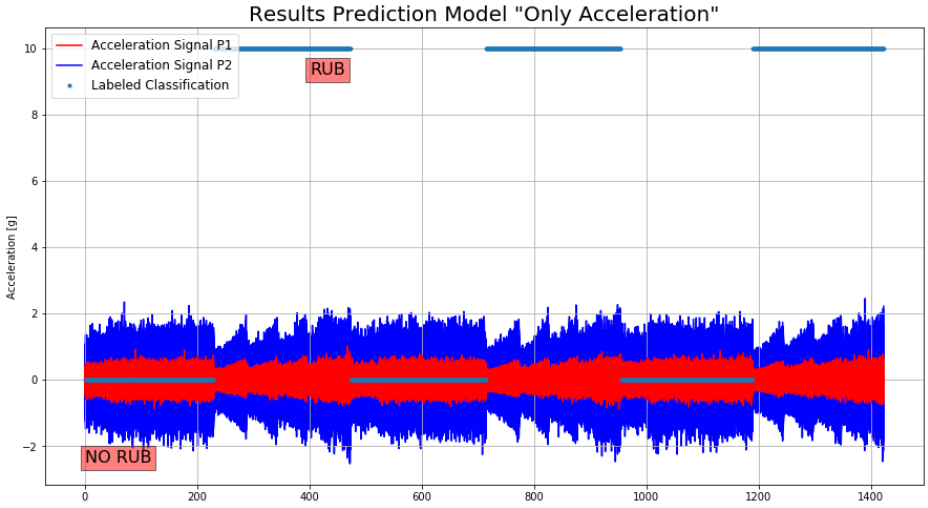
	Loss	Accuracy
Train	0.1636	0.9751
Validation	0.1043	0.9924
Test	0.1039	0.9945

Table 5.28: Training Results - "Acceleration and DFT" - Model 2

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

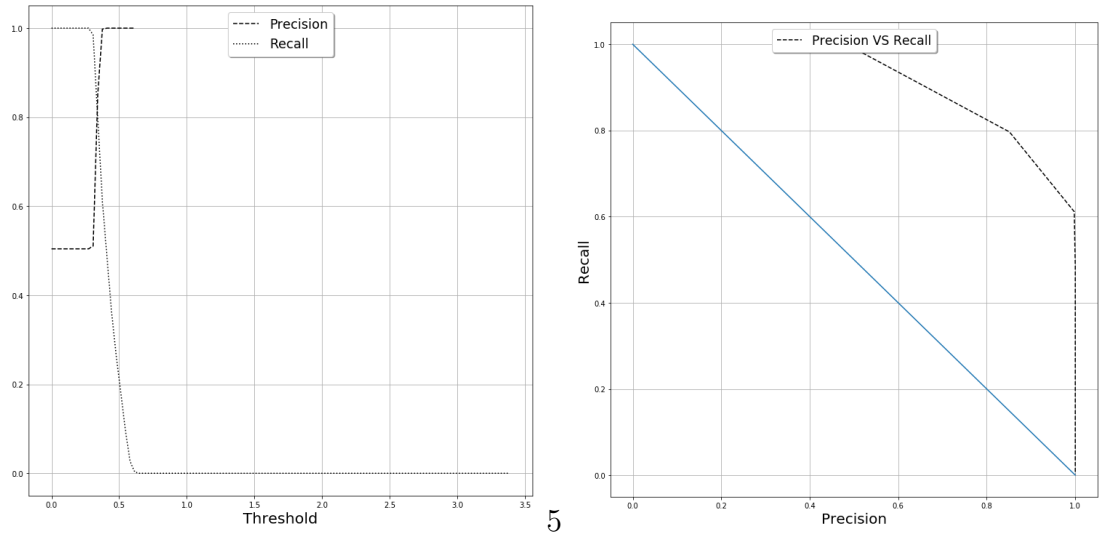
	Predicted NO Rub	Predicted Rub
Label NO Rub	553	153
Label Rub	127	591

Precision	79.43	%
Recall	82.31	%
Accuracy	80.34	%
Especificity	78.32	%

Table 5.29: Confusion Matrix and Results - "Time and Frequency Domain Acceleration" - Model 2

The model works, only it presents slightly lower results compare to the other one. The Prediction/Recall curve 5.42 (a) shows an excessive sensitivity to the threshold value that makes difficult to obtain repeatable results by re-training the network.

Model Evaluation

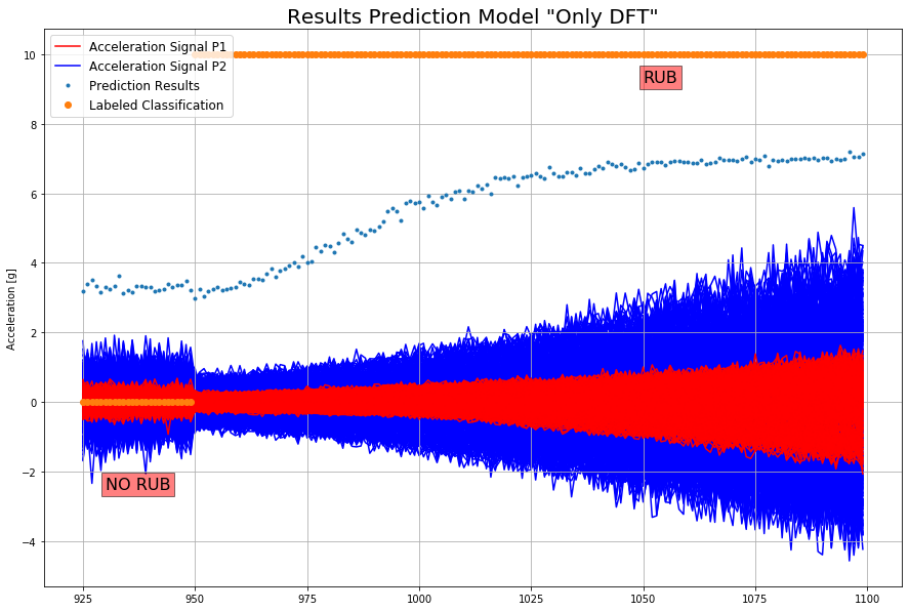


5

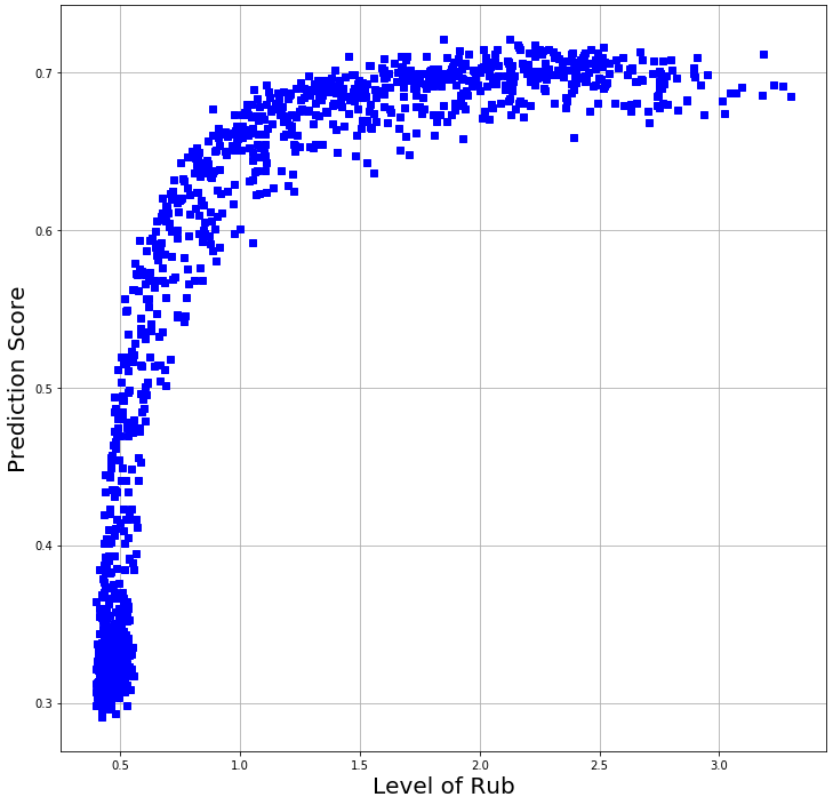
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.42: Relation between Threshold/Precision/Recall - "Time and Frequency Domain Acceleration" - Model 2



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.43: Level of Rub Prediction Analysis - "Time and Frequency Domain Acceleration" - Model 2

Model 3 "Time and Frequency Domain Acceleration"

Network Architecture

In this latest model we tried to create a network that does not use functions that present saturation. All the layers have the "elu" activation function, also the last one. The number of layer and neurons is the same one of the Model 2 "Time and Frequency Domain Acceleration".

	N° Neurons	Activation	Dropout Rate
Layer Input	861	X	0.5
Layer 1	1200	elu	0.5
Layer 2	1200	elu	0.5
Layer 3	1200	elu	0.5
Layer 4	1200	elu	0.5
Layer Output	1	elu	X

Number of Epochs	5
Batch Size	20
Total Parameters	5.359.201

Table 5.30: Network Architecture - "Time and Frequency Domain Acceleration" - Model 3

Training Results

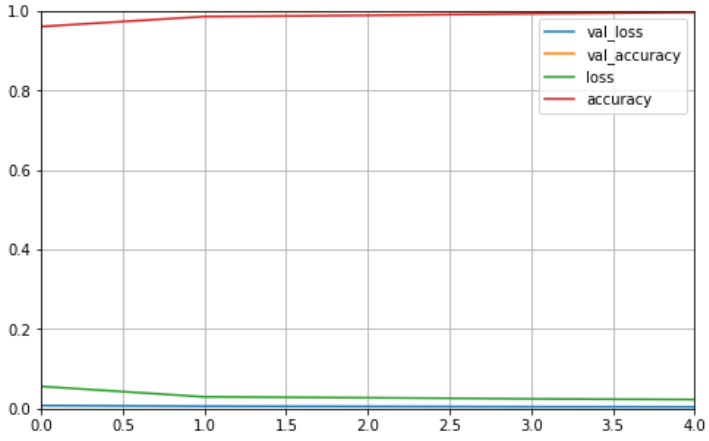
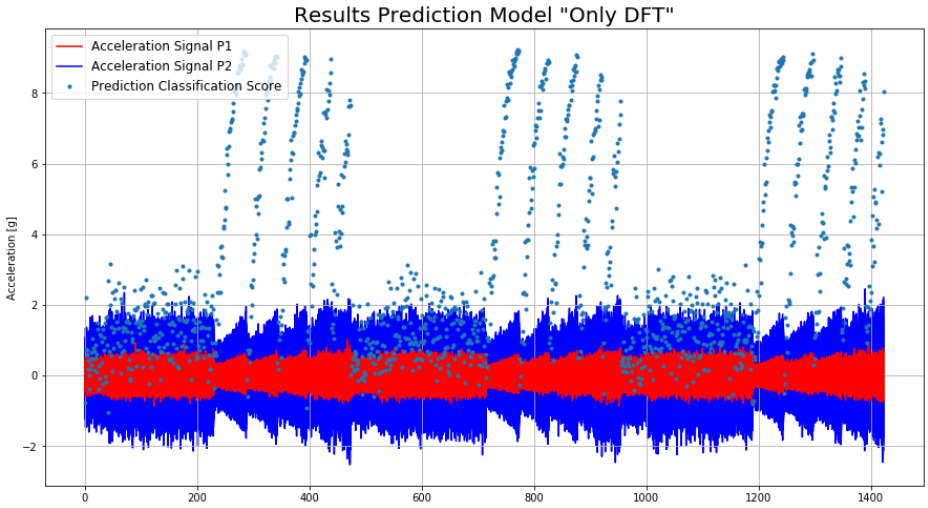


Figure 5.44: Learning Curves - "Time and Frequency Domain Acceleration" - Model 3

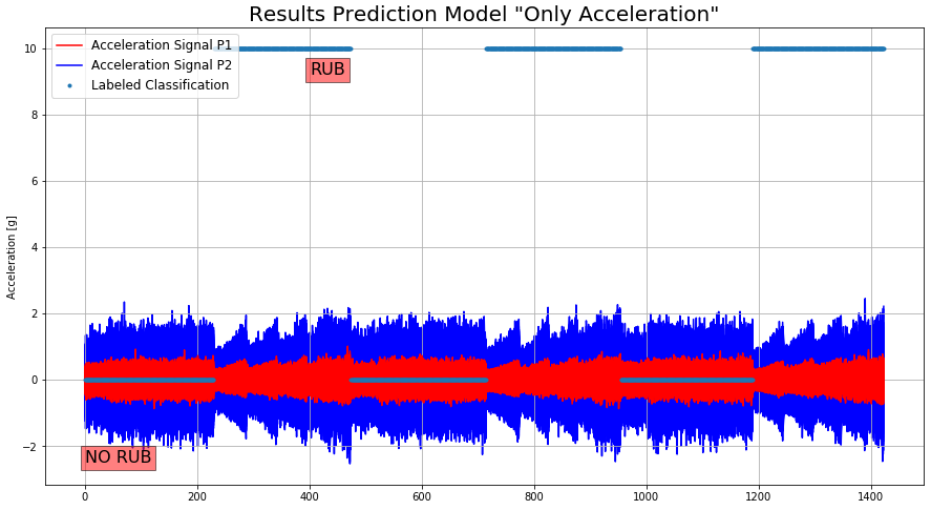
	Loss	Accuracy
Train	0.0380	0.9882
Validation	0.0063	0.9975
Test	0.0054	0.9989

Table 5.31: Training Results - "Acceleration and DFT" - Model 3

Prediction Results



(a) Prediction Results



(b) Labeled Classifications

Figure 5.45: Prediction and Labeled Classification comparison - "Time and Frequency Domain Acceleration" - Model 3

	Predicted NO Rub	Predicted Rub
Label NO Rub	612	94
Label Rub	86	632

Precision	87.05	%
Recall	88.02	%
Accuracy	87.35	%
Especificity	86.68	%

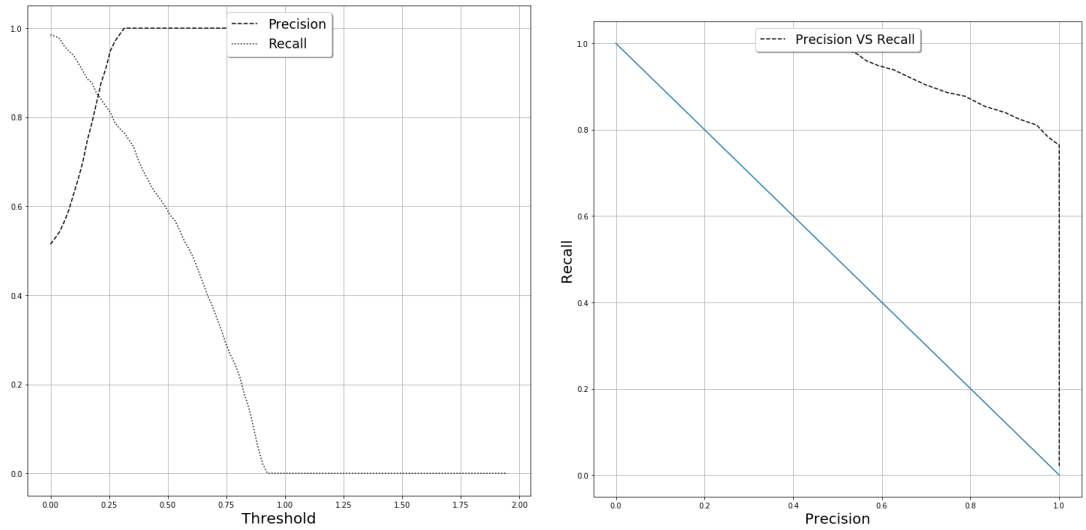
Table 5.32: Confusion Matrix and Results - "Time and Frequency Domain Acceleration" - Model 3

The results of this last model are the highest of all.

This model also starts to predict the rub before everyone else, as we can see in the figure 5.47.

Precision and Recall are not very sensible to the variation of the threshold 5.46.

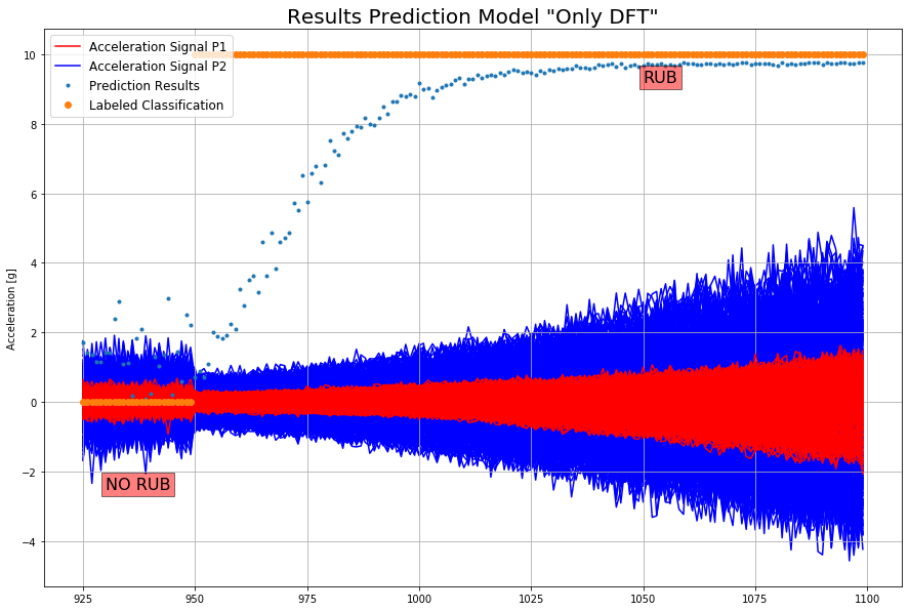
Model Evaluation



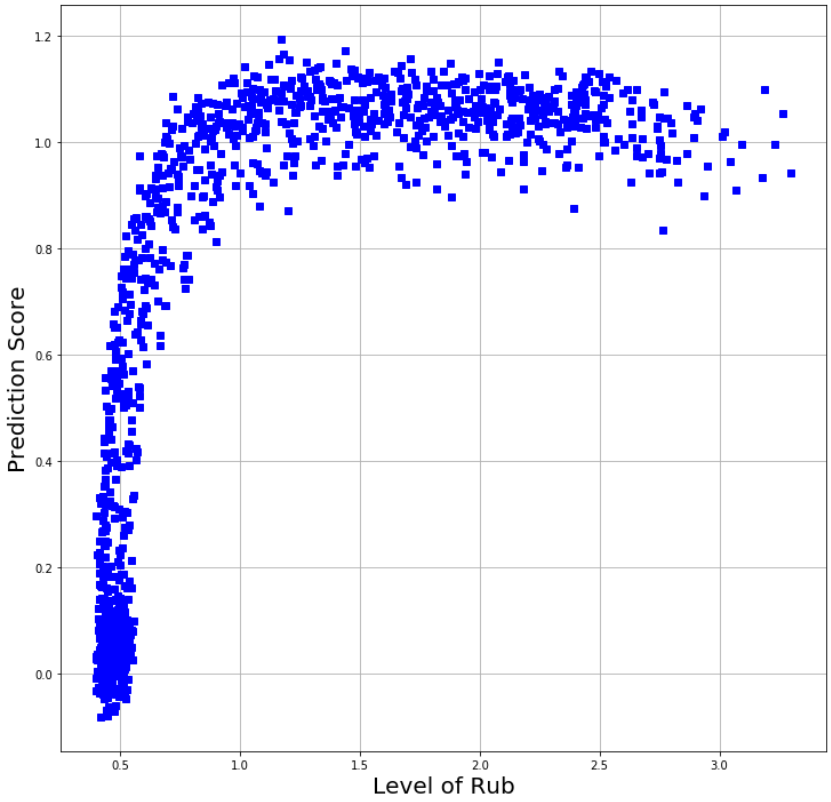
(a) Relation between the Threshold and the Precision/Recall

(b) Precision VS Recall

Figure 5.46: Relation between Threshold/Precision/Recall - "Time and Frequency Domain Acceleration" - Model 3



(a) Minimum Rub Level Detection



(b) Relation between Prediction Results and Level of Rub

Figure 5.47: Level of Rub Prediction Analysis - "Time and Frequency Domain Acceleration" - Model 3

5.6 Result Comparisons

We compare Accuracy and Recall of all models designed to compare their classification performance.

	Time Dom Acc			**Freq Dom Acc**			**Tim Freq Dom Acc**		
	M 1	***M 2***	***M 3***	***M 1***	***M 2***	***M 3***	***M 1***	***M 2***	***M 3***
Precision [%]	78.21	84.78	87.44	24.79	84.71	84.09	84.96	79.43	87.05
Recall [%]	78.97	83.84	88.30	25.76	85.65	84.67	85.79	82.31	88.02

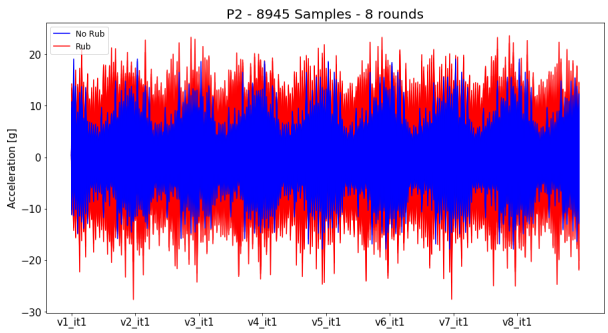
Table 5.33: Comparison Model Results

As we can see, the three strategies may seem, at first glance, to lead back to similar results, except for some models which, at present, yield much worse results. Going to analyze the minimum detectable rub intensity, we can immediately observe that models trained only with the unprocessed accelerometer data need higher oscillation level before labeling the rotation as rub affected (5.13, 5.17), with only the third model that presents better performance 5.21. It is possible to notice that models trained with the Fourier transform detect the intensity of white noise added in the background. We can observe it in these images (5.26, 5.27, 5.29 and 5.33), where the "Frequency Domain Acceleration" neural networks give scores to the samples also in relation to the intensity of background noise. We can imagine that these results can be subsequently reworked by a second neural network that allows us to distinguish more precisely whether the signal is disturbed by background noise or by the stator/rotor contact.
Analyzing the graphs that relate the precision and the Recall (for example 5.47) we can considerate how these are very sensitive to the variation of the threshold value. Currently, to perform the classification, a threshold value of the prediction score is used while another possibility could be to build a second network downstream that analyzes the scores of several consecutive instances.
The models built with both data sources do not present significant improvements in the neural network prediction accuracy and recall. Another possible solution could be to realize a second network that would accommodate the results of two networks optimized specifically to work with acceleration data or with Fourier transform data. Presently, a network working with both data sources has not yielded significant benefits.
We can also note that the choice of the activation function is one of the parameters that most influences the behavior of the network, altering its learning behavior. It is possible to observe how the functions that present saturation of the activation value for positive values (e.g. sigmoid) allow the network to have a more gradual prediction correlated with the variation of the oscillation intensity, as we can see by comparing models 2 and 3 "only Acc" [5.17 and 5.21]. Subsequent studies must be carried out to find the best combination of activation functions.

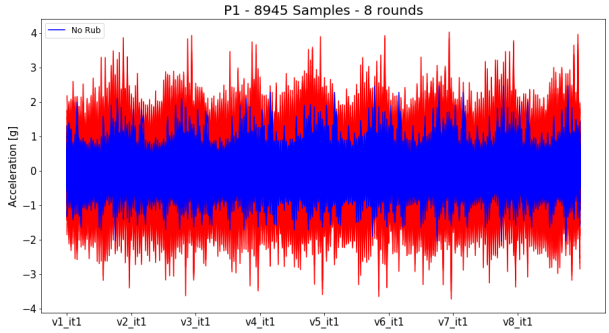
Then we will test the best neural networks trained of each category, with data from the experimental rotary machine model.

5.7 Test with Experimental Data

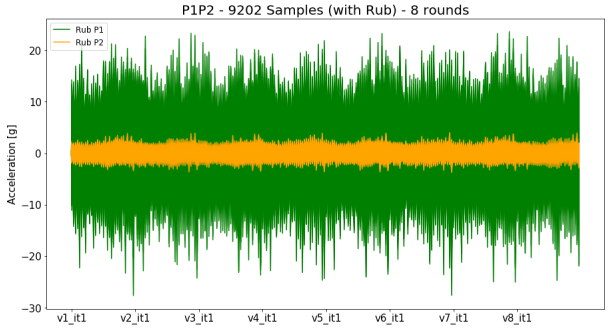
Now we try the prediction classification with the best neural networks trained with synthetic data on the experimental model data.



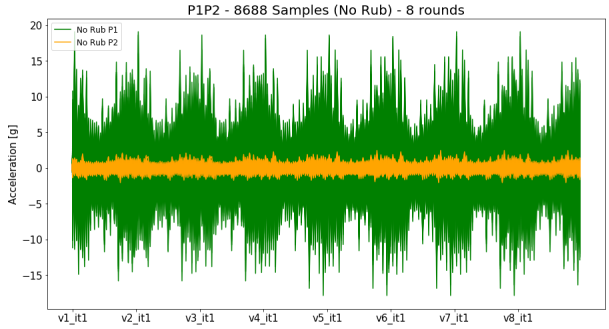
(a) P2 real sensor signal - 8 revolutions - with and without rub



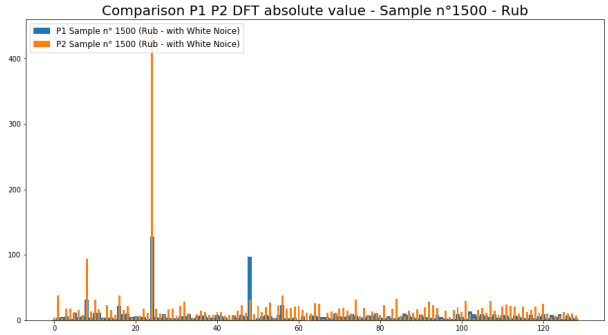
(b) P1 real sensor signal - 8 revolutions - with and without rub



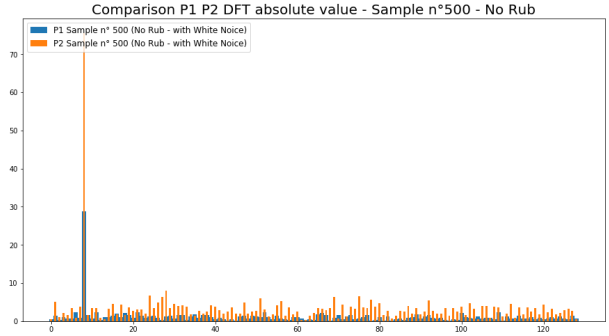
(c) P1 and P2 real sensor signal - 8 revolutions - with rub



(d) P1 and P2 real sensor signal - 8 revolutions - without rub



(e) P1 and P2 real DFT Sample n°500 - 8 revolutions - without rub



(f) P1 and P2 real DFT Sample n°500 - 8 revolutions - with rub

Figure 5.48: Experimental Data Plots

5.7.1 Prediction on Experimental Data - Neural Network "Time Domain Acceleration" Model 3

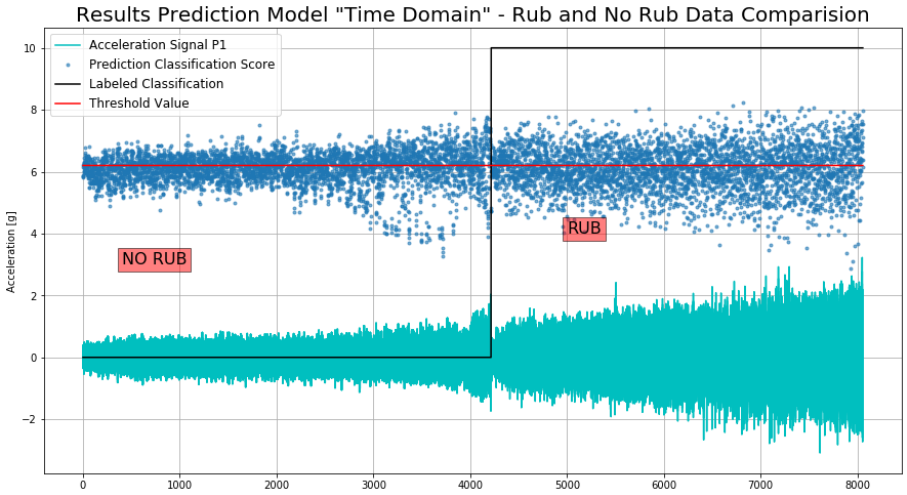
In order to compare the results obtained using the neural network on the experimental data with the data of the numerical model, the instances have been divided into rub and no rub, and then sorted according to the intensity of the oscillation. It is thus possible to observe the model results when the stator/rotor contact is present or when the vibration intensity increases.

As we can notice in observing table 5.34 and figure 5.49, the "Time Domain Acceleration" Model 3 does not generalized well the information. Hence, the results of the prediction are confused and they do not present a correlation between the average intensity of the oscillation and the prediction score. Especially, the figure 5.49 shows how the increase of oscillation is only correlated with an increment in the dispersion of result predictions symmetrically with respect to the threshold.

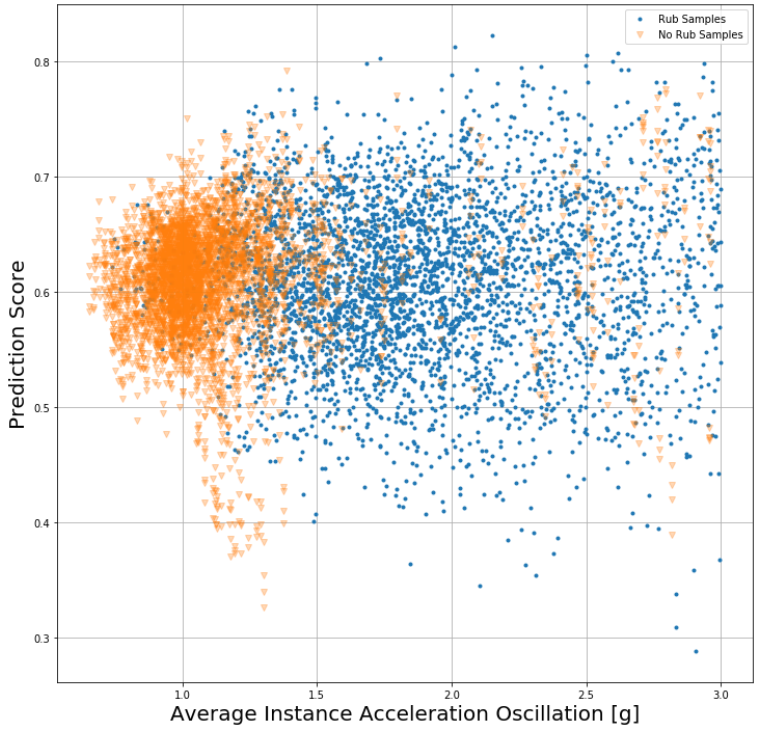
	Predicted NO Rub	**Predicted Rub**
Label NO Rub	2003	2341
Label Rub	2088	2513

Precision	51.77	%
Recall	54.61	%
Accuracy	50.48	%
Especificity	46.11	%

Table 5.34: Experimental data prediction Confusion Matrix and Results - Neural Network "Time Domain Acceleration" Model 3



(a) Experimental data sorted according to the oscillation intensity



(b) Relation between Average Oscillation Intensity and the Prediction Results

Figure 5.49: Prediction on experimental data - Neural Network "Time Domain Acceleration" Model 3

5.7.2 Prediction on Experimental Data - Neural Network "Frequency Domain Acceleration" Model 2

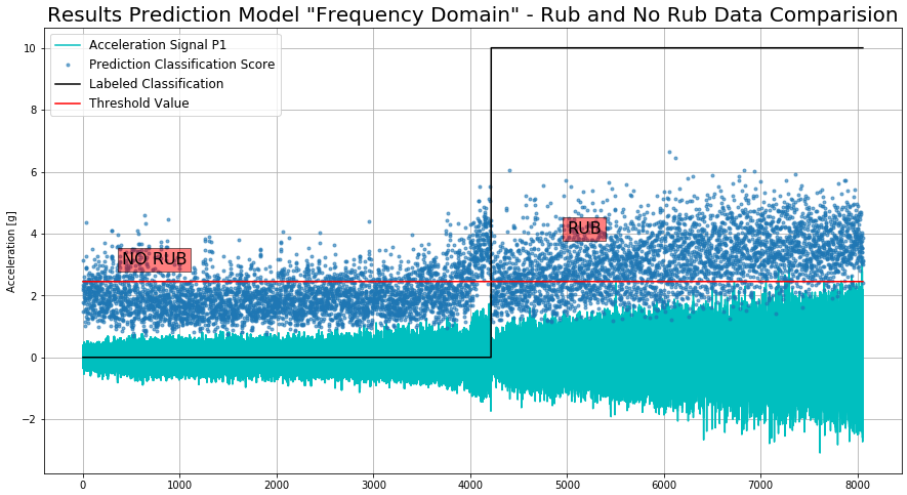
The "Frequency Domain Acceleration" model is capable to better generalize the information contained in the training database. The results of the prediction on the experimental model data are slightly worse than the values achieved on the data of the numerical model. Despite all, we have shown that is possible to create a deep neural network capable of works on a real machine even if the algorithm has been trained with synthetic data.

Compared to the results of the previous model, in this case the network is considerably more capable of classifying instances. Hence, we can see in the figure 5.51 (b) that the prediction values for samples with rub have an average value higher than the threshold, and in graph 5.51 (a) it is also possible to observe a clear division between the two types of data

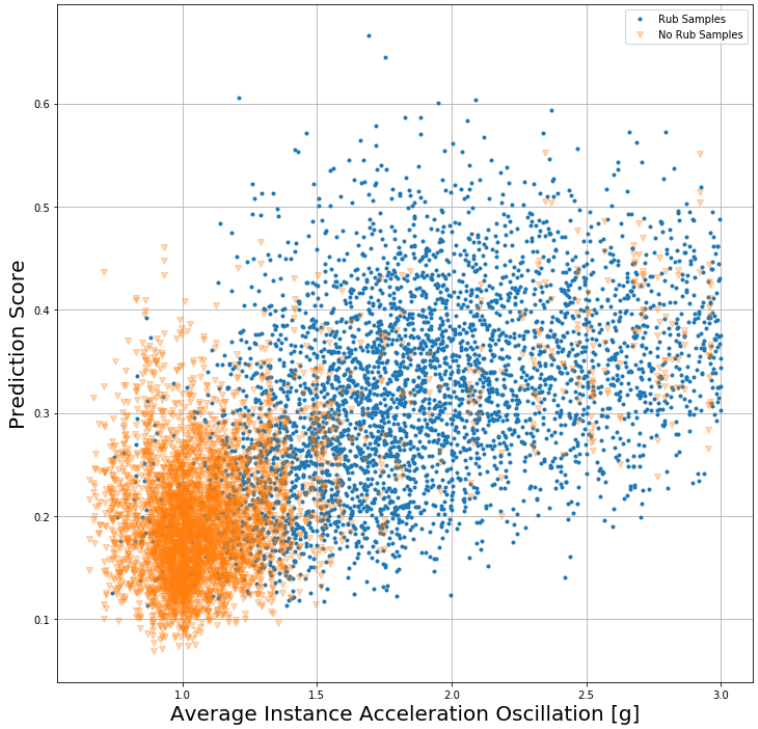
	Predicted NO Rub	**Predicted Rub**
Label NO Rub	3403	941
Label Rub	1062	3539

Precision	79.00	%
Recall	76.92	%
Accuracy	77.61	%
Especificity	78.34	%

Table 5.35: Experimental data prediction Confusion Matrix and Results - Neural Network "Time Domain Acceleration" Model 3



(a) Experimental data sorted according to the oscillation intensity



(b) Relation between Average Oscillation Intensity and the Prediction Results

Figure 5.50: Prediction on experimental data - Neural Network "Frequency Domain Acceleration" Model 3

5.7.3 Prediction on Experimental Data - Neural Network "Time and Frequency Domain Acceleration" Model 3

The results of the "Time and Frequency Domain Acceleration" Model 3 are similar to the "Frequency Domain Acceleration" model 2. However, the latter reaches slightly better values of prediction and recall. Both are able to fulfill the objective of this thesis project.

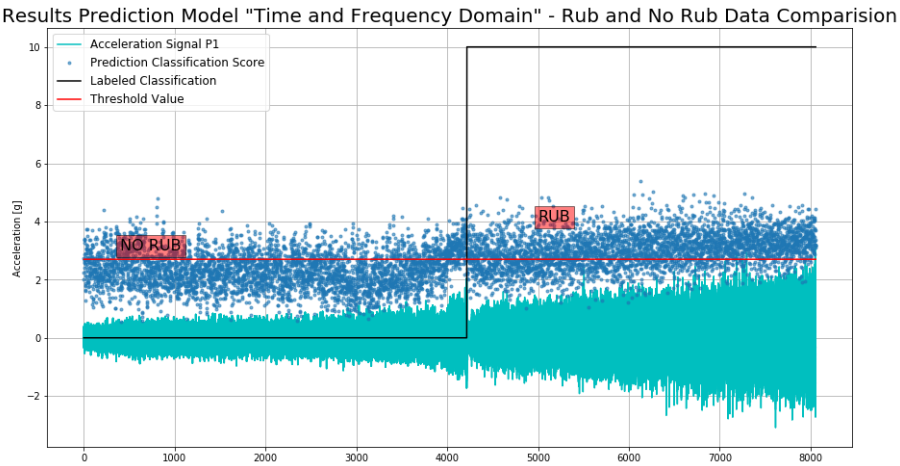
	Predicted NO Rub	**Predicted Rub**
Label NO Rub	3303	1041
Label Rub	1140	3461

Precision	76.87	%
Recall	75.22	%
Accuracy	75.62	%
Especificity	76.03	%

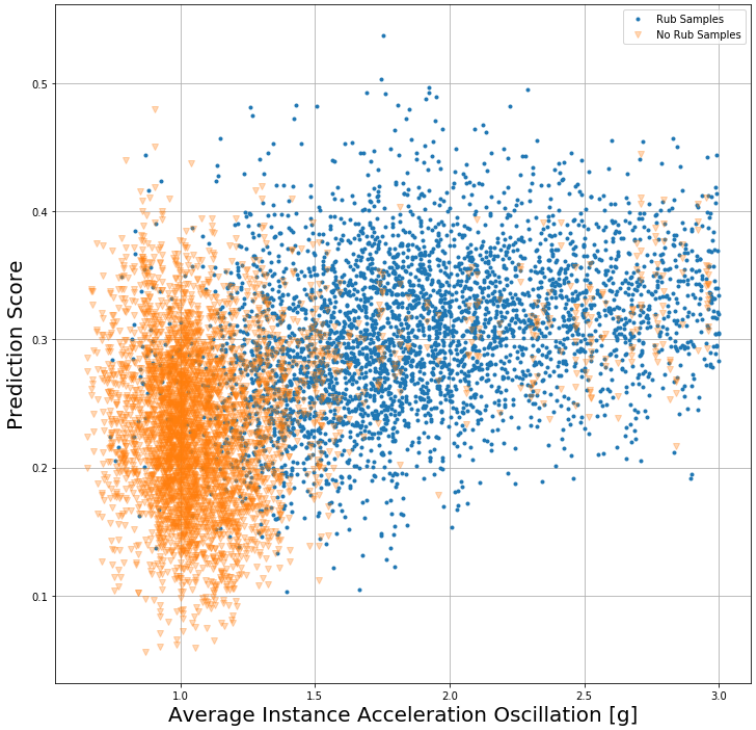
Table 5.36: Experimental data prediction Confusion Matrix and Results - Neural Network "Time and Frequency Domain Acceleration" Model 3

	M3 Time Dom	**M2 Freq Dom**	**M3 Time and Freq Dom**	
Precision	51.77	79.00	76.87	%
Recall	54.61	76.92	75.22	%

Table 5.37: Comparison Model Results - Experimental Data



(a) Experimental data sorted according to the oscillation intensity



(b) Relation between Average Oscillation Intensity and the Prediction Results

Figure 5.51: Prediction on experimental data - Neural Network "Frequency Domain Acceleration" Model 3

5.8 Conclusion and Future Works

As can be seen, although the results on the synthetic data are similar in the three different solutions, feed the network with DFT data allows the neural network to better generalize the patterns present in the accelerometers signal, giving back much higher prediction results in the classification of the real experimental machine data. Neural networks trained only with acceleration data have not shown satisfactory results when it came to detecting the rub within the experimental model.

In addition, the models that have as input only DFT-generated data present one more processing step than the time domain signal. It is therefore important to look for other forms of data pre-processing to see if it is possible to improve the prediction of the "Only time domain" models on experimental data.

The most important result of this thesis project has been to demonstrate that it is possible to build a preventive maintenance software capable of detecting the malfunction of the rub on rotary machines using the signal of the accelerometers located on the external stator part. Hence, this is a possible solution for detecting malfunctions inside an aeroderivative gas turbine.

Above all, the possibility of training the network with data generated by a numerical model similar to the real model saves time and resources for the training of the algorithm, also solving the problem of the shortcoming of a large amount of data of the malfunction.

It is important to underline that the signal processing, so the study and the elaboration of the input data, is one of the crucial factors for a successful design of the architecture of a numerical deep learning neural network model. In our case, thanks to the Fourier transform it has been possible to generalize the data contained in the database and allow the model to work even with real data. Hence, it will be important, in order to advance the project, to study new methods of accelerometers signal processing and implement them in the code to improve the performance of the network.

Moreover, currently the use of both types of information (DFT and acceleration) has not led to a tangible improvement of the results. In the future it will be important to find new solutions to extract more information from the database using both of these information sources.

Although this work is able only to detect one typology of rotating machine malfunction, a new software based on a machine learning algorithm will be able to detect other types of problems, as long as they generate a variation in the operating conditions of the system, which will result in an alteration of the vibrations of the machine.

So, in summary, the next important steps for the project are:

- Optimize network hyperparameters of each model with respect to the type of data used to feed them
- Improve the models that use both data sources (Acceleration and DFT), allowing to extract as much information from each one as possible
- Improve the compatibility of the numerical model with respect to the experimental model, to improve network performance
- Implement the "Wavelet Transform" and see if it can extract more information to improve network prediction
- Optimize network architectures to improve prediction with data generated by the experimental model
- Build a neural network that allows to classify also other types of malfunctions in a aeroderivative gas turbine, such as misalignment

Bibliography

[1] Chupp, R E, R C. Hendricks, S B. Lattime, and B M. Steinetz. "Special Section: Turbine Science and Technology - Sealing in Turbomachinery." Journal of Propulsion and Power. 22.2 (2006): 313

[2] Ma, H, F Yin, B Wen, Y Guo, and X Tai. "A Review on Dynamic Characteristics of Blade-Casing Rubbing." Nonlinear Dynamics. 84.2 (2016): 437-472

[3] Jacquet-Richardet, G, M Torkhani, P Cartraud, F Thouverez, Baranger T. Nouri, M Herran, C Gibert, S Baguet, P Almeida, and L Peletan. "Rotor to Stator Contacts in Turbomachines. Review and Application." Mechanical Systems and Signal Processing. 40.2 (2013): 401-420

[4] Piezoelectric accelerometer sensor Image https://www.google.com/search?q=piezoelectric+accelerometer+sensor&tbm=isch&ved=2ahUKEwiEyamanbnqAhUU8RoKHYB-DwAQ2-cCegQIABAA&oq=piezoelaccelerometer+sensor&gs_lcp=CgNpbWcQARgAMgYIABAHEB46CAgAEAgQBxAeUIgHWI8SYI sclient=img&ei=rWkDX8S-OZTia4D9PQ&bih=686&biw=1396#imgrc=Q8Of1rkSg9nn3M, Accessed 31 August 2020.

[5] Proximity Sensor Image https://www.google.com/search?q=proximity+sensor+image&tbm=isch&ved=2ahUKEwiLrM3gnLnqAhVBYBoKHflsB5IQ2-cCegQIABAA&oq=proximity+sensor+image&gs_lcp=CgNpbWcQAzIECAAQEzIICAAQBRAeEBMyCAgAEAUQHhATOgIIADoECAAQHlC4 sclient=img&ei=NGkDX4vCNsHAafnZnZAJ&bih=686&biw=1396&hl=it#imgrc=Lh6H2WwtXvOaGM, Accessed 31 August 2020.

[6] Gas Turbines Components Image https://www.google.com/search?q=gas+turbine+components&sxsrf=ALeKk03GdmNDuBt6Bh2zsv9igq2qr1Z_rA:1594301396053&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjJpYHDo8DqAhVORBoKHTtvA2cQ_AUoAXoECA0QAw&biw=1396&bih=686#imgrc=j2WCz0T4j8THlM, Accessed 31 August 2020.

[7] Brayton Ideal cycle Image https://www.google.com/search?q=Brayton+cycle&tbm=isch&ved=2ahUKEwiu1t6nurvqAhWB_IUKHXUXDvYQ2-cCegQIABAA&oq=Brayton+cycle&gs_lcp=CgNpbWcQAzICCAAyBAgAEB4yBAgAEB4yBAgAEB4yBAgAEB4yBAgAEB4yBAgAEB4yBAgAEB4yBAgAEB4y sclient=img&ei=opQEX679CoH5lwT1rriwDw&bih=686&biw=1396#imgrc=z-8w0DPIBZ3IaM, Accessed 31 August 2020.

[8] Jet turbines, https://en.wikipedia.org/wiki/Jet_engine, Accessed 31 August 2020.

[9] Mitsubishi Hitachi Power Systems' aeroderivative gas turbine lineup, https://www.powermag.com/the-power-interview-new-directions-for-aeroderivative-gas-turbines-at-pwps/, Accessed 31 August 2020.

[10] SDOF UnderDamped Response Image, https://www.google.com/search?q=underdamped+free+vibration&sxsrf=ALeKk039X0RxLdbnkjgUNongpGM-yB65Iw:1594900437869&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjImLSQ29HqAhUGA2MBHXl9DJ4Q_AUoAXoECA0QAw&biw=1396&bih=686#imgrc=yKGhLI8vvEhb6M, Accessed 31 August 2020.

[11] Resonance Transmissibility, https://www.google.com/search?q=resonance&sxsrf=ALeKk01O6N8wY82AlkeukKVBDTdIPVKe7g:1594903959598&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiNsdmf6NHqAhUKEBQKHaJtAKMQ_AUoAXoECBMQAw&biw=1396&bih=686#imgrc=XaZy8RR8-FDCeM, Accessed 31 August 2020.

[12] Catania, Andrea E. Complementi Di Macchine. Torino: Levrotta Bella, (1979)

[13] Ratliff, Phil, Garbett, Paul, Fischer, Willibald, "The New Siemens Gas Turbine SGT5-8000H for More Customer Benefit" VGB PowerTech Siemens Power Generation, (2007)

[14] "International Energy Outlook 2019 with projections to 2050" EIA - U.S. Energy Information Administration, (2019).

[15] Jie, Hong, Li Tianrang, Liang Zhichao, Zhang Dayi, and Ma Yanhong. "Research on Blade-Casing Rub-Impact Mechanism by Experiment and Simulation in Aeroengines." Shock and Vibration. 2019 (2019).

[16] Robb, Drew. "Cover Story Aeroderivative Gas Turbines." Turbomachinery International. 58.7 (2017): 16-23.

[17] "Global Aeroderivative Gas Turbine Market 2016-2020", Technavio, (2016).

[18] "Why GE's Aeroderivative Gas Turbines Are Critical For Grid Stability" General Electric's YouTube channel, (2019).

[19] Siemens SGT-A65 Aeroderivative gas turbines Image, https://www.siemens-energy.com/global/en/offerings/power-generation/gas-turbines/sgt-a65-tr.html, Accessed 2 September 2020

[20] Heavy duty gas turbine GT26 Image https://www.researchgate.net/figure/Heavy-duty-gas-turbine-GT26_fig1_316901070, Accessed 02 September 2020

[21] "J-Series Gas Turbine" , Mitsubishi Heavy Industries Ltd.

[22] Kellner, Tomas. "Here's Why The Latest Guinness World Record Will Keep France Lit Up Long After Soccer Fans Leave" , General Electric, (2016).

[23] Gregor Stacker "Getting the most out of aeroderivative turbines", PEI Power Engineering International, (2017).

[24] "Aeroderivative or Industrial Gas Turbines", Integra technical service , (2018).

[25] R. Tony, "Defining Preventive Predictive Maintenance", LinkedIn Profile, (2018).

[26] "Types of Maintenance". https://www.roadtoreliability.com/types-of-maintenance/. Accessed 31 August 2020.

[27] Poole, David L, Alan K. Mackworth, and Randy Goebel. "Computational Intelligence: A Logical Approach". New York: Oxford University Press, (1998).

[28] Kaplan, A, and M Haenlein. "Siri, Siri, in My Hand: Who's the Fairest in the Land? on the Interpretations, Illustrations, and Implications of Artificial Intelligence." Business Horizons. 62.1 (2019): 15-25.

[29] A. Turing, "Computing Machinery and Intelligence", University of Manchester, (1950).

[30] Turing test diagram, https://en.wikipedia.org/wiki/Turing_test, Accessed 31 August 2020.

[31] Rosenblatt, F. The Perceptron, a Perceiving and Recognizing Automaton <project Para>. Buffalo, NY: Cornell Aeronautical Laboratory, (1957)

[32] Introduzione al Machine Learning (ML Zero to Hero, parte 1), https://www.youtube.com/watch?v=KNAWp2S3w94&list=PLPKNvFYGf41lOwuHY54XZa9MH9dV_4Gar, Accessed 31 August 2020.

[33] Biological Neural Network Image, https://www.google.com/search?q=biological+neural+network&sxsrf=ALeKk02-Fw3cWuHv9triW-KkDklrN749TA:1594917808163&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiO4Jrrm9LqAhUKahQKHeT1B5oQ_AUoAXoECBIQAw&biw=1396&bih=686#imgrc=8QftKYZVTU7WNM, Accessed 31 August 2020.

[34] Geron, Aurelien. Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Incorporated, (2019).

[35] Sarah Knapton, Leon Watson, "Entire human chess knowledge learned and surpassed by DeepMind's AlphaZero in four hours", Telegraph.co.uk, (2017).

[36] "Introduzione al Machine Learning (ML Zero to Hero, parte 1" https://www.youtube.com/watch?v=KNAWp2S3w94&list=PLPKNvFYGf41lOwuHY54XZa9MH9dV_4Gar, TensorFlow, Accessed 31 August 2020.

[37] Bently, Donald E, Charles T. Hatch, and Bob Grissom. "Fundamentals of Rotating Machinery Diagnostics". Asme, (2003),

[38] Eisenmann, Robert C, and Robert C. Eisenmann. "Machinery Malfunction Diagnosis and Correction: Vibration Analysis and Troubleshooting for the Process Industries". Upper Saddle River, N.J: Prentice Hall PTR, (1998).

[39] Aggarwal, Charu C." Neural Networks and Deep Learning: A Textbook". (2018).

[40] F. Rosenblatt. "The perceptron: a probabilistic model fot the information storage and organization in the brain", Psychological Review, (1958).

[41] "*The role of bias in Neural Networks*", pico.net/kb/the-role-of-bias-in-neural-networks#:~:text=Bias%20allows%20you%20to%20shift,transposed%20by%20the%20constant%20value., Accessed 31 August 2020.

[42] "*Backpropagation calculus | Deep learning, chapter 4*", 3Blue1Brown, https://www.youtube.com/watch?v=tIeHLnjs5U8&list=TLPQMTkwNzIwMjCYue_msIDu7A&index=1, Accessed 31 August 2020.

[43] "*Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization*", https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/, Accessed 31 August 2020.

[44] "*Proximity sensor*", https://en.wikipedia.org/wiki/Proximity_sensor, Accessed 31 August 2020.

[45] R. C. Eisenmann and H. Prentice, "Machinery Malfunction Diagnosis and Correction", (2005).

[46] Adams, Maurice L. "Rotating Machinery Vibration: From Analysis to Troubleshooting". New York: Dekker, (2001).

[47] Silva, Alejandro, Alejandro Zarzo, Juan M. Munoz-Guijosa, and Francesco Miniello. "Evaluation of the Continuous Wavelet Transform for Detection of Single-Point Rub in Aeroderivative Gas Turbines with Accelerometers." Sensors (basel, Switzerland). 18.6 (2018).

[48] Newmark, "N.M. A method of computation for structural machine", J. Eng. Mech. Div, (1959).

[49] Peters, Tim. "The Zen of Python". Python Enhancement Proposals. Python Software Foundation, (2004)

[50] "About Train, Validation and Test Sets in Machine Learning", https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7, Accessed 31 August 2020.

[51] Brkovic, Aleksandar, Dragoljub Gajic, Jovan Gligorijevic, Ivana Savic-Gajic, Olga Georgieva, and Gennaro S. Di. "Early Fault Detection and Diagnosis in Bearings for More Efficient Operation of Rotating Machinery." Energy. 136 (2017).

[52] Bengio, Yoshua. "Learning Deep Architectures for Ai" Hanover (2009).

[53] "Deep Neural Networks for Acoustic Modeling in Speech Recognition". (2012)

[54] "Bits of knowledge About Balancing Machines", https://www.shimadzu.com/industry/products/bm/0601.html, Accessed 31 August 2020.

[55] Muszynska, Agnieszka, Agnieszka Muszynska, Agnieszka Muszynska, and Agnieszka Muszynska. Rotordynamics. Boca Raton, Fla: Taylor Francis, (2005).

[56] "Neural Networks for Beginners: Popular Types and Applications", https:

//blog.statsbot.co/neural-networks-for-beginners-d99f2235efca, Accessed 31 August 2020.