

POLITECNICO DI TORINO

Master Degree in Mechatronic Engineering

Predictive Maintenance

A brain for the machine



Advisor

prof. Marcello Chiaberge

Candidate:

Claudio Filippo Vitiello

Company tutors

Prima Industrie S.p.A

Stefano Albrile

October 2018

ABSTRACT

Nowadays, in the industrial sector, one of the main aspects for which companies decide to invest most of their time and resources, to obtain ever more accurate results, is maintenance.

Maintenance refers to the ability to keep the machine in good condition, through a series of appropriate operations and controls. Normally, this process accompanies the whole life of the machine and, in some cases, it is difficult to find the best trade-off in the use of this tool.

In recent years, in a resource optimization process, the desire to have predictive maintenance is increasingly widespread: the goal, therefore, is to anticipate and predict any problems on the machine by performing the right operations in the right time to avoid the waste of time or serious failures.

Furthermore, this process assumes even greater importance if the context in which it was conceived is considered: Industry 4.0 or "4th industrial revolution".

This is where this thesis is placed: the general aim is to realize a predictive maintenance system able to detect and, consequently, to prevent unexpected failures of a laser cutting system by exploiting the analysis of the harmonic impulse response and elaborating the results with the use of pattern recognition models.

In this context, first and foremost, tools, resources and their use have been defined to conduct a feasibility study on the developed method (ie accelerometer and neural networks, as better described later), since - in the absence of similar examples in the literature - it was not possible to predetermine its validity in advance and, therefore, its suitability to produce results in line with the expectations.

The experimentation has been divided into two phases: the first one consists of doing a modal analysis on the machines following an impulsive force; the second one requires the use of pattern recognition models based, in the studied case, on neural networks, to perform a categorization of the examined signal, establishing if and, eventually, which anomaly is going to occur on the machine.

These topics will be dealt with in detail throughout the thesis structured in six chapters.

The chapter 1 contextualizes the theoretical problems of the project, presenting the notion of Industry 4.0 and the changes that will involve in the way of relating to the machine; the concept of maintenance will be defined, identifying the most

widespread methods including the predictive one; the modal analysis and its key points will be described; finally, the concept of machine learning, its characteristics and its potential will be illustrated.

In chapter 2, an overview of the neural networks will be provided, describing their behavior and the process that allows them to learn and understand situations. Finally, some parameters and functions produced by the network (ROC, confusion matrix) will be analyzed.

The chapter 3 describes the machine used during the tests, ie a 3D laser cutting machine. Its general structure as the fixed axes, the mobile ones and the head, in addition to the main components such as the numerical control (CNC), on which it is based, will be analyzed. This analysis is essential because it is useful to understand potential machine failures: two of these anomalies have been selected, based on criteria that are better detailed inside the chapter.

The chapter 4 is dedicated to the illustration of the various tools, functions and work environments used during the experimentation, with reference to the impulse generation used to excite the machine, the sensors used during the experimentation (accelerometer and encoder), the modalities and protocols of communication between accelerometer and PC, and MATLAB.

The chapter 5 is intended to the analysis of the two techniques used to build the system: the first one involves the use of an accelerometer, while the second one of an encoder whose signal has been transmitted to a virtual oscilloscope. All the operations performed to achieve the desired result will be described step by step: the data acquisition phase (using sensors), the data elaboration phase, the creation of the neural networks used to identify and categorize the different situations and, finally, the testing phase in which the neural networks have been further validated.

Finally, in chapter 6 the conclusions on the work will be drawn, going to examine the next phase of the project, listing some possibilities and future strategies.

List of Figures

1.1	Industry 4.0	13
1.2	Maintenance	15
1.3	Modal Analysis	16
1.4	Machine Learning	18
1.5	Basic structures of the supervised, unsupervised and reinforced learning	20
2.1	Artificial Neural Network	21
2.2	Biological Neural Networks	22
2.3	Structure of a basic Neural Network	23
2.4	Behavior of a neuron	23
2.5	Activation functions	25
2.6	Generic scheme Neural Network - backpropagation	26
2.7	Gradient descent	27
2.8	Backpropagation process	28
2.9	Confusion matrix - example	29
2.10	ROC - example	30
2.11	Neural Network - MATLAB example	32
3.1	Laser Next 1530	35
3.2	LN1530 - Machine module	36
3.3	LN1530 - front part	37
3.4	LN1530 - rear part	37
3.5	LN1530 - movable structure	38
3.6	LN1530 - Main components of the movable structure	39
3.7	LN1530 - optical chain	40
3.8	LN1530 - CNC	41
3.9	LN1530 - Cabin	42
3.10	LN1530 - Turntable module	43
3.11	Servo drive parameters - velocity loop gain	45
3.12	Servo drive parameters - position loop gain	45
3.13	Closing carriage anomaly - screws loose	46

4.1	CNC - pulse generation	48
4.2	Pulse characteristics	49
4.3	Accelerometer Sequoia	49
4.4	Accelerometer and Machine axes orientation	50
4.5	Error position on oscilloscope	51
4.6	Command file - example	51
4.7	File data.csv - example	52
4.8	Event capture - example	53
4.9	JavaApplication - execution	53
5.1	Project - main phases	55
5.2	CS1: Command File - Accelerometer	56
5.3	CS1: time responses - normal situation	57
5.4	CS1: time responses - servo drive anomaly	58
5.5	CS1: time responses - closing carriage anomaly	58
5.6	CS1: frequency responses - normal situation	59
5.7	CS1: frequency responses - servo drive anomaly	59
5.8	CS1: frequency responses - closing carriage anomaly	60
5.9	CS1: correct responses of more machines	61
5.10	CS1: anomalies responses - comparison	61
5.11	CS1 - MP: general scheme	62
5.12	CS1 - MP: Reference file	63
5.13	Neural Network - architecture	63
5.14	CS1- MP: Neural network dataset of first phase	64
5.15	CS1- MP: Confusion matrix and ROC of first phase	64
5.16	CS1- MP: Neural network dataset of second phase	65
5.17	CS1- MP: Confusion matrix and ROC of second phase	65
5.18	CS1- MP: Test	67
5.19	CS1- MP: Application	69
5.20	CS1- SP: general scheme	70
5.21	CS1- SP: Neural Network dataset	70
5.22	CS1- SP: Confusion matrix and ROC	71
5.23	CS1- SP: Test	72
5.24	CS2: Acceleration signal	74
5.25	CS2: time responses - normal condition	74
5.26	CS2: time responses - servodrive anomaly	75
5.27	CS2: time responses - carriage anomaly	75
5.28	CS2: frequency responses - normal condition	76
5.29	CS2: frequency responses - servo drive anomaly	76
5.30	CS2: frequency responses - carriage anomaly	77
5.31	CS2: correct responses of more machines	78
5.32	CS2: anomalies responses - comparison	78

5.33	CS2 - MP: Reference file	79
5.34	CS2 - MP: general scheme	79
5.35	CS2 - MP: Neural network dataset of first phase	80
5.36	CS2 - MP: Neural network dataset of second phase	80
5.37	CS2 - MP: Confusion matrix and ROC of first phase	81
5.38	CS2 - MP: Confusion matrix and ROC of second phase	81
5.39	CS2 - MP: Test	82

List of Tables

4.1	CNC comand	48
-----	----------------------	----

Contents

List of Figures	5
List of Tables	8
1 Introduction	11
1.1 Industrie 4.0	11
1.2 Maintenance	13
1.3 Modal analysis	15
1.4 Machine Learning	17
1.4.1 Supervised machine learning	18
1.4.2 Unsupervised machine learning	19
1.4.3 Reinforced machine learning	19
2 Neural Networks	21
2.1 Biological Neural Network	22
2.2 Structure	22
2.3 Activation function	24
2.4 Learning process - Backpropagation	25
2.5 Tools	28
2.6 In MATLAB	31
2.7 Pros and cons	32
3 Machine	35
3.1 Machine Module	36
3.1.1 Fixed Structure	36
3.1.2 Movable structure	37
3.2 Optical chain	40
3.3 CNC	40
3.4 Cabin and Turntable module	42
3.5 Problem analyzed	43

4	Instruments	47
4.1	Impulse generator	47
4.2	Accelerometer	49
4.3	Encoder - Oscilloscope	50
4.4	PC-Accelerometer connection	51
4.5	MATLAB	54
5	Method	55
5.1	Case study 1 : accelerometer	56
5.1.1	Data Analysis	61
5.1.2	Multi-phases	62
5.1.3	Single-phase	70
5.1.4	Conclusion	73
5.2	Case study 2 : encoder	74
5.2.1	Data analysis	78
5.2.2	Multi-phases	79
6	Conclusion	83
6.1	Next steps	83
A	JAVA application - code	85
A.1	JavaAccelerometer	85
A.2	Sequoia	88
A.3	SerialPortCommunication	89
A.4	CalibrationAccelerometer	95
A.5	AccelerometerComponents	97
A.6	Command	99
A.7	FileHelper	102
	Bibliography	105

Chapter 1

Introduction

This introduction provides a description of the industrial and social context in which this thesis is placed and the main thematic that it covers.

1.1 Industrie 4.0

In recent years, following the delicate economic situation that the world has faced, the companies started to experiment with new solutions to optimize resources and consequently reduce the waste of money. In particular, in the industrial sector, a set of innovative activities was conceived which took the name of Industria 4.0 during the 2011 Hannover fair [6].

This program is so vast and ambitious that it is quite difficult to predict the results and the impact these new solutions will have over time. Surely, it will lead to a benefit in all sectors, from industrial to social with important economic consequences: for this reasons, scholars defined this historical passage as "Fourth industrial revolution". In fact, briefly reviewing the various stages, the first was the steam engine, the second was the assembly chain, the third the robots while the latter in action wants to combine the physical systems with intelligence and data [34, 18].

The programs included in this project [4, 7] are:

- Advanced manufacturing solution: advanced production systems, ie interconnected and modular systems that allow flexibility and performance. These technologies include automatic material handling systems and advanced robotics such as collaborative robots.
- Additive manufacturing: additive manufacturing systems will allow increase the efficiency of the use of materials since the company will start to produce parts and small products inside. They are at the beginnning but to think to

produce part for their own machine at home, reducing the waiting cycle times, is no longer utopian.

- Augmented reality: vision systems with augmented reality to better guide operators in performing their daily activities. This allows a reduction of risk and a greater accuracy as the worker will be guided in each operation, indicating how and where to intervene [5].
- Simulation: simulation between interconnected machines to optimize processes.
- Horizontal and vertical integration: integration and exchange of information both horizontally and vertically, among all elements of the production process to make it even more efficient.
- Industrial internet of things (IoT): communication between elements of production, not only within the company but also outside it, thanks to the use of the internet.
- Cloud: implementation of all cloud technologies such as online information storage, the use of cloud computing, and external data analysis services. The cloud also covers methodologies for managing very large amounts of data through open systems. The final result is the conception of the industry at an increasingly virtual and abstract level [18].
- Cyber-security: the increase in internal and external interconnections opens the door to the whole issue of information security and systems that must not be altered from the outside.
- Big Data & Analytics: techniques to analyze and manage large amounts of data obtained from different sources for preventive and prediction scopes. In this way, the data assume more meaningful and importance since, using them, the machine can discover relationship connected to specific situations and create model to detect them [6].

Prior to Industry 4.0, some of these programs such as Big Data & Analytics, Autonomous Robots and Additive Manufacturing already existed but acted as independent disciplines. The goal, now, is to interact as many fields as possible with each other, and this will allow companies to take new paths in the way of producing bringing benefits for everyone [44].

Several applications, in as many branches, have been already developed using these new technologies and way of thinking: for example in public transportation sphere, new automated vehicles have been designed in such a way that, by means of sensors and AI, they are able to deal with the majority of the situations. Other

example, in industry field is, of course, predictive maintenance, the project described in this thesis. Generally, all these products have an increased reliability, flexibility, efficiency, since all manufacturing stages are optimized reducing waste and downtime, with the overall cost lowered.

However, like most innovations of this magnitude, there are not only advantages but also disadvantages, especially in the short term: the Industry 4.0 project was not welcomed by everyone in a positive way as came true the spectrum of the substitution of the man with an automaton, a delicate subject nowadays [2, 19]. In fact, it is undeniable that most of the mechanical and basic jobs, such as the transport of materials or the control of resources, are and will be carried out by robots, which will replace the human worker. On the other hand, these new technologies will bring with them the request of numerous softwriters able to program, analysts and people with specific skills able to develop new models. Moreover, looking at the theme of augmented reality, there will be the possibility to help the less experienced to perform all kinds of operations by giving them all the necessary instructions in real time.



Figure 1.1: Industry 4.0

1.2 Maintenance

Maintenance is one of the most delicate aspects within a company and in some cases leads to employ a number of resources, both economic and personal, not indifferent.

In fact, it plays an important role in the life cycle of a machine as it preserves or even increases its functionalities and, consequently, its value over time [33].

However, to accurately estimate how often the machine needs to be serviced is difficult: find the right compromise between suspending the production of the asset to perform maintenance tests to avoid breakdowns, with the risk of having interrupted the process unnecessarily and waiting for the failure to happen with the consequent replacement of the piece at high costs and times, is the most difficult challenge that faces a company[8].

Nowadays, there are three methodologies to do maintenance[17]:

- **Reactive maintenance:** also known as "breakdown maintenance" requires the intervention of the technician only after the malfunction of the resource, such as the breaking of a piece. In this case the keyword is repair and, therefore, bringing the resource back to its normal working condition. It is the easiest technique to implement but not very efficient: if among the analyzed techniques it requires the least number of people involved, the time and the cost taken for repair operations, with the product not available, determine important avoidable expenses using other methodologies. Moreover, another aspect should not be underestimated are the consequences deriving from these malfunctions: if in some cases the damages are only of an economic nature, more delicate cases can lead to far more serious consequences since can implicate human life. Therefore, the need to find new solutions of maintenance is increased.
- **Preventive maintenance:** also called "planned maintenance" or "scheduled maintenance", is one of the most widespread techniques and aims to maintain the correct status of the resource by carrying out periodic checks to detect anomalies. The advantage of this method is to avoid the event of the failure with all the consequences of the case such as the cost of repair and interruption of the production process. However, the criticality lies in establishing the periodization of controls: this parameter is often chosen high as a precautionary measure and it may unnecessarily slow down the production process, involving unnecessary interventions on pieces that are still able to do their job. This problem has been solved partly by performing monitoring and analysis activities, obtaining statistical data with which it is possible to establish with greater accuracy the choice of this parameter without losing the reliability of the method. A common example is the revision of the machine, which must be performed periodically.
- **Predictive maintenance:** it is one of the cornerstones of the Industria 4.0 project as it seeks through the analysis of data, obtained in the Big Data & Analysis and IoT (Internet of Things) context [40, 38], to establish in what condition the resource is and will be. It aims to carry out targeted actions in order

to prevent any malfunctioning of the resource, only when necessary. It is a complex and innovative process because it allows to build relation between data obtained from different sources and instruments (such as acceleration, temperature, machine cycle time...) using specific models, obtained for example with machine learning. Undoubtedly, it represents the answer to the problems found in the previous cases as it anticipates not only the event of the failure but avoids early and useless interventions. Indeed, the advantages deriving from the use of this methodology are indisputable: lower labor costs, increased reliability, risk mitigation, increased equipment life. On the other hand, it is the technique that surely requires a greater transversal knowledge of the problems and of the argument, as well as a request for data in the following that must be as high as possible to obtain more and more accurate models.

In conclusion, it is clear that from several point of view the predictive maintenance technique is the one that brings greater benefits: in fact, considering only the economic field [21], as shown in the graph (1.2), even if the maintenance costs deriving from the predictive technique are greater than the other cases, the waste of production time is rather limited, leading to a much lower total cost compared to the other two cases.

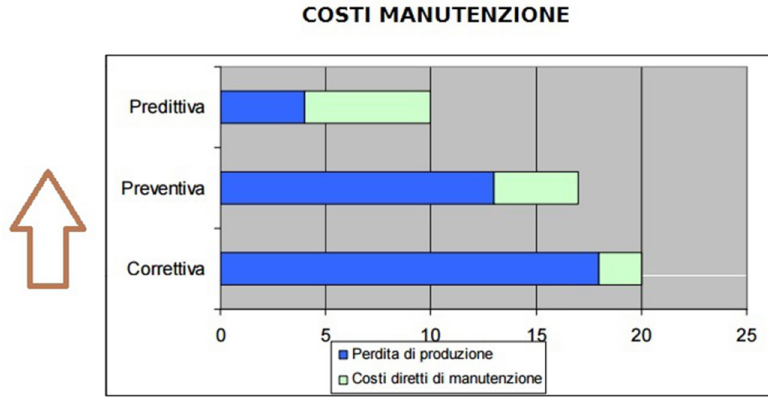


Figure 1.2: Cost of Maintenance [21]

1.3 Modal analysis

Modal analysis is the study of the dynamic behavior of a structure when it is subjected to vibration[29]. It is based on the study of the natural frequency and

of the proper ways of vibrating associated with the various components of the structure, as a result of a forcing. To date, to conduct this analysis, it is required:

- the use of sensors, such as accelerometers, encoders.
- a data acquisition system: it is often required an analog-digital converter front-end to digitize analog instrumentation signals.
- PC host (personal computer) to display and analyze data.

In principle, the main steps of a modal analysis are:

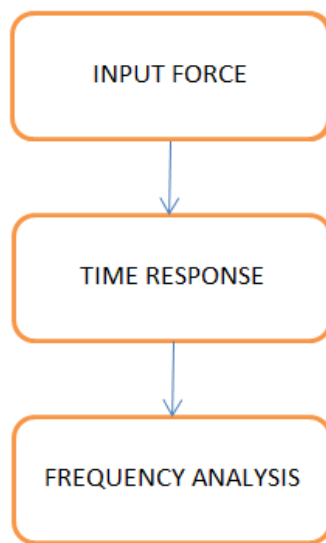


Figure 1.3: Key points of Modal Analysis

The analysis can be conducted in different ways[16]:

- SISO (single-input, single-output): the structure is excited at one point and the response is collected by a single sensor placed in a single position.
- SIMO (single-input, multiple-output): the structure is excited at one point and the response is collected by sensors positioned at different points.
- MISO (multiple-input, single-output): the most common analysis, when the machine was excited by an instrument, like a hammer. A fixed accelerometer was placed to acquire the signal following hammering applied at different points.
- MIMO (multi-input, multiple-output): it may be possible to associate the response with its source following a good analysis.

The most used excitation signals are pulse, square wave, swept sine, chirp.

One of the most common applications about modal analysis is maintenance[15]. Doing this kind of analysis is comparable to "listen" the inside of the machine and, so, it allows to understand what happens. Each component vibrates in its own way, generating a characteristic noise, which is even more highlighted in the spectrum. If damage is present, this characteristic is distinguished from background noise. The specialist is, thus, able to identify the problem and distinguish the kind of it: for example, it is possible to detect an imbalance, a misalignment or a damage to the bearings. Moreover, in a view of predictive maintenance, it is also possible carry out this diagnosis with an estimation how serious the problem is, in order to establish whether it is necessary to act quickly.

1.4 Machine Learning

Over time, the primary objective of human being is to create structures able to simplify the situations they face [27, 26]. Hence the desire to deepen the issue of machine learning: as the name suggests, it consists of training and educating the instrument to recognize certain situations by making it "think" like a human being.

This is possible because this tool is able to find correlation and connection between the tested data, returning in the output the desired result.

The strong point is that even if the analyzed data come out from different sources, as acceleration and temperature, the model is still able to find relationships, extrapolating only the informations that it considers most important, through a weight system, providing accurate results in minimum time. Considering this, it is quite evident the potential of this instrument: suffice it to say that, although the human brain is able to solve most situations in a few milliseconds, the performance that can reach the computer is of a higher order of speed (microseconds).

This tool is used in various disciplines, from medicine to psychology, but it is in the scientific field that is providing more applications [32]:

- security heuristics that attack patterns to protect, for instance, ports or networks;
- image analysis to identify distinct forms and shapes, such as medical analyses and face and fingerprint recognition;
- deep learning to generate rules for data analytics and big data handling, such as marketing and sales promotions;
- object recognition and predictions from combined video streams and multisensor fusion for autonomous driving;

Going into detail, there are three major categories of machine learning: supervised control, unsupervised control and reinforced control [28, 24, 20].

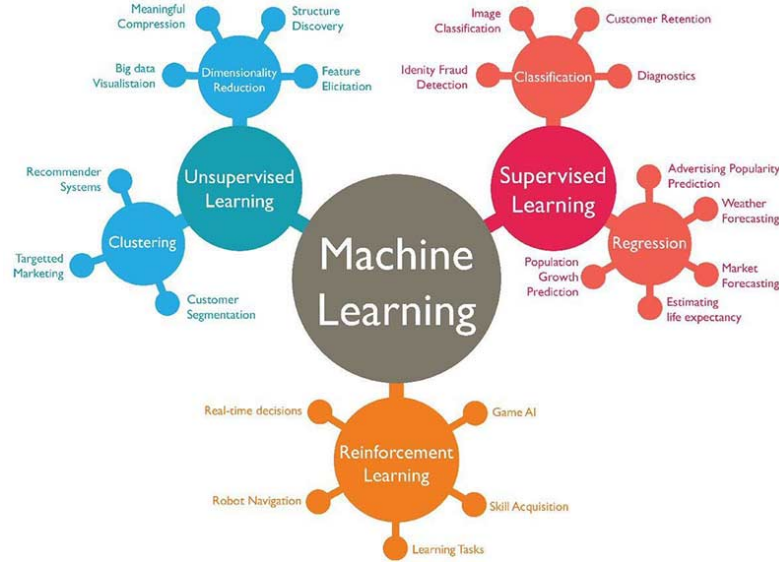


Figure 1.4: Machine Learning [43]

1.4.1 Supervised machine learning

Both the inputs and the correct outputs are provided by the user. In this case the machine will have to find the correlations between input and output in order to develop a behavior that responds adequately to the future data it will receive [32].

So, the main point here is to create a large enough database with all informations, data and experiences in such a way the system has simply to draw on it to give the right solution to the problem. Therefore, this type of machine learning is, in some way, delivered already packaged and the machine must only be able to choose which is the best response to the stimulus given to it.

Supervised control includes two main sub-categories:

- Classification[11]: in the training phase, to each incoming data is associated a class to which it belongs. Then, it will be up to the machine to associate future data in the right category.
- Regression: in this case the machine, using the input data and the respective outputs, must identify the correct trend in order to provide a suitable result for the subsequent inputs.

In this thesis, with a view to obtain a classification model, this kind of learning has been used, by means of neural networks whose characteristics and functionalities are described in the next chapter.

1.4.2 Unsupervised machine learning

In this case the user only provides input data. The machine looks for common characteristics in the given signals categorizing and organizing them, learning their importance in order to produce a reasonable results.

Thus, the machine uses informations and data, without having any example of their use and, consequently, no knowledge about the desired output. This type of machine learning has more degree of freedom with respect to the previous case since, autonomously, the machine has to organize the informations in a smarter way and learn which are the best results according to the different situation that arise.

The unsupervised control provides two sub-categories:

- Clustering: it is a process similar to classification. In this case, not being provided the outputs with which to associate the input, the task of the machine will be to find affinity between the data in order to identify the classes.
- Reduction Data set: it is a process that allows to identify which data, among those under examination, are of little relevance for the purposes of classification and, consequently, can be discarded from the discussion.

1.4.3 Reinforced machine learning

The reinforced machine learning is the most complex among all them. The machine or the software agent interacts with the surrounding environment and learns its behavior on the basis of "trial and error" procedure using feedback from its own actions and experiences [36].

In this case, in addition to the machine, auxiliary elements, as sensors and cameras, are used to better understand and detect what happens in the surrounding environment allowing to the system to make the best choices possible in each situation.

The most common applications that use this technique are videogames, self-driving cars and in the field of robotics when an efficient adaptive control system is desired.

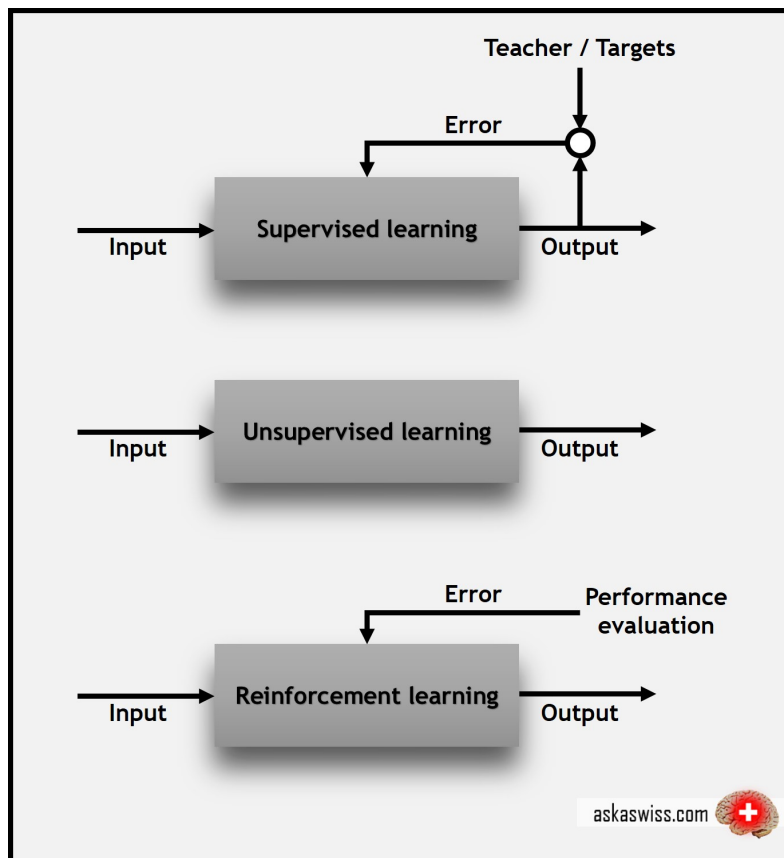


Figure 1.5: Basic structures of the three learning paradigms: supervised, unsupervised and reinforced learning[20]

Chapter 2

Neural Networks

The artificial neural network (ANN), also known simply as a neural network, is one of the models of machine learning[10]. As the name suggests, its behavior recalls that one of biological neural networks: inspired by them, they possess data processing units, artificial neurons, which actually perform the same task as biological neurons do in the brain. Therefore, each unit is linked to the other in order to create a real network of interconnections that allows the model created to interact with the outside world like the brain of a human being.

For this reason, nowadays, they represent the foundation of artificial intelligence: there are many applications that already involve the use of neural networks, including google translate, the detection of spam in e-mail and the facial recognition.

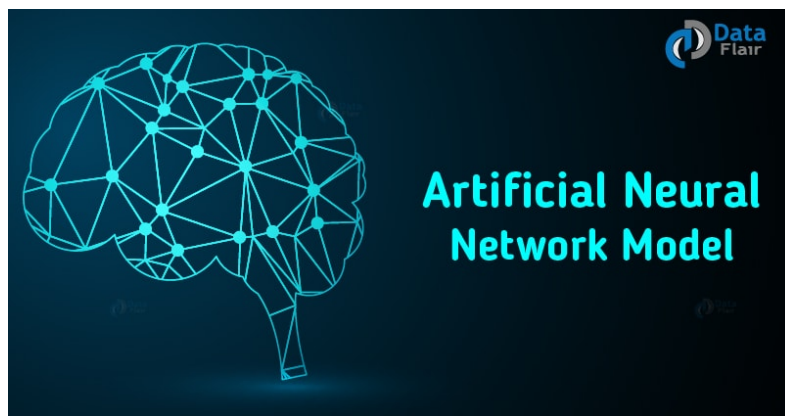
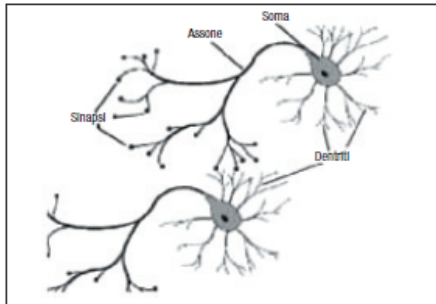


Figure 2.1: Artificial Neural Network [41]

2.1 Biological Neural Network

Before proceeding with the description of the ANN, a briefly introduction on how the biological neural networks work is given[22].

The human brain is composed of billion of unit called neurons. Each of them receives as input the electrical signals from all the dendrites, elaborates the information in the soma and, if the sum weighing exceeds the value of activation, emits a pulse electric output towards the axon. Then, the neurons are linked each other in a contact points called synaptic contact. Here the electric signal is trasformed in chemical substance (neurotransmitter) before returning electric once it is passed into the other part. The synapse is in charge of exciting or inhibiting connections between the two neurons by varying neurotransmitters.



(a) Dendrites, soma and axon [22]



(b) Synaptic contact [1]

Figure 2.2: Biological Neural Networks

2.2 Structure

The artificial neural network, in the same way, has units, called artificial neurons, which are grouped in three stages [42]:

- Input layer: layer in which the inputs, used to train the neural network, are provided. It is comparable to dendrites.
- Hidden layers: internal layers (H - hidden), introduced in 1986 by David Rumerlhart, which are responsible for processing and transmitting data. Their introduction allowed the elaboration of more complex learning models as the Multi-Layers Perceptron (MLP) networks. They do the same job of the soma.
- Output layer: layer in which the neural network transmits the final result. For this reason, it refers back to the axon.

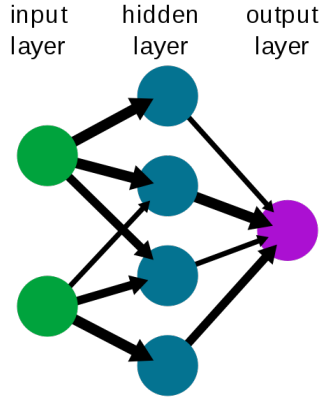


Figure 2.3: Structure of a basic Neural Network [1]

Each neuron is typically connected to all neurons of the next layer by weighted connections. To each connection, a numerical value (*weight*) is associated, which is multiplied by the value of the connected neuron.

After that, each neuron sums the weighted values of all neurons of the previous layer connected to it and adds to the total a value that takes the name of bias. Therefore, the equation related to a generic neuron is:

$$net_i = \sum_{j=1}^N x_j * w_j^i + b_i \quad (2.1)$$

where N is the amount of neurons linked to this one.

Finally, an activation function is applied to this result, which normalizes the output in a range of values that varies according to the chosen one, before passing it to the next layer:

$$y_i = f(net_i) = f\left(\sum_{j=1}^N x_j * w_j^i + b_i\right) \quad (2.2)$$

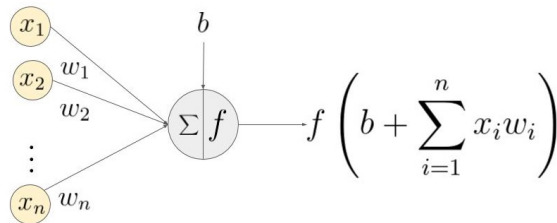


Figure 2.4: Behavior of a neuron

This process is propagated between all stages until output layer, where a final result is provided.

2.3 Activation function

The activation function is used to decide the shape of the output of a neuron. It can be both linear and non-linear and, as mentioned previously, it also has a normalizing effect on the neuron output: this last effect avoids that the output of neurons, after several layers, becomes very large due to the cascading effect.

The most used activation functions are:

- Heaviside or unit step: the simplest one, if the input is greater or equal to 0, then the output will be 1; viceversa, the output will be 0.

$$f(net_i) = \begin{cases} 0 & \text{if } net_i < 0 \\ 1 & \text{if } net_i \geq 0 \end{cases}$$

- rectifier linear unit (ReLU): only positive values are passed by the function. Thus, the negative values become 0, while the positive ones are mapped in a linear way.

$$f(net_i) = \begin{cases} 0 & \text{if } net_i < 0 \\ net_i & \text{if } net_i \geq 0 \end{cases}$$

- sigmoid: non-linear function that converts the inputs into a range between 0 and 1, in a way illustrated by the below equation.

$$f(net_i) = \frac{1}{1 - e^{-net_i}} \quad (2.3)$$

- tanh: the hyperbolic tangent has a similar shape of the sigmoid function, but it converts the input between -1 and 1.

$$f(net_i) = \frac{e^{net_i} - e^{-net_i}}{e^{net_i} + e^{-net_i}} \quad (2.4)$$

- softmax: function used in the probability field. It calculates the probabilities distribution of the event over ‘n’ different events. So, this function will calculate the probabilities of each target class over all possible target classes. Therefore, the calculated probabilities will be helpful for determining the target class for the given inputs.

The main advantage of using Softmax is the output probabilities range: the

value is between 0 and 1, and the sum of all the probabilities is equal to one. If the softmax function is used for multi-classification model, it returns the probabilities of each class and the target class will have the highest probability.

$$f(net_i) = \frac{e^{net_i}}{\sum_{j=1}^N e^{net_j}} \quad (2.5)$$

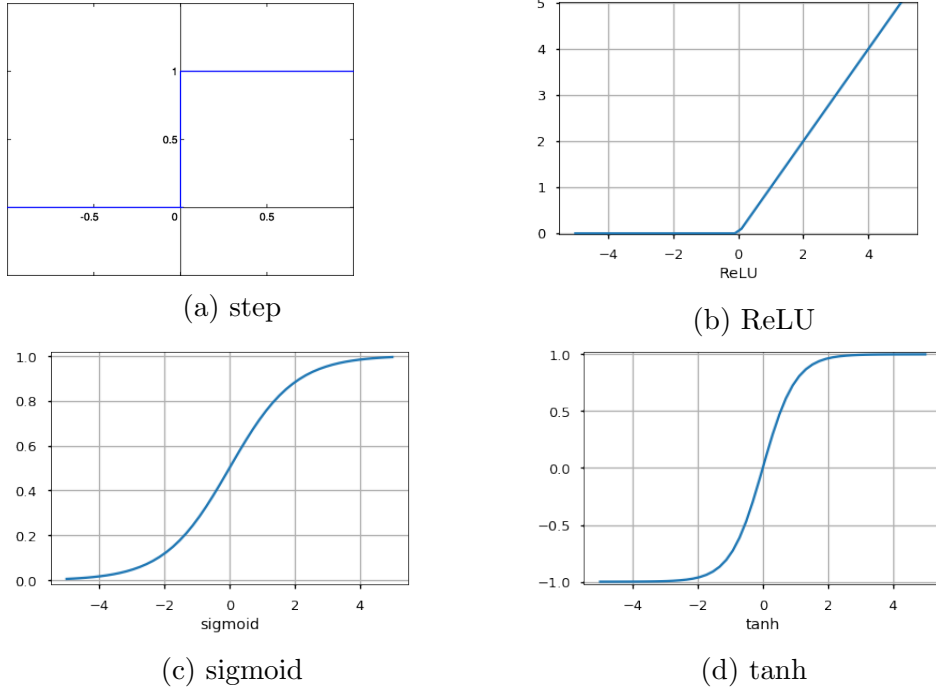


Figure 2.5: Activation functions

2.4 Learning process - Backpropagation

In this section, the most common way to learn the feedforward neural network will be analyzed: the Backpropagation algorithm [3, 31, 35].

The Backpropagation algorithm is used to set the weights of a multilayer neural network with a fixed architecture. It performs gradient descent to try to minimize the sum squared error between the network's output values and the given target values, called cost or loss function, going to modifying the weights.

To understand the process, let us recall the following parameters:

- w_{kj} is a generic weight from the hidden(j) to the output layer(k).

- w_{ji} is a generic weight from the input(i) to the hidden layer(j).
- y is the actual output.
- t is the target output.
- net is the net input.

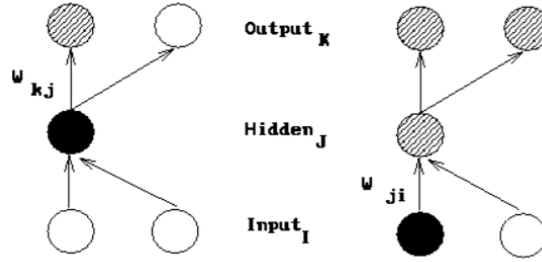


Figure 2.6: Generic scheme Neural Network - backpropagation

As mentioned above, the sum of all errors is described analitically by the cost function:

$$C = \frac{1}{2} \sum_k (y_k - t_k)^2 \quad (2.6)$$

The aim is to reduce as minimum as possible the value of this function. For this purpose, the only parameter able to do it is the actual output. Going more deeply, it appears that each output depends on weights and more accurate results can be obtained changing and updating these values:

$$\Delta W = W_{new} - W_{old} = -\gamma * \frac{\partial C}{\partial W} \quad (2.7)$$

where γ represents the learning rate, introduced to define the length of variation step of the weights.

At this point, it is important to note that the change to a hidden to output weight (w_{kj}) depends on error at the output node and activation at the hidden node, while the change to a input to hidden weight (w_{ji}) depends on error at the hidden node (which in turn depends on error at all the output nodes) and activation at the input node.

Let us start to analyze the first case.

$$\Delta w_{kj} = w_{kj_{new}} - w_{kj_{old}} = -\gamma * \frac{\partial C}{\partial w_{kj}} \quad (2.8)$$

Then, using the chain rule the following relation is obtained:

$$\frac{\partial C}{\partial w_{kj}} = \frac{\partial C}{\partial y_k} * \frac{\partial y_k}{\partial net_k} * \frac{\partial net_k}{\partial w_{kj}} \quad (2.9)$$

Performing all computation and substituting in equation (2.8), the new values of weights are found:

$$w_{kj_{new}} = w_{kj_{old}} - \gamma * (y_k - t_k) * [y_k * (1 - y_k)] * y_j \quad (2.10)$$

Finally, it is possible to determine the weight change for an input to hidden weight. This is a bit more complicated because it depends on the error in all of the nodes this weighted connection can lead to.

$$\Delta w_{ji} = w_{ji_{new}} - w_{ji_{old}} = -\gamma * \frac{\partial C}{\partial w_{ji}} \quad (2.11)$$

$$\frac{\partial C}{\partial w_{ji}} = \left[\sum_k \frac{\partial C}{\partial y_k} * \frac{\partial y_k}{\partial net_k} * \frac{\partial net_k}{\partial y_j} \right] * \frac{\partial y_j}{\partial net_i} * \frac{\partial net_i}{\partial w_{ji}} \quad (2.12)$$

Now, substituting the results obtained from above equation back into our original one(2.11), the new values of weights are:

$$w_{ji_{new}} = w_{ji_{old}} - \gamma * \left[\sum_k (y_k - t_k) * y_k * (1 - y_k) * w_{kj} \right] * y_j * (1 - y_j) * y_i \quad (2.13)$$

This process is iterated until the variation of the cost function are so small that is meaningless continue to update the weight: it means a minimum local of the function has been found.

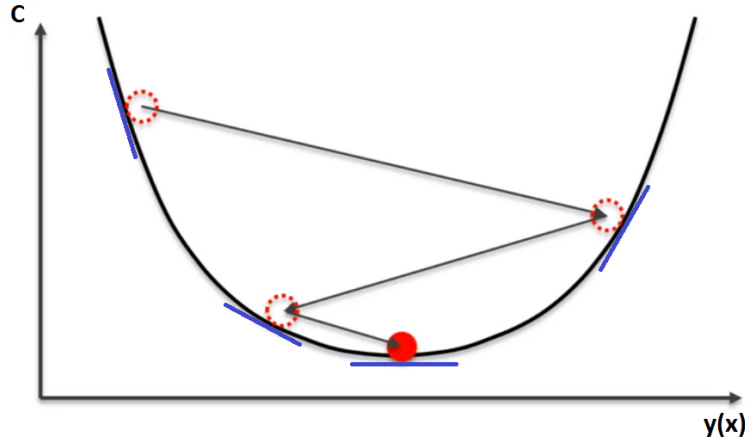


Figure 2.7: Gradient descent [30]

To summarize, here is what the learning process on neural networks looks like:

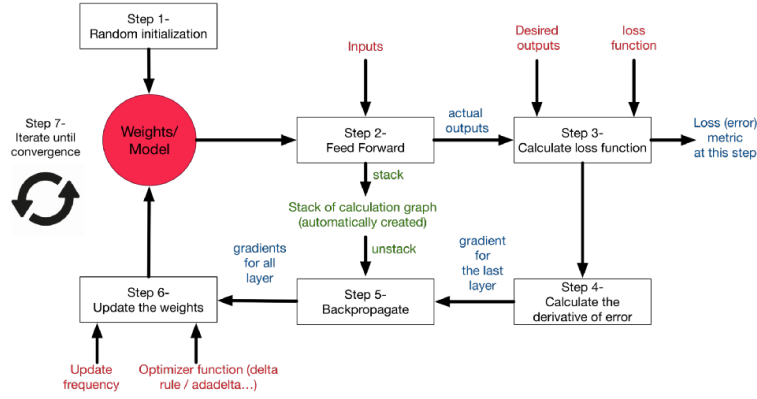


Figure 2.8: Backpropagation process [3]

2.5 Tools

Some tools and graphs such as the Confusion matrix and the Receiver Operating Characteristic(ROC) [39] can be used to verify the accuracy of the obtained classifier:

- Confusion matrix: as the name suggests, it is a matrix which columns indicate the instance of the actual outputs - the outputs that neural network would consider - while the rows indicate the target classes. When the data is processed, if it is listed in the correct classification, then both actual and target class are the same; otherwise, when a mismatch occurs, the two classes will be different and it means the neural network could not place data in the right class. On the basis of this, it is possible to understand which problems the model is not able to identify and whether it may need to do a new training to solve the problem.

Considering the example of figure (2.9), 444 and 238 data that belong, respectively, to class 1 and 2 have been placed correctly while 3 data that should belong to class 2 have been wrongly considered in class 1 and, viceversa, 14 data of class 1 have been collocated in class 2.

- ROC: it is another tool used to check the quality of classifiers. To better understand how this tool works, it is better to start examining the easiest case, ie the binary (two-classes) problem. In addition, these new terminologies are introduced:
 - True positive (TP): data that should belong to class i are classified correctly in class i.
 - True negative (TN): data that should not belong to class i are not classified in class i.

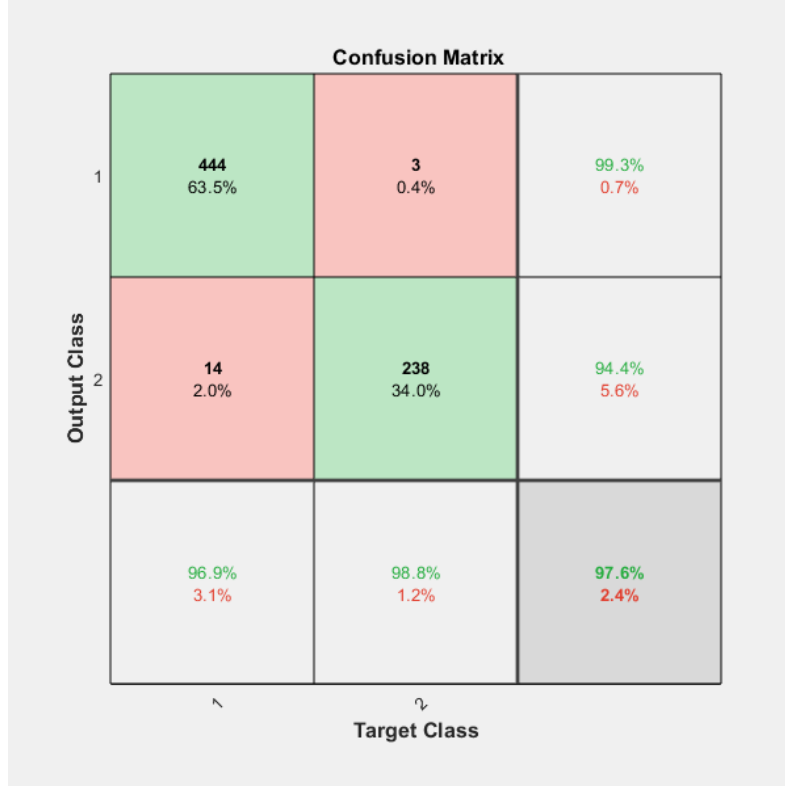


Figure 2.9: Confusion matrix - example [25]

- False positive (FP): data that should not belong to class i are classified wrongly in class i .
- False negative (FN): data that should belong to class i are not classified in class i .

For each class, the classifier computes other quantities, as the sensitivity and specificity [12]. The sensitivity is the ratio between number of outputs whose actual and predicted class is class i divided by the number of outputs whose predicted class is i :

$$sensitivity = \frac{TP}{TP + FN} \quad (2.14)$$

The specificity, indeed, is the ratio between number of data whose actual and target class is not class i and all data whose target class is not class i .

$$specificity = \frac{TN}{TN + FP} \quad (2.15)$$

For each class of a classifier, ROC applies threshold values across the interval $[0,1]$ to outputs. For each threshold, two values are calculated, the True Positive Ratio (TPR) and the False Positive Ratio (FPR). Taking into account a specific class i , TPR is the sensitivity while FPR is $(1 - \text{specificity})$, ie the number of outputs whose target class is not class i , but the actual one is class i , divided by the number of outputs whose target class is not class i :

$$TPR = \text{sensitivity} = \frac{TP}{TP + FN} \quad (2.16)$$

$$FPR = 1 - \text{specificity} = \frac{FP}{TN + FP} \quad (2.17)$$

To pass from binary case to multiclass case, the easiest way is to reduce the multiclass problem to multiple binary classification problems [14]: in this way, each class is compared to all others. Then, the ROC defines a bidimensional space in which the TPR represents the x-axis while the FPR the y-axis. To measure the percentage of accuracy of the classifier, it is necessary to calculate the area under the curve and multiply the result by 100.

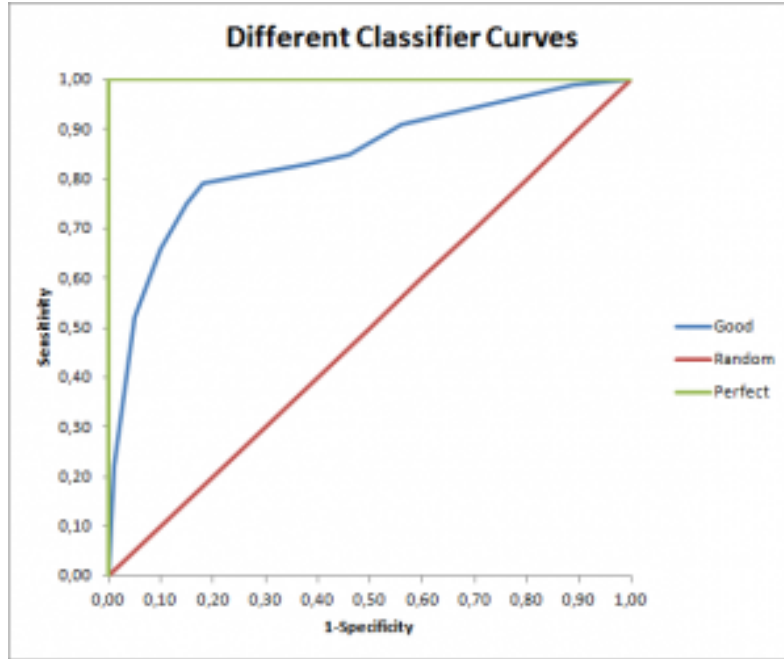


Figure 2.10: ROC - example [13]

From the figure (2.10), it emerges the more the classifier is accurate, more the curve tends to the upper left corner, corresponding to have an high sensitivity and an high specificity. If a point has $(0,1)$ coordinates, it means a perfect accuracy of the classifier.

2.6 In MATLAB

The Neural Network can be implemented using different environment. Here, it will be described how to generate a neural network using the MATLAB environment, by means of the Neural Network Toolbox [9].

First of all, according to the target, four different types of neural network are proposed:

- **Fit Data:** used to describe fitting problems, in which the neural network has to map between a data set of numeric inputs and a set of numeric targets.
- **Pattern Recognition:** used for pattern recognition problems, the neural network classifies inputs into a set of target classes.
- **Time Series analysis:** used in the field of prediction, it uses past values of one or more time series to predict future values.
- **Cluster Data:** in this kind of problems, the neural network is used to group data having similar characteristics.

For my purpose, the Pattern Recognition one has been selected. It has the following characteristics:

- **Two-layer feedforward network:** the outputs always proceed in one direction, ie forward, and they never create a cycle;
- **Sigmoid activation function** in the hidden layer;
- **Softmax activation function** in the output layer;
- **Number of network inputs = number of problem inputs;**
- **Number of neurons in output layer = number of problem outputs;**
- **Number of neurons in the hidden layer:** there is no way to know a-priori this parameter. Anyway, it is suggested to stay near to the value, using a "try and error" procedure, obtained by the following equation:

$$N_h = \sqrt{N_i * N_o} \quad (2.18)$$

where N_i is the number of network inputs and N_o is the number of neurons in the output layer. The choice of this parameter is very important: if it is too small, the classifier is not able to establish the right correlation and give the right importance to data and, consequently, does not provide the appropriate results; in the other case, the overfitting problem ([37]) can occurs. In this latter case, the classifier becomes too much sensible: subsequently all small variations and noise play an important role in the outgoings causing a loss in the model's accuracy and validity.

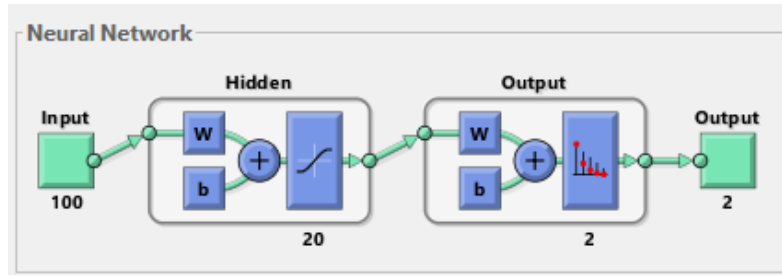


Figure 2.11: Neural Network implemented in MATLAB - Pattern Recognition example

Then, the user has to import the dataset (inputs and outputs - supervised learning) used. It is splitted by the classifier: the 70% is used for training, the 15% for validating the network and stop training before overfitting, and the remaining 15% for the test phase.

Finally, the classifier starts the learning process based on scaled conjugate gradient backpropagation method.

Once it ends, its accuracy is valuated using the Confusion matrix and the ROC: if it satisfies the user, it is possible to export the correspondent code in MATLAB. In the other case, it is possible to retrain the model modifying, if necessary, the number of hidden neurons or changing the percentage of division of data until a good solution is found.

2.7 Pros and cons

It is important that the user knows the potentiality and the limits of a neural network [23].

The pros are:

- Neural networks are able to solve problems that are difficult to figure out using other algorithms.
- They produce good results even if data are complex, imprecise or subject to noise.
- They are easy to implements.

The cons are:

- The user does not know in which way the net finds the solution (black-box).
- It is necessary to have a large dataset to have a good learning process and a low output error.

- It is not possible to know a-priori whether the problem will be solved: for example, it is difficult to find a solution with inputs belonging to an high number of categories.

Chapter 3

Machine

In order to better understand the problem analyzed during the thesis, a general overview is given of the main elements of the machine: useful for understanding which problems and anomalies can damage it.

In particular, the analysis will focus on the type of machine used during the tests, ie the 3-D laser cutting machine and even more specifically the Laser Next 1530 (LN1530). It is the symbol and spearhead of the company and one of the most demanded by the costumers, thanks to its perfromances and characteristics. The term *1530* stands for its dimension: *15* means 1.5m of maximum extension reached by the Y axis, while *30* means 3.0m of maximum extension of X axis.

The machine has the following functional groups:

1. Machine module
2. Optical Chain
3. CNC
4. Cabin and Turntable module



Figure 3.1: Laser Next 1530

3.1 Machine Module

As undelined by the figure (3.2), the machine module is composed by the following components:

1. Fixed structure
2. Movable structure
3. Head

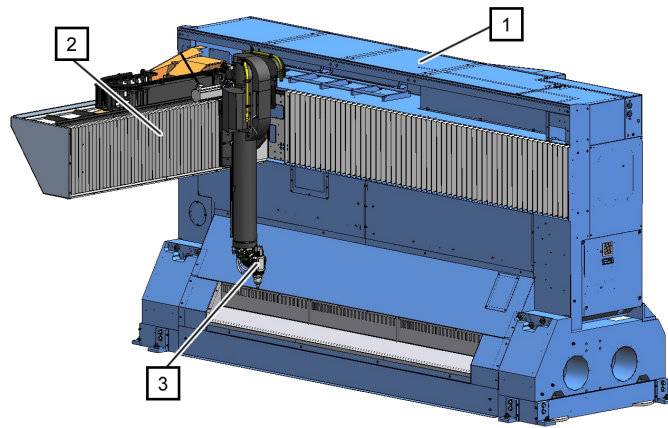


Figure 3.2: LN1530 - Machine module

3.1.1 Fixed Structure

The main component of the fixed structure is the base (1) made of synthetic granite, which guarantees excellent absorption of the vibrations and optimal thermal stability. The base is fastened on the floor through the abutments (2) and the anchoring brackets (3).

The front part of the base has a housing for the movement system of the X carriage (4) and the exhaust hood (5). The movement system of the X carriage is protected by bellows (6).

The X carriage movement system (4) allows the movement of the carriage unit along the X axis. The movement is achieved by using a linear motor and re-circulating ball bearing guides. The transduction of the position is carried out through an optical line included in the guide and a reading head connected to a sliding block. In the upper part of the base there is the cable holder chain (7).

In the rear part of the base there are: the pneumatic panel (7), the air treatment unit (air dryer and filtering) (8), the automatic lubrication system (9), the

electromechanical cabinet (10), the chiller (11) and the water systems panel (12). The laser support (13), used during the machine transport phases, is fixed on the right side of the base.

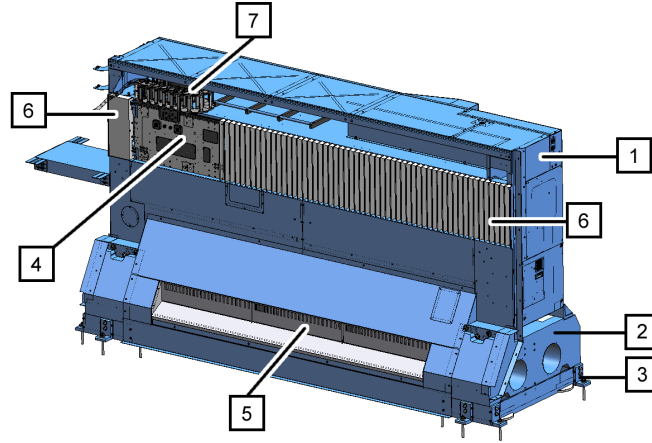


Figure 3.3: LN1530 - front part

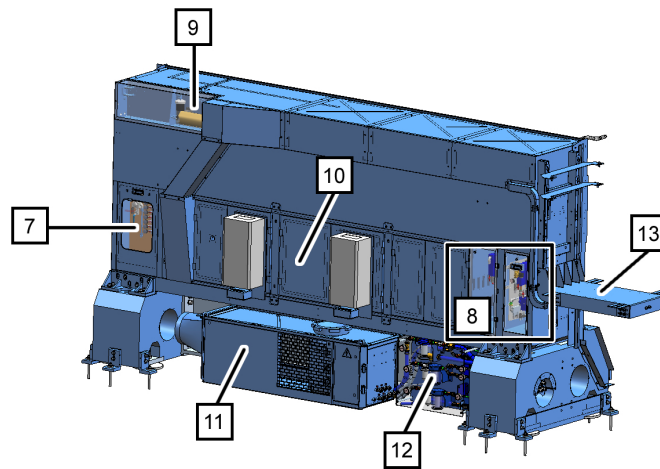


Figure 3.4: LN1530 - rear part

3.1.2 Movable structure

The mobile structure (1) consists of the following components:

- X,Y carriage
- Z carriage

- Cutting head

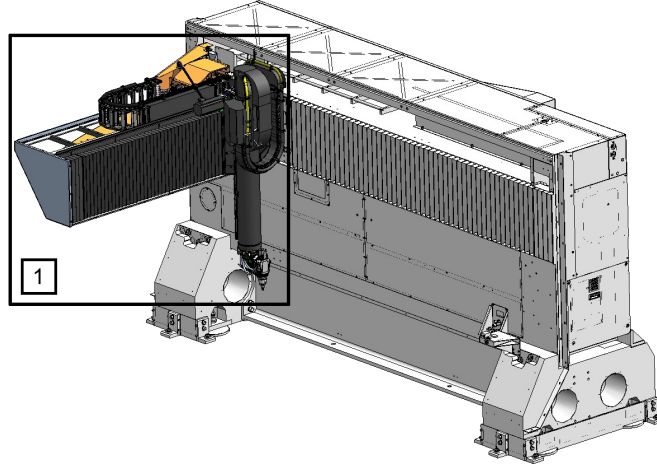


Figure 3.5: LN1530 - movable structure

X,Y carriage X carriage (1) is fastened to the interface plate by means of the hinge (2) and the four screws. The hinge (2) allows to fold back the carriage unit when shipping and moving the machine.

The Y carriage is fitted with two external bellows (3) for the recirculating ball slide guides protection and with two inner bellows (4) for the linear motor and optical line protection. During machine operation the bellows (3) and (4) are closed and hooked to the Y carriage. They are free to move, sliding on specific guides along the Y axis, adapting to the movement of the Y carriage. The electrical box (5) and the overhead cable tray (6), on which is fastened also the delivery fibre (7) are installed on the top of the carriage.

The front part of the X carriage includes the housing for the Y carriage movement system that is fastened to the X carriage through four sliding blocks.

The Y carriage movement system consists of two recirculating ball slide guides and a linear motor. The transduction of the position takes place by means of an optical line included in the guide and a reading head connected to a sliding block.

Z carriage The Z carriage (1) slides along the Z axis thanks to the guides (2) assembled on it. The related sliding blocks are assembled on the Y Carriage. The movement of the axis is made by a geared motor (3) on which is keyed to a pinion with helical teeth which engages on a rack fixed to the Z carriage. The braking of the axis is carried out by means of a brake assembled inside the motor and two pneumatic brakes assembled on the guides. On the front part of the Z carriage there is the cable holder chain (4) on which is fixed also the delivery fiber (5).

Cutting head The main features are:

- A axis rotation = 360° (continuous)
- B axis rotation = $\pm 135^\circ$
- C axis stroke = $\pm 12\text{mm}$
- focal lengths = 5"
- capacitive sensor: it keeps constant the distance between the tip and piece surface.

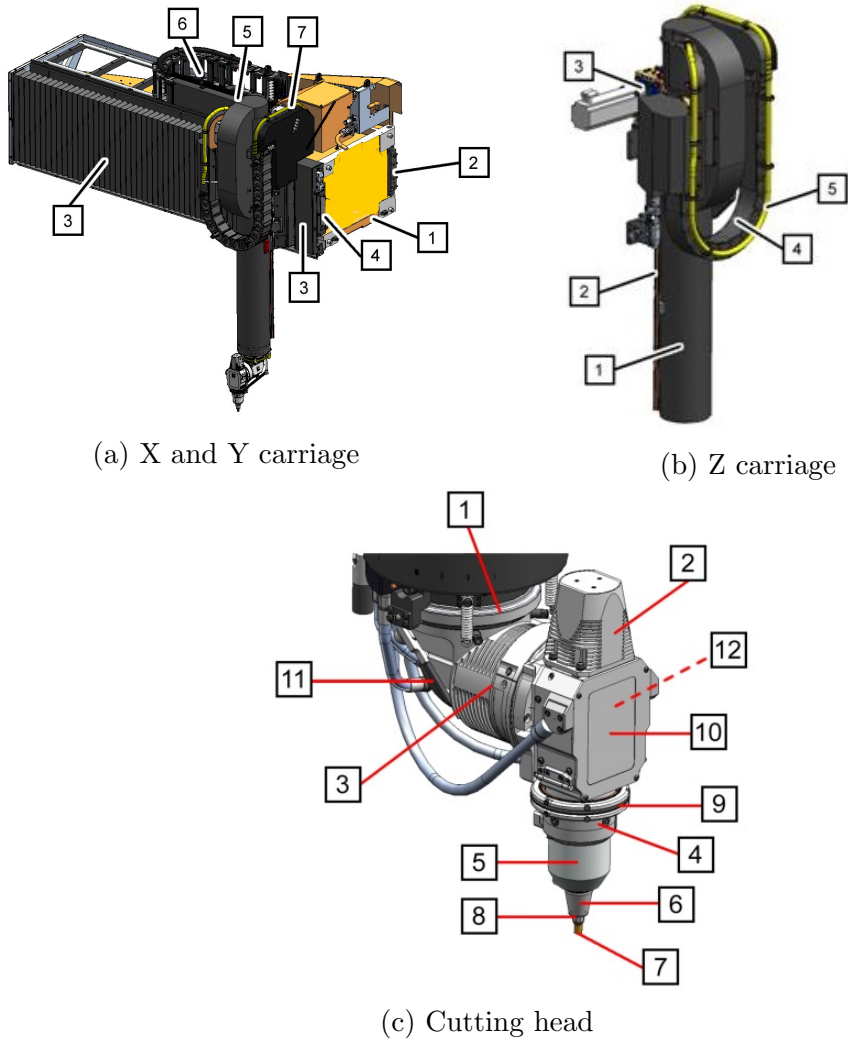


Figure 3.6: LN1530 - Main components of the movable structure

3.2 Optical chain

The optical path of the machine is constituted by an optical fibre cable (A) which guides the laser beam from the generator, through the X chain system (B), the Y chain system (C) and the Z chain system (D) to the collimator (E). From the collimator, the laser beam (F), through the mirrors of the head (G), is sent to the focusing lens (H).

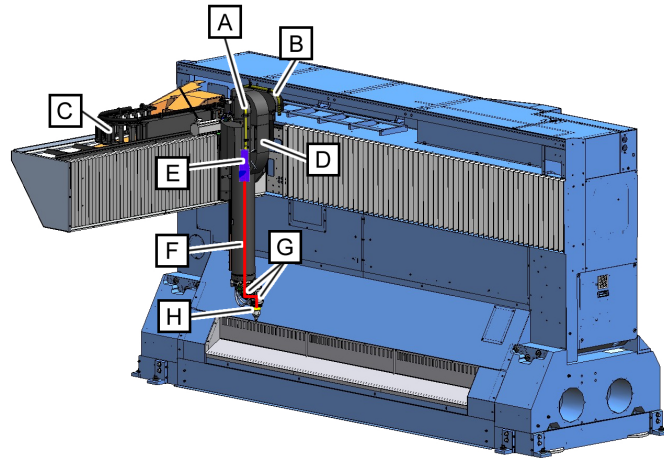


Figure 3.7: LN1530 - optical chain

3.3 CNC

The Prima Power Laser Next is equipped with the latest version of the P30L numerical control with 17" touch screen and integrated Human Machine Interface (HMI). The P30L numerical control is developed and manufactured by Prima Electro, a company of the Prima Industrie Group. P30L controls X, Y, Z, A and B axes of the machine, the C-axis, the laser generator, the turn table and additional axes.

The multitasking controls allow work preparation and set up while the machine is running. Thus, the time spent for changing from one work to another is minimized.

The software provides an information system with alarm code explanation, user manuals, dynamic work queue function as well as fast file transfer. All these support the operator in several ways facilitating self-learning possibilities, giving recovery instructions and simple access to electronic manuals, spare part manuals etc.

A data base (TOB) with a large number of cutting parameters for various materials is stored on the P30L computer. To quickly optimize parameters for new or different materials the laser cutting parameters can be edited in the database. In addition, the system offers to the operator the possibility to customize cutting

parameters (e.g. feed rate, cutting gas pressure and so on) online, directly on the touch screen panel, even during the laser process.

The CNC is capable of high speed approach to the workpiece using the Fast Approach function.

Main features:

- Microsoft Windows 7 O.S;
- ISO G-code language;
- Fiber optic full-connected digital servodrives;
- Editing, preview and tracing functions;
- Technological parameters on CNC (integrated data base);
- Check scrap function;
- Integrated diagnostic system to identify and correct possible malfunctions;
- Machine remote monitoring (compliant with Industry 4.0 requirements);
- Teleservice connection to Prima Power Customer Service;
- Tools for automatic check and calibration;
- Integrated profibus board for connection with external devices;
- Maintenance manager;



Figure 3.8: LN1530 - CNC

3.4 Cabin and Turntable module

In order to conform with safety regulations (EN 60825-1), the machine is equipped with a safety cabin designed to protect the operator and any personnel located in the area around the machine from laser radiation (direct or diffused) and from moving mechanical parts.

The cabin encompasses the entire machine working area. It is equipped with a roof, which has air vents designed to compensate the internal vacuum produced by the air suction system, that ensures that the working area is completely closed.

The lighting inside the cabin is provided by six fluorescent lamps.

Two manual doors located on both sides of the cabin front part allows the operator to access the working area. Each manual door is equipped with a safety micro-switch. In case of accidental opening during the laser process, the shutter closes, the laser is turned off and the machine stops. The manual doors can be opened from the inside by pushing.

Two windows arranged on the right side and the left side of the cabin allow visual inspection of the machine work area. The windows prevent the escape of diffused laser radiation and are equipped with an active safety system which turns off the laser if the window is damaged by direct laser radiation.

The turntable in front of the machine is installed for the parts loading/unloading. It is equipped with a safety wall, which closes the front of the cabin and avoids exposure to laser radiation during the cutting process.

A LED monitor, located on the front of the cabin above the turntable, is connected to an indoor camera, which allows to control the work area.

The loading/unloading area in front of the machine is limited by protection devices (safety light curtains, scanner or fences) to control the access.

In the case of configuration with scraps conveyor there are fixed guards that protect the scraps unloading zone. The area is accessible through a manual door equipped with safety micro-switches which stop the movement of the conveyor.

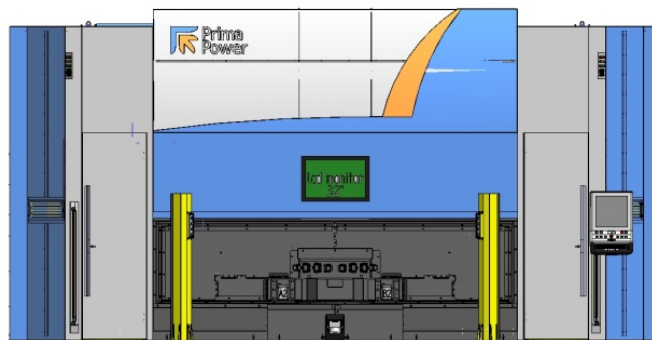


Figure 3.9: LN1530 - Cabin

The turntable module is the interface for loading and unloading parts and is constituted by:

1. Turntable unit
2. Tunnel group
3. Barrier group
4. Safety light curtains

The turntable is of the type with two positions and it carries out the alternate transfer of the equipment from the loading-unloading area to the working area of the machine and vice-versa. It is a CNC servo-controlled axis with a fast rotating time (2,3s) and a very short stop time ($<300\text{ms}$) in case of emergency stop. In its standard version the turntable is equipped with a CANBUS connection to the automatic fixtures.

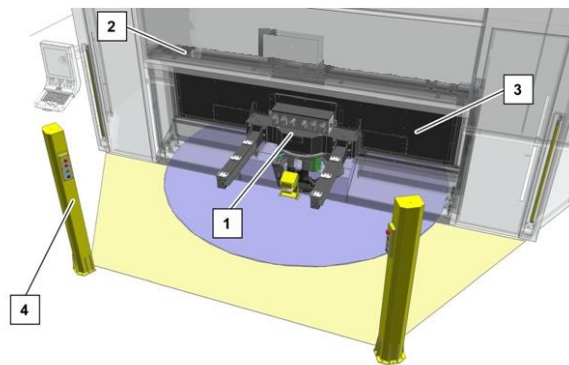


Figure 3.10: LN1530 - Turntable module

3.5 Problem analyzed

Some company departments - global service, installation and production - have been interviewed in order to determine the most frequent problems and the most difficult situations to manage. As a result, a selection has been made to identify the most significant:

- X-axis roller bearing damaging
- Setup servo drive incorrect
- Closing X carriage incorrect

- Closing Z carriage incorrect
- Faulty foundation / basement
- Defective brushless motor (linear or rotary)
- Optical line misalignment
- Bearing of ball screw damaged
- Loose belt
- Linear motors not well fixed
- Mechanical backlash position transducer

After an analysis, it has been decided to concentrate on the identification of the bad setup servo drive and on the loosening of the screws of the X carriage. The reasons behind the choice are the following:

1. They allowed immediately to make a first distinction between mechanical and software problem.
2. They were the only two problems that were easy to simulate and to reproduce over time because they did not involve an invasive intervention and, therefore, did not compromise the working status of the machine.

From the practical point of view, the first problem has been simulated going to intervene on the numerical control and modifying at a time the following parameters:

- velocity loop gain: set by default at 1450 Am/m and it has been reduced at 700 Am/m (figure 3.11).
- position loop gain: set by default at 120 1/s and it has been modified at 60 1/s and 180 1/s (figure 3.12).

These parameters, as many others, have been sent to the servo drive, placed in the electromechanical cabinet, which was in charge of giving the corresponding informations to the machine.

The second one consisted of loosening the four screws of the X carriage with the use of a dynamometric forceps (3.13): starting from condition in which they have been closed with a torque value of 200 Nm, different combinations have been carried out in order to cover the generality of the cases.

Two considerations have to keep in mind from now on:

1. The problems have been considered independently of each other: at this stage, the two situations have not been considered simultaneously.
2. The tests simulating these problems have been many and not all compromised the good quality of the cut of the machine: in some circumstances, they have been used as trend and a warning that continuing in that way a more serious problem would be happened.

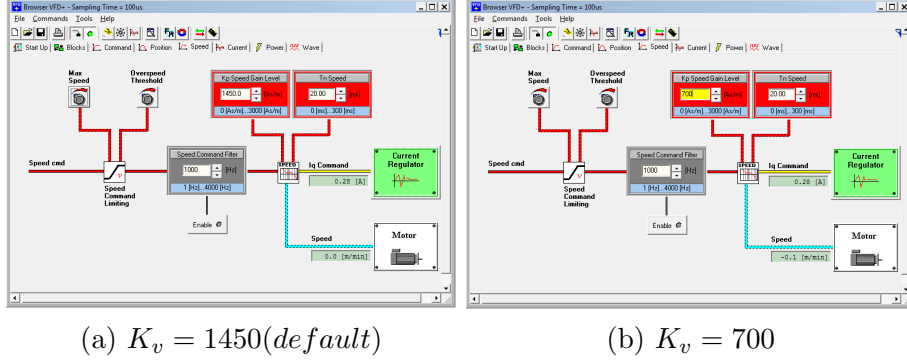


Figure 3.11: Servo drive parameters - velocity loop gain

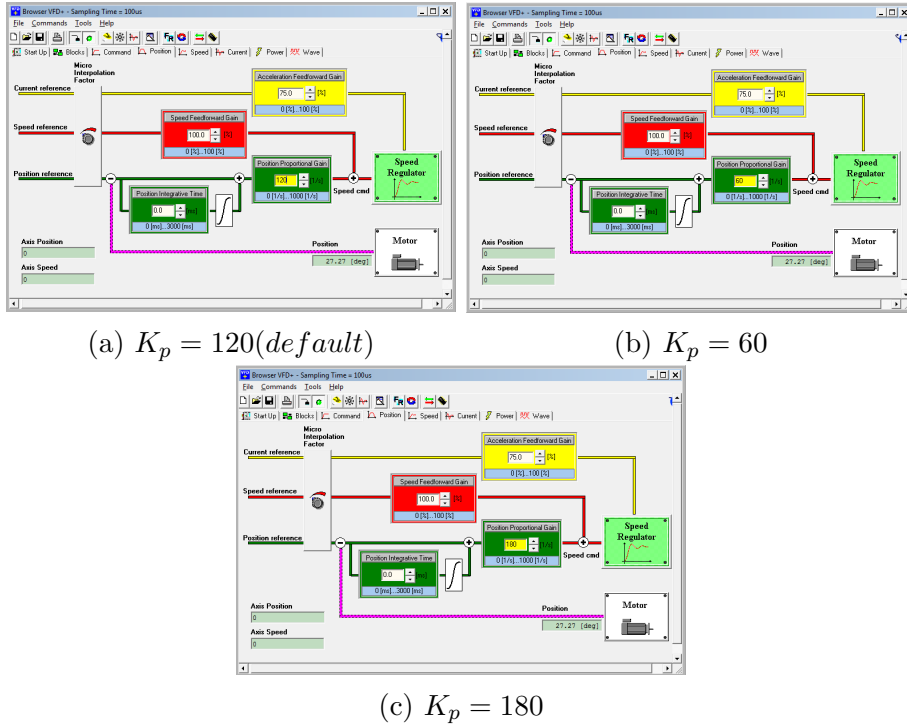


Figure 3.12: Servo drive parameters - position loop gain



(a) I condition



(b) II condition

Figure 3.13: Closing carriage anomaly - screws loose

Chapter 4

Instruments

Several tools and programs have been used during the execution of the project, in order to make it as compliant, uniform and adaptable as possible. In particular, the program developed for pulse generation, the sensors (accelerometer, encoder), the application for the accelerometer/PC connection and MATLAB will be discussed.

4.1 Impulse generator

In order to conduct an accurate modal analysis, the way in which the impulse has been generated has played an important role. The impulse has been simulated by a signal implemented via software: a series of strings of commands executed by numerical control of the machine (P30L). Unlike a physical signal - typically in these cases the easiest thing is to hit the machine with a tool (a hammer, for example) - the use of this method allowed to obtain the following advantages:

- The signal is always the same for all tests.
- The signal starts at the beginning of the control system covering the whole system chain and therefore all the components are stressed.
- The covered harmonic content is almost constant in the frequency range involved.

Before launching the execution of the program, some preliminary operations have been performed that allowed to amplify and to highlight the result. For this purpose, the parameters shown in the table 4.1 have been modified:

After that, it has been possible to proceed with the execution of the program: it is a short string of command repeated for as many times as desired that recreating a situation of abrupt braking. In the figure (4.1) is described one single pulse.


CNC command	Impulse	Original	Description
C99001	0.0012	0.1667	Minimum acceleration time [s]
C99068	6.0e7	2.4e6	Tangential dejerk [mm/s ³]
C00006	3.6e7	1.4e6	Cartesian dejerk [mm/s ³]
C01006	3.6e7	1.4e6	Cartesian dejerk [mm/s ³]
C02006	3.6e7	1.4e6	Cartesian dejerk [mm/s ³]

Table 4.1: CNC comand

In the first place, the machine has been brought into a starting situation where the X axis was retracted, the Y and Z extended: this configuration was chosen as a design specification at the beginning.

Subsequently, the machine started a movement at constant low speed - almost 10 times lower than cutting speed - along the positive X axis for 1s before decelerating, similar to emergency braking, along the same axis: this allowed to obtain a signal similar to a pulse. Obviously, actually, an ideal pulse can not be achieved, but the characteristics of the resulting signal have made it possible to speak of pulse.

As the figure (4.2) illustrates, the deceleration reached $9m/s^2$ for 0.008s.

 **IMPULSO-singolo-X - Blocco note**

File Modifica Formato Visualizza ?

G90
G08
G71
G73
G01

(* PROGRAMMA PER ANALISI ACCELEROMETRICHE AUTOMATICHE

G90
G100 A=0 B=0 C=0 X=-9 Y=1400 Z=-500 LF=3 LDO=1 LDE=1
G04 F1000G91 G01 X=100 LF=-1 LDO=100 LDE=1G04 F1000
G90
G100 A=0 B=0 C=0 X=-9 Y=1400 Z=-500 LF=3 LDO=1 LDE=1M30

Figure 4.1: CNC - pulse generation

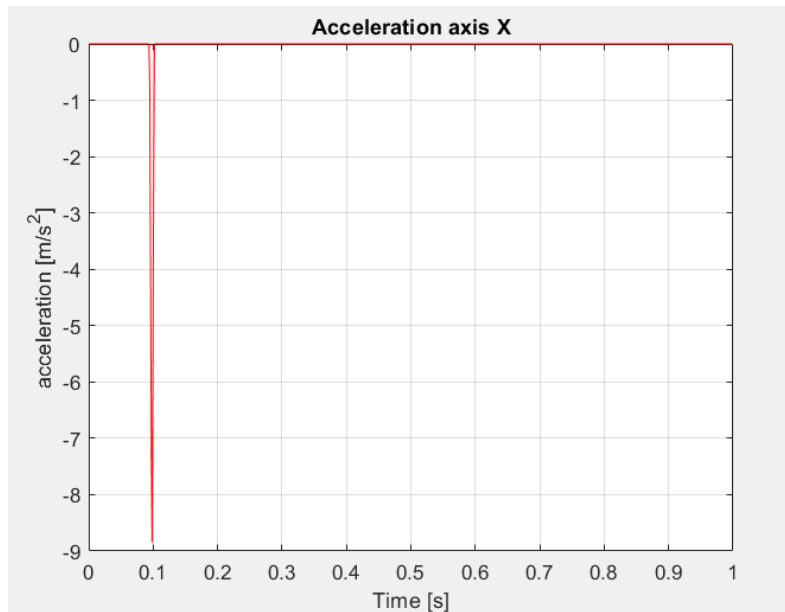


Figure 4.2: Pulse characteristics

4.2 Accelerometer

The accelerometer used is the Sequoia. It presented the following features:



Figure 4.3: Accelerometer Sequoia

- Triaxial: it gave the availability to get the acceleration on the three components (X, Y, Z), especially useful in anticipation of future scenarios.
- Sampling frequency: 8192 Hz
- Full scale: 5g
- Offset: ± 2 mg
- Sensitivity: ± 0.3

It has been decided to place it at the end of the kinematic chain (at the bottom of the Z axis) using beeswax, a technique commonly used in these circumstances. The position has been chosen following various reflections carried out at the beginning of the project, which had lead to the conclusion that this situation reflexs the best scenario in which the effects would be more visible. Its orientation with respect to the machine axes is as follows:

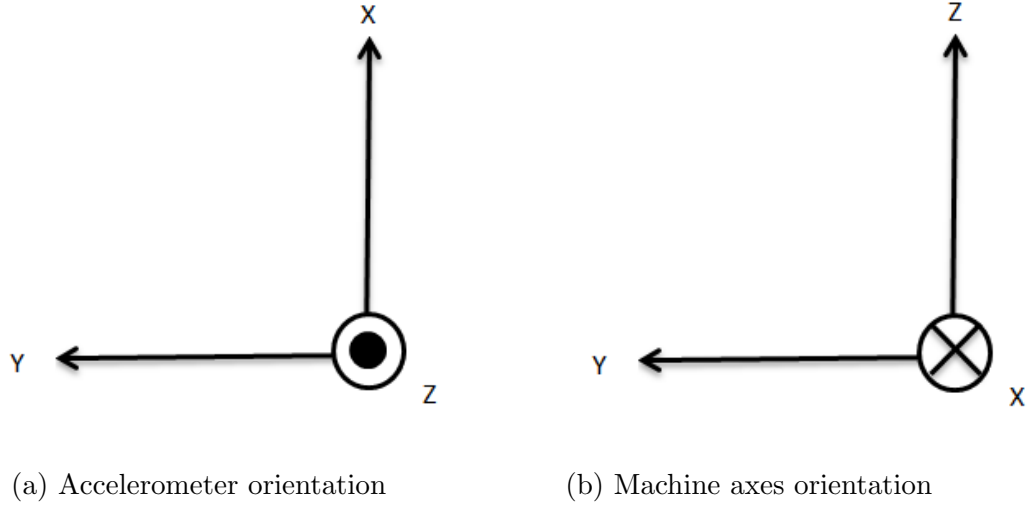


Figure 4.4: Accelerometer and Machine axes orientation

In order to standardize the method, during the processing of data, translation operations have been performed to associate the accelerometer axes with those of the machine, as explained better later.

4.3 Encoder - Oscilloscope

In addition to the use of the accelerometer, it has been decided to exploit the virtual oscilloscope present by default in the numerical control. On it, by means of an encoder, the servo position errors of the various fixed axes (X, Y, Z), mobile (a, b, c) and the acceleration have been represented. The operating principle was the same for all cases: through optical fibers, the numerical control sent the position control to the servo-drive, which in response sent the corresponding power signal to the motor. At this point the encoder, positioned along the optical line integrated in the guide along which the linear motor ran, recorded the real position of the motor obtaining a position error given by:

$$position_{error} = position_{cmd} - position_{real} \quad (4.1)$$

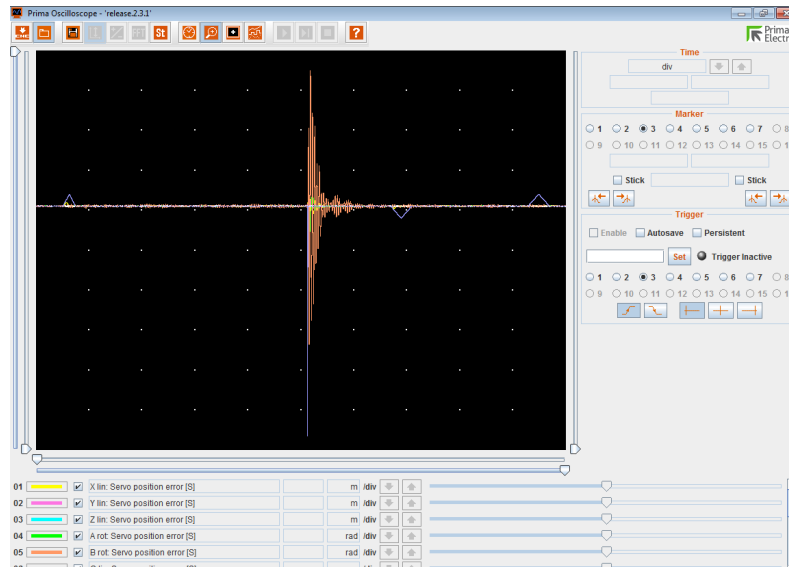



Figure 4.5: Error position on oscilloscope

This difference was shown on the oscilloscope which sampled with a sampling time of 1.2 ms - one order of magnitude lower than the accelerometer. During the data collection, to accelerate the operations, all tests performed in that configuration have been saved on a single file and, only subsequently, they have been separated using the acceleration as reference parameter.

4.4 PC-Accelerometer connection

To transmit the data from the accelerometer to the computer, an application has been created to let the accelerometer communicate directly with the PC. This procedure has been implemented in Java via the NetBeans platform. Using this type of programming language involves a series of innumerable advantages, among which surely the portability of the code on every operating systems. First of all, a command text file (*cmd.txt*) has been created in which a series of parameters have been inserted that must be recalled by the code to obtain the information regarding the serial connection and the accelerometer operation:

 **cmd - Blocco note**

File Modifica Formato Visualizza ?

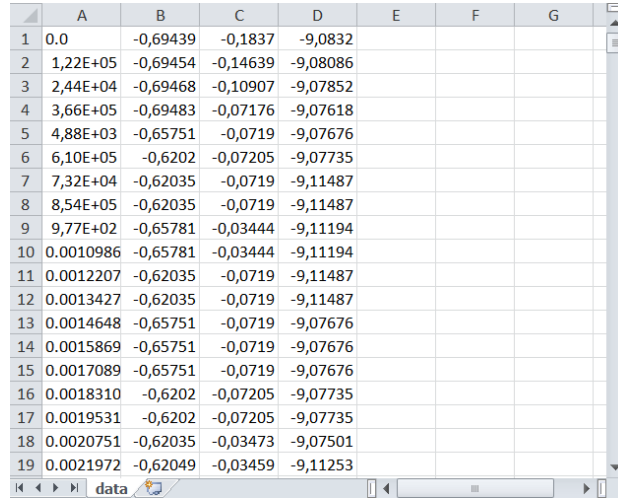
COM3;Z;1;30;;

Figure 4.6: Command file - example

The content of this file is just an example. Then, when the signals have been acquired from machine, the values have been changed as explained better later. Anyway, the meaning of the parameters is the following:

- '*COM3*' is the name of the serial port. According to the PC, it changes.
- '*Z*' is the acceleration component of the accelerometer to be analyzed. Respecting the different orientations, it changes depending to the axis on which the analysis is conducted.
- '*1*' represents the threshold above which the program starts to record the signal. The unit of measurement is m/s^2
- '*30*' is the waiting time interval within which the event must occur. It is expressed in second.
- '*space empty*' needs to the program to let it know that no file is present and, therefore, must take care to create a .csv file on which to write the data. In testing phase, it will be contain the name of the file.

After that, it has been drafted the code that took the signal, acquired by the accelerometer, when the event occurred and saved the content in a established order (time,accX,accY,accZ) on a .csv file, called *data.csv*.



	A	B	C	D	E	F	G
1	0.0	-0,69439	-0,1837	-9,0832			
2	1,22E+05	-0,69454	-0,14639	-9,08086			
3	2,44E+04	-0,69468	-0,10907	-9,07852			
4	3,66E+05	-0,69483	-0,07176	-9,07618			
5	4,88E+03	-0,65751	-0,0719	-9,07676			
6	6,10E+05	-0,6202	-0,07205	-9,07735			
7	7,32E+04	-0,62035	-0,0719	-9,11487			
8	8,54E+05	-0,62035	-0,0719	-9,11487			
9	9,77E+02	-0,65781	-0,03444	-9,11194			
10	0.0010986	-0,65781	-0,03444	-9,11194			
11	0.0012207	-0,62035	-0,0719	-9,11487			
12	0.0013427	-0,62035	-0,0719	-9,11487			
13	0.0014648	-0,65751	-0,0719	-9,07676			
14	0.0015869	-0,65751	-0,0719	-9,07676			
15	0.0017089	-0,65751	-0,0719	-9,07676			
16	0.0018310	-0,6202	-0,07205	-9,07735			
17	0.0019531	-0,6202	-0,07205	-9,07735			
18	0.0020751	-0,62035	-0,03473	-9,07501			
19	0.0021972	-0,62049	-0,03459	-9,11253			

Figure 4.7: File data.csv

In this case, the event consisted in overcoming a certain threshold by the signal - acceleration on one of the axes [m/s^2] - and only later were executed a series of instructions that led to saving of the same in a range of time ranging from 0.01s before the event until the next second.

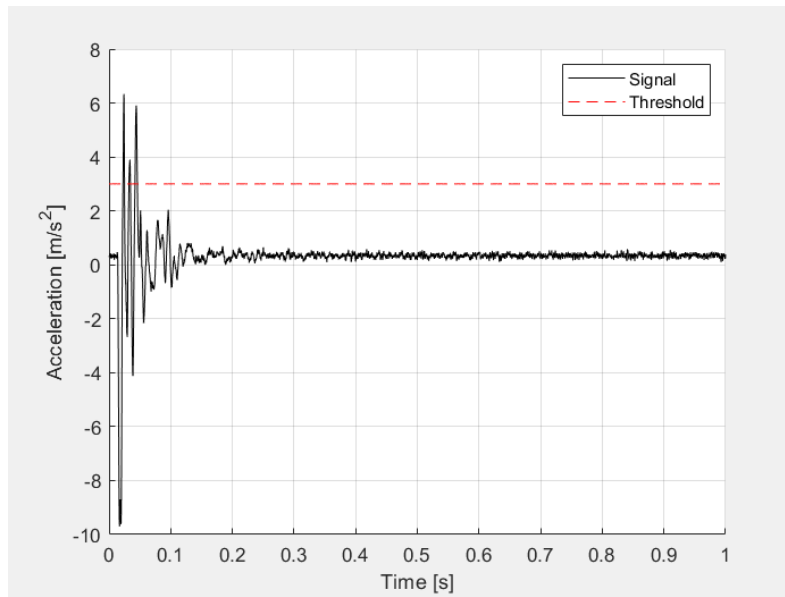


Figure 4.8: Event capture - example

```

Test Results  Output - JavaAccelerometer (run) ×
run:
File already present: null
Port name: COM3; TriggerLevel: 3.0; trSel: 3
Trigger Direction = 2 --> 2
Trigger Level = 3.0 m/s^2
Waiting for trigger...
Trigger Catch..!!!
Data Ready...writing to file....
Data has been writed to file! Exit process...
BUILD SUCCESSFUL (total time: 3 seconds)

```

Figure 4.9: JavaApplication - execution

For each acquisition, the user is always aware of the status of the acquisition by means of a series of strings, as illustrated in the figure (4.9):

For the realization of the code 7 classes have been created:

- JavaAccelerometer: the main class and it performs the main actions, ie reading the text file with the instructions and performing the thread, "Sequoia" and writing the signal output in an excel file.
- Sequoia: the thread is described, the accelerometer calibration functions are called and the "SerialPortCommunication" class is executed.

- **SerialPortCommunication**: describes the used serial port, the way in which it communicates with PC and all instructions that must be performed once the signal is acquired.
- **CalibrationAccelerometer**: contains all information necessary to describe the accelerometer (sensitivity, calibration vector and calibration matrix). These information have been taken from datasheet.
- **AccelerometerComponents**: in this class the bytes received from the accelerometer are processed to transform them into corresponding double values.
- **Command**: extract all information from the .txt file.
- **FileHelper**: invokes the "Command" class and inserts the acquired information into a list.

For the complete drafting of the code, please refer to Appendix A.

4.5 MATLAB

MATLAB is the most important tool for the realization of this thesis. Platform used to perform the analysis phase, data processing and for the construction of neural networks. It has also been used to implement the testing phase and to create a graphical interface (using the APP figure tool). The version used is 2018a. As partially mentioned before, given the power of this tool, some specific tools such as the Neural Network Toolbox and the Application Compiler (used for the development of the app) have been used.

Chapter 5

Method

The project consisted of performing the following operations:

1. Applying a pulse to the machine.
2. Seeing the response using the accelerometer and the encoder. In the first case, data have been saved directly on the PC, using the specific serial-communication protocol (written in JAVA) between the two devices. In the other case, data have been saved on the CNC and, then, imported into the PC using USB flash drive.
3. Analysis of the data: Fourier transform, interpolation of the data and other operations.
4. Pattern recognition: creation, training and validation of the neural networks.

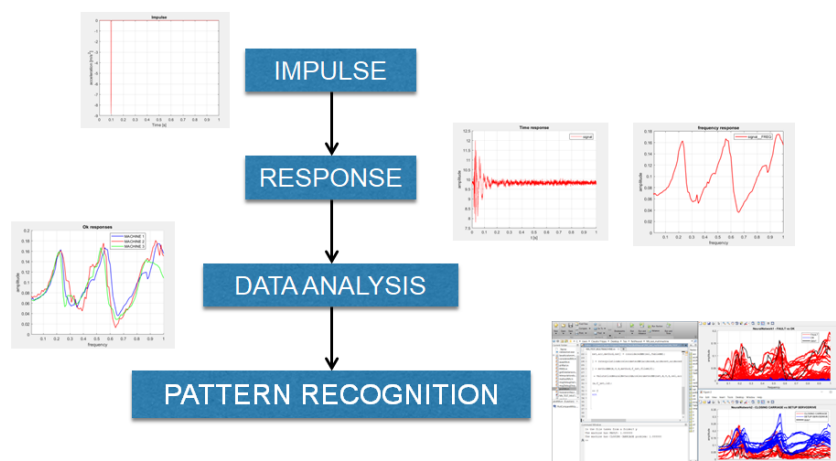


Figure 5.1: Project - main phases

The aim of the experiment is to prevent failures due to two potential anomalies, ie closing carriage problem and setup servo drive problem. For this purpose, several tests have been carried out simulating each of them.

In particular, 1093 signals have been recorded and saved using the accelerometer:

- 247 tests represented the normal condition
- 498 tests simulate the setup servo drive anomaly
- 348 tests simulate the closing carriage X anomaly.

Meanwhile, 510 data have been collected using the encoder:

- 105 tests represented the normal condition
- 225 tests simulate the setup servo drive anomaly
- 180 tests simulate the closing carriage X anomaly.

Since the sensors used were two, as many case studies have been carried out:

- case study 1: based on the use of an accelerometer. The involved machines were 9: 4 have been used for training the classifier and the remaining 5 for verifying and adding validity.
- case study 2: based on the use of encoder. The involved machine were 5: 3 for the training phase and the remaining for increasing the number of tests.

Both case study have been implemented in MATLAB.

5.1 Case study 1 : accelerometer

As mentioned before, each time the machine has been excited by the pulse, its response has been detected by the accelerometer. Using a MATLAB function, the cmd.txt file has been modified in order to adapt to our situation:

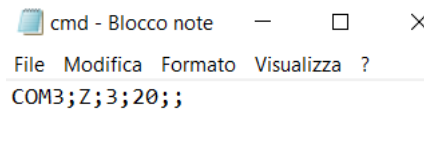


Figure 5.2: Command File - Accelerometer

Thus, in our case the reference acceleration component was the Z one and the threshold was fixed at $3m/s^2$. Then, the JAVA application has been recalled: when the event happened, the signal was saved in the PC in a .csv file. Considering

the characteristics of the accelerometer (sample time = 0.12ms) and the signal duration, all files contained 8192 data for each field - the time and the acceleration components(X,Y,Z).

Since the accelerometer and the machine axes had a different orientation, as emerged by the figure (4.4), in order to standardize the data, they have been translated:

- Acceleration along the machine axis X = - Z accelerometer component
- Acceleration along the machine axis Y = Y accelerometer component
- Acceleration along the machine axis Z = X accelerometer component

Once I do this operation, the time-responses of all situations have been plotted distinguishing the three cases:

- normal situation: represented in blue color.
- setup servo drive anomaly: represented in red color.
- closing carriage anomaly: represented in green color.

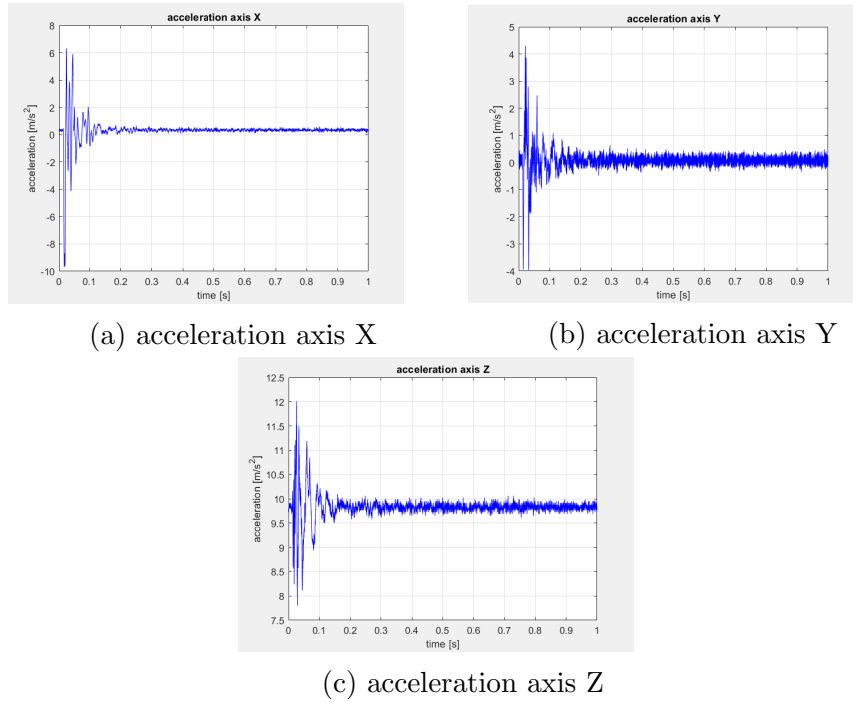


Figure 5.3: Time responses - normal situation

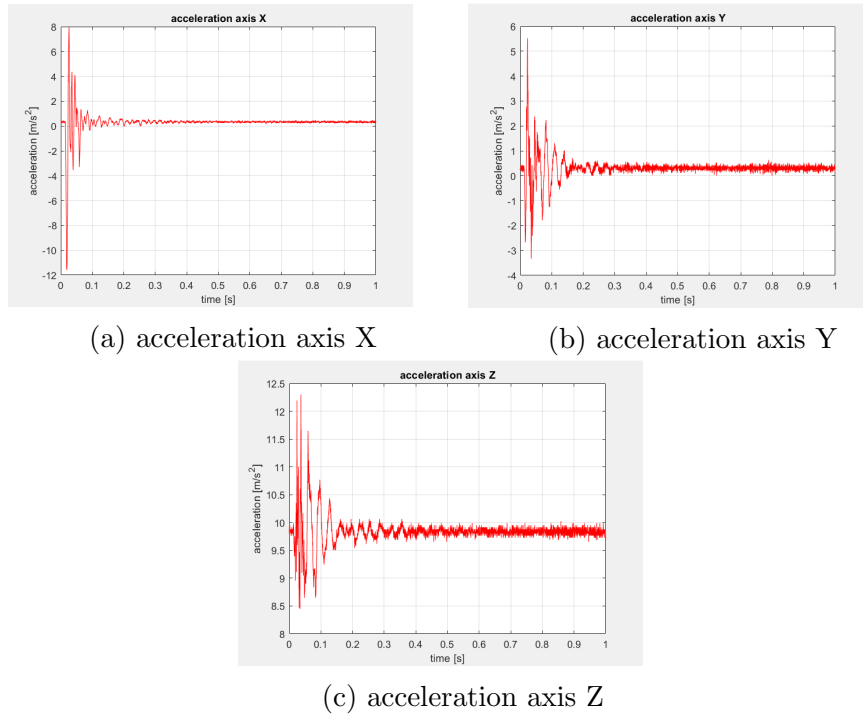


Figure 5.4: Time responses - servo drive anomaly

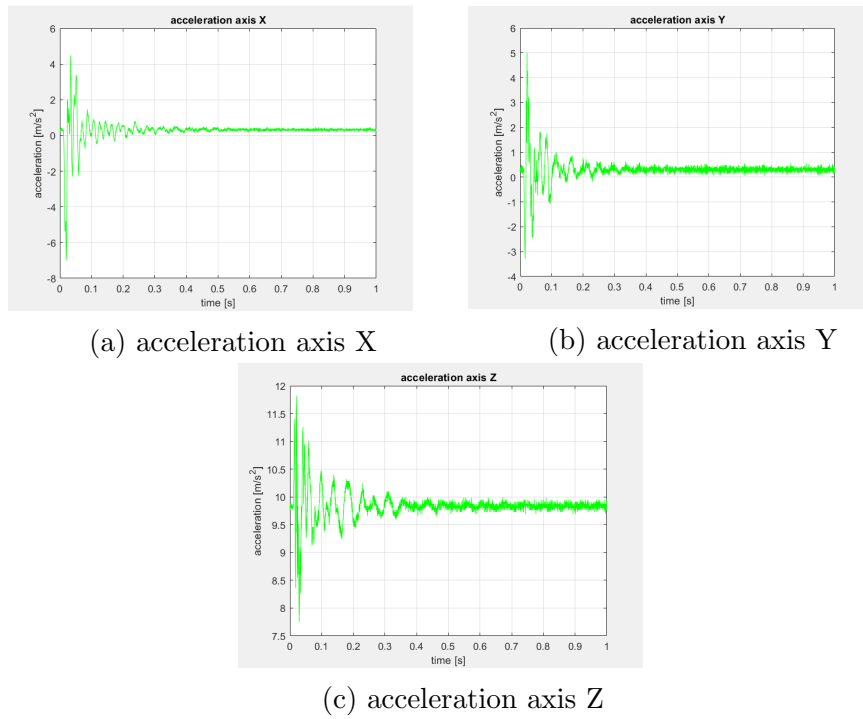


Figure 5.5: Time responses - closing carriage anomaly

From the above graphs, it is difficult to determine unique indications on the different situations. For this reason, the domain of interest has been changed from time to frequency, performing the Fourier transform.

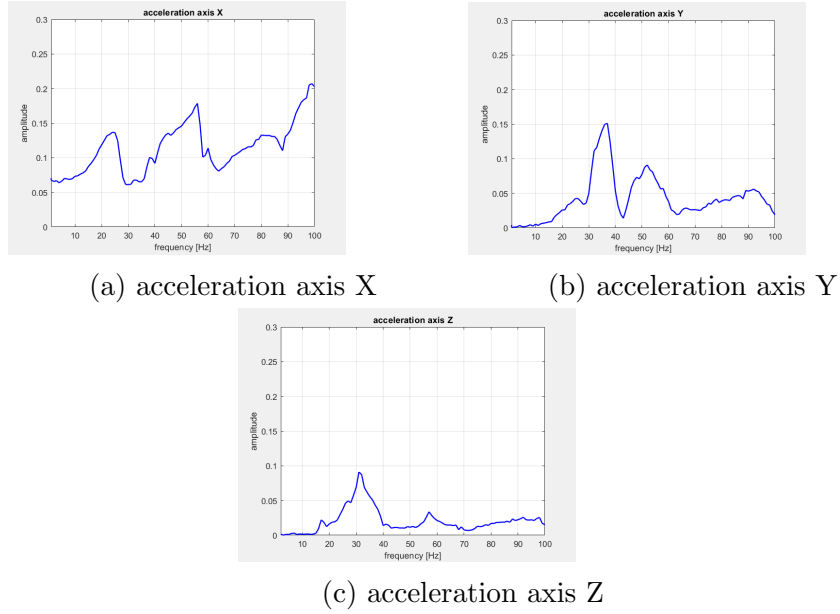


Figure 5.6: Frequency responses - normal situation

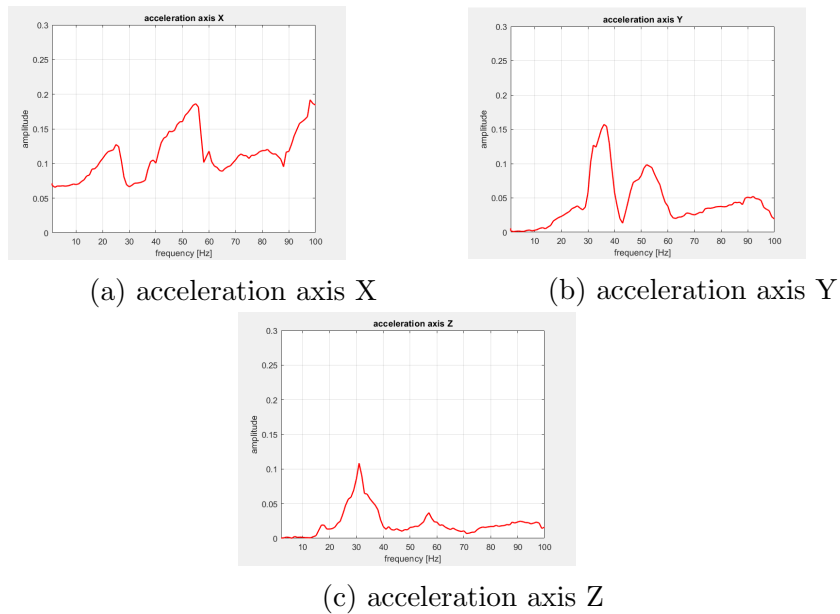


Figure 5.7: Frequency responses - servo drive anomaly

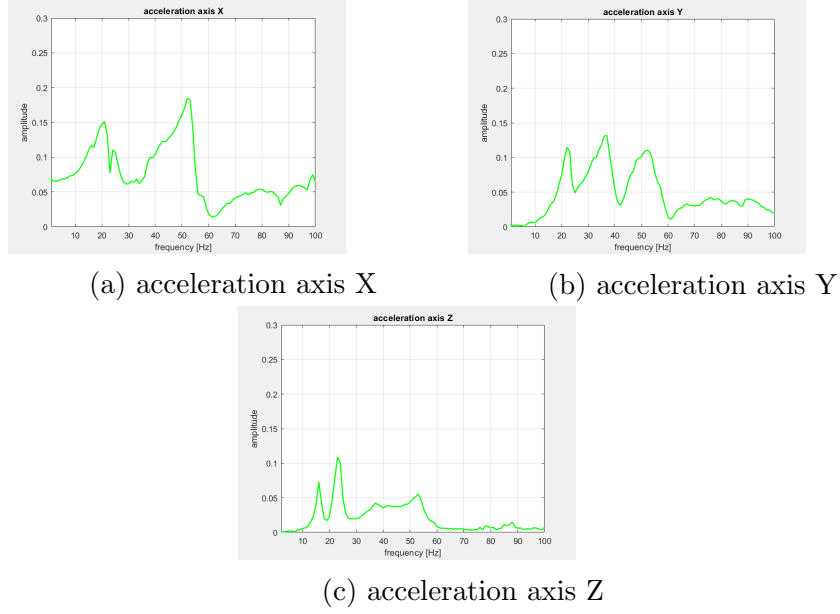


Figure 5.8: Frequency responses - closing carriage anomaly

Then, since the considered files were many and of big size, to reduce computational cost and to make the analysis faster, their dimensions have been reduced by means of linear interpolation.

Starting from 8192 values, only 100 have been considered: this number came out from the fact that the analysis has been focused in the range from 1 to 100 Hz and, therefore, the amplitude is result of the association with each unit of frequency. If higher frequencies had been considered, there would be high probability that noise and external factors would have too much importance.

Moreover, always in this context of optimization of the method, the further analysis have been carried out using only the signals relating to the X axis since, from graphical study, it was the case that highlights the differences.

5.1.1 Data Analysis

Following the analysis, these two considerations have been drawn:

1. the machines in normal condition had a behavior that was similar but not identical between each other.

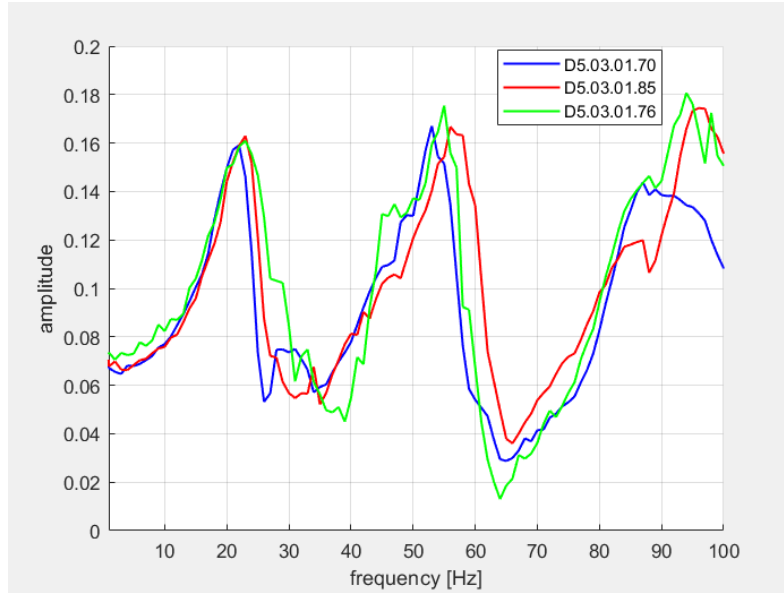
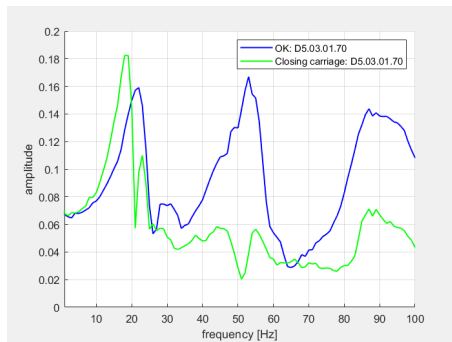
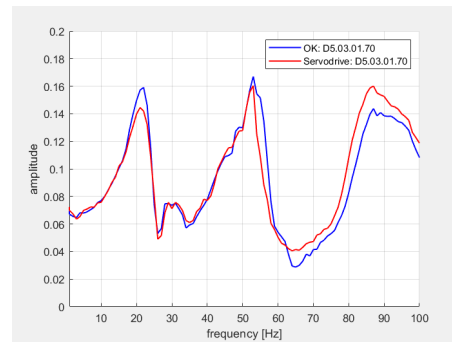


Figure 5.9: Correct responses of more machines overlapped

2. the machine in anomaly conditions had a frequency response whose can be very different (closing carriage problem) but also similar (setup servo drive anomaly), in most of the considered frequencies, when compared with the response in normal condition.



(a) ok vs closing carriage



(b) ok vs Setup servodrive

Figure 5.10: Anomalies responses - comparison

Thus, the problem was: how can finding the relations and providing the correct outputs be made easier for neural network in this condition??

Two answers have been designed:

- Multi-phases
- Single-phase

5.1.2 Multi-phases

The first solution is to split the entire problem in two phases:

1. Detect if an anomaly is occurring.
2. Detect which anomalies between setup servodrive and closing carriage problem is occurring.

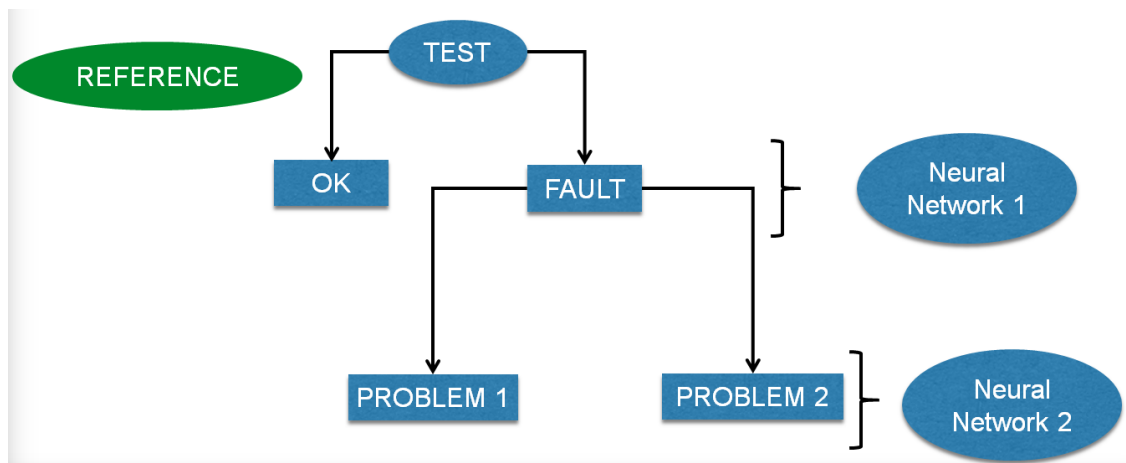


Figure 5.11: Multi phases - general scheme

For doing this, in the first phase, a new element has been introduced: the reference file.

For each machine a signal corresponding to normal condition is kept, as a sort of machine ‘fingerprint’. Practically speaking, when the machine is ”born”, and it surely works good, a signal is acquired and set aside.

During the machine lifetime, the subsequent signals will be always compared to this signal, through the absolute value of the subtraction. In this way, the neural network easily is able to give to the user a warning if something is going wrong.

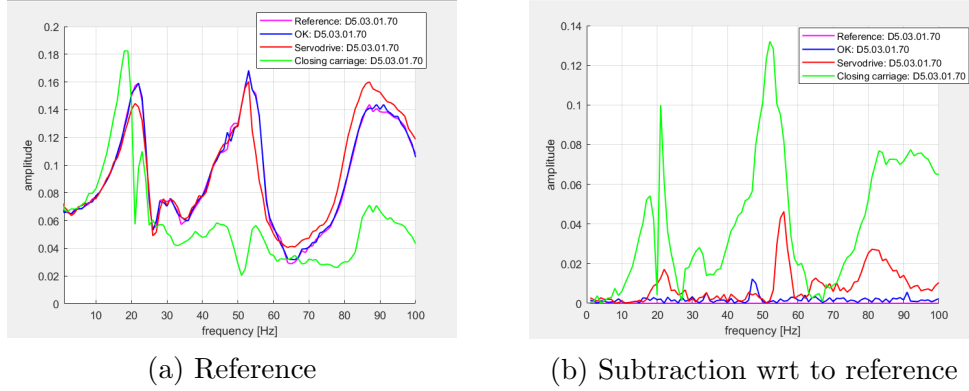


Figure 5.12: MP - Reference file

Neural Networks

To implement this kind of solution, two neural networks have been used that had the following characteristics:

- Number of network inputs = 100;
- Number of network outputs = 2;
- Number of hidden neurons = 20;

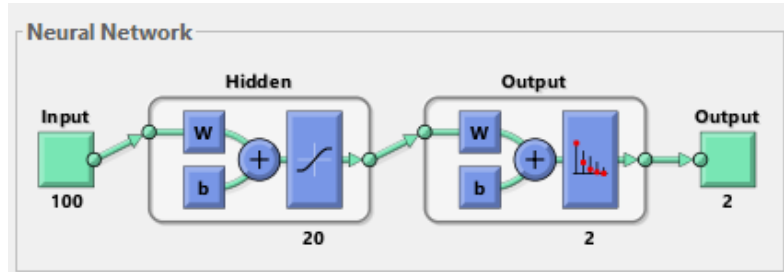


Figure 5.13: Neural Network - architecture

The first neural network has been trained to detect whether an anomaly was occurring and, for this purpose, a dataset has been produced with two types of inputs: anomaly situation (closing carriage + setup servodrive) and normal situation.

Furthermore, a graph has been created in order to have a clear idea on what in this phase the neural network had to learn, on which the blue signals, corresponding to normal condition, and red ones, to anomaly condition have been plotted (figure 5.14).

The neural network produced the following Confusion matrix and ROC (figure 5.15).

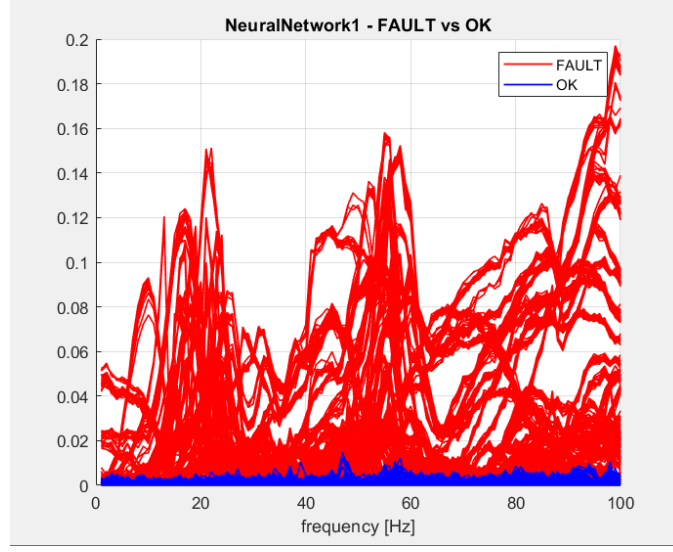


Figure 5.14: Neural network dataset - first phase

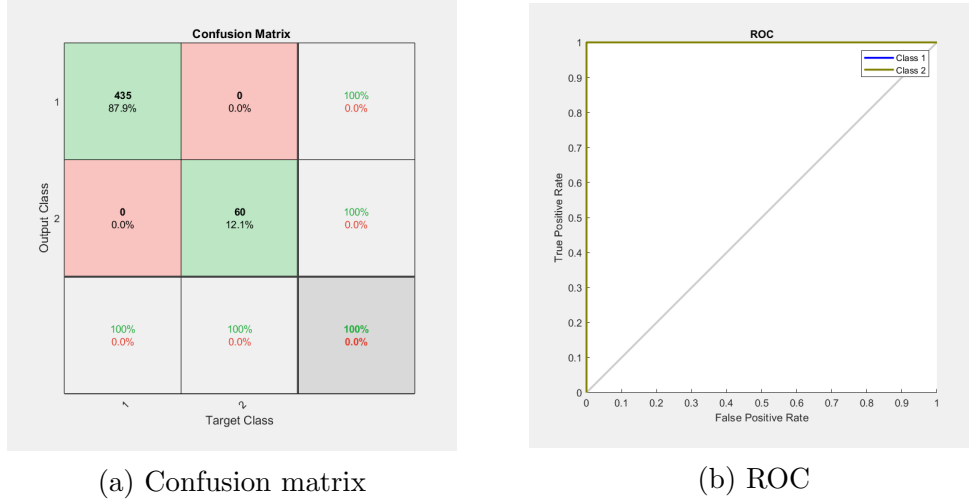


Figure 5.15: Confusion matrix and ROC - first phase

As evident from above, the training has been carried out efficiently and, so, it has not been necessary to intervene in order to improve the model.

Using a similar procedure, the second phase has been implemented. In this stage, no additional computations have been necessary and, so, the signals remained in their original form. In this phase, the second neural network had the task to distinguish which problem was occurring on the machine, between setup servo drive and closing carriage anomaly.

As before, a dataset has been created in which, on the one hand, there were files corresponding to closing carriage problem and, on the other, there were ones

related to setup servo drive anomaly. The corresponding plot is given in the figure (5.16).

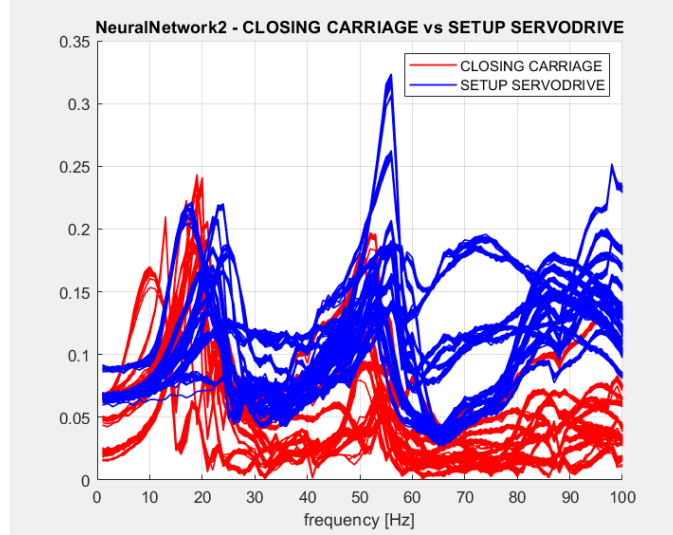


Figure 5.16: Neural network dataset - second phase

The Confusion matrix and the ROC generated from the training are illustrated in the figure (5.17).

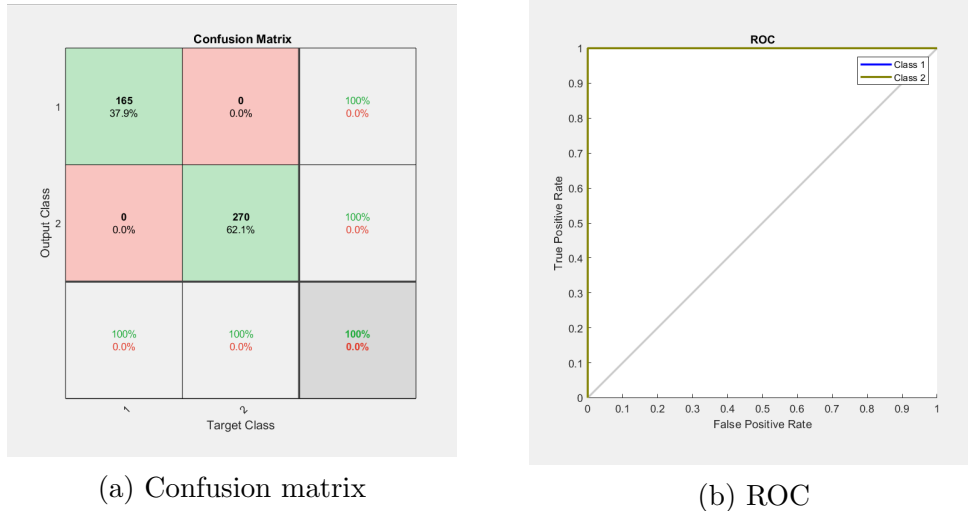


Figure 5.17: Confusion matrix and ROC - second phase

Also in this case, the training had a good accuracy.

Test

To follow, a MATLAB script simulating the testing procedure has been implemented.

First of all, the user has to select the reference file according to the machine he/she wants to analyze. Then the system will request if the file to test is already present or it has to be acquired in real-time by the accelerometer. When selected the desired option and acquired the sample, the system automatically will start the cycle that will lead to the result: the appropriate neural networks will be recalled according to the actual phase and the corresponding outgoings will be written in a string in the command window of MATLAB. If from the first phase no anomaly has been detected, the system is forced to not do further analysis.

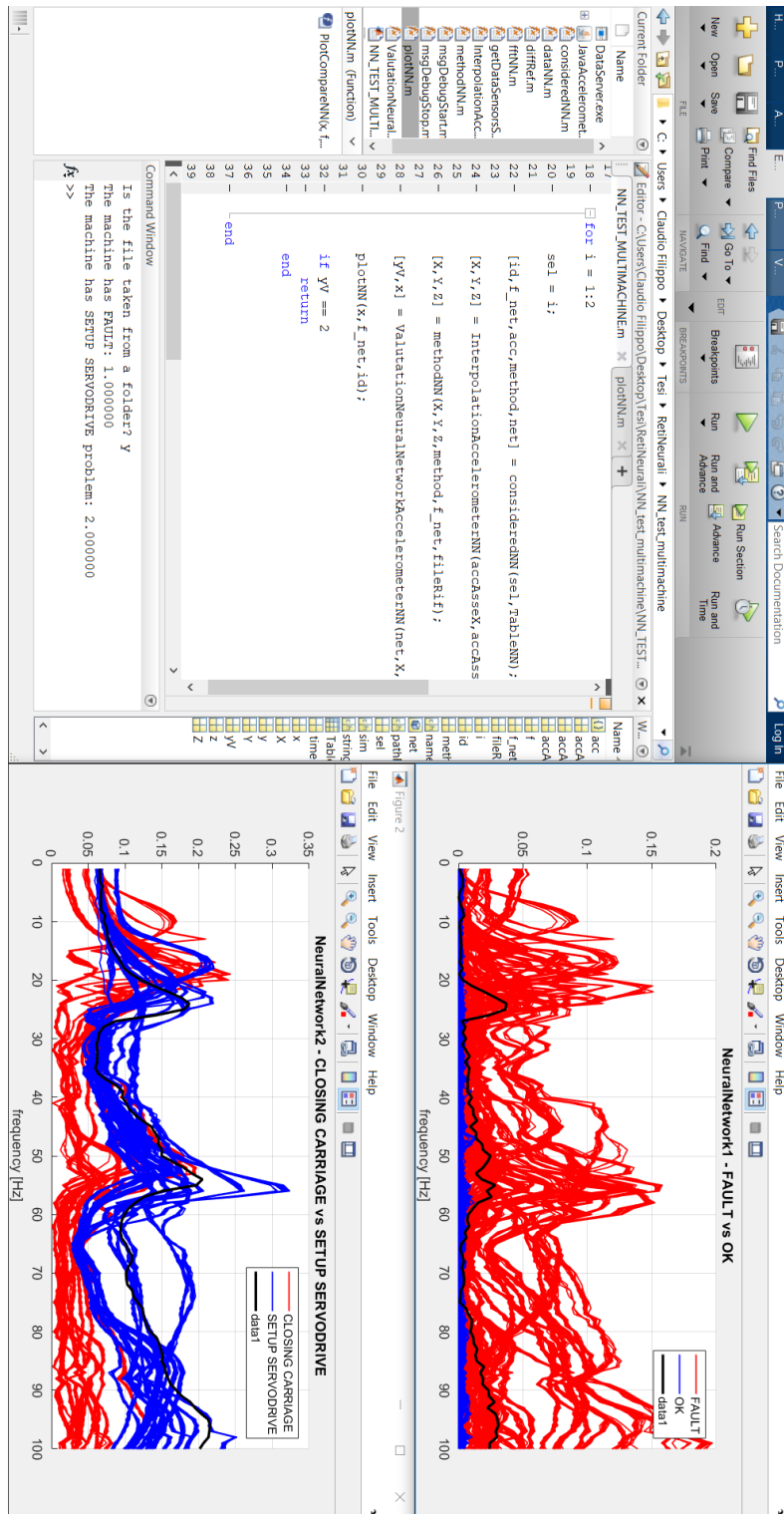


Figure 5.18: Test - multi phases

APP

As supplementary operation to testing script, the corresponding graph interface has been implemented using the APP figure environment and from which an APP has been developed using the APP development ToolBox. In this way, this tool has become independent from MATLAB environment and can run and work in each PC. Referring to the figure (5.19):

1. *Port Address* button serves to indicate which port is used by accelerometer to connect itself with PC.
2. *File Reference* button allows to select the reference file used for the test. Once selected a string with the corresponding machine will appear in the below field. In case of missing of the selection of the reference file, a warning dialog will appear informing the user to make a choice.
3. *Start* is used to take the test. In particular, a led becomes green when the system is ready to consider a new sample and will turn red when the signal will be acquired (real-time) or selected in a folder.
4. *Clean* button allows to cancel all information considered until now and, so, if pressed, it is necessary to reselect the file reference.
5. List of flag command: the *simulation* flag allows to decide whether the test must be taken from a folder or acquired in real-time; the *enable graph* decides whether to show the graphs below; the *average data training set* flag creates an average signal of the different categories of the figure (5.14) and (5.16) instead of representing them: it is used to make the graphs a little more clear; the *show Ok curve* and *show led* are used to show or hide, respectively, the Ok curve in the right graph and the decisional led.
6. It is the main part: here the result of the neural networks are shown lighting up the appropriate leds in red. The *Info NN1* and *Info NN2* buttons, if clicked open two files containing the corresponding informations about the neural networks. In addition to the usual cases until now treated, a new one has been added called *Warning*, which collects all unknown cases: it is a reminder that the neural networks work well only for cases for which they have been trained. It is introduced in the first phase to set aside all signal considered in the *ok* case but, as graphically evidenced, they are not. It is something that falls out the neural networks, since it has been implemented in mathematical way: if the area under the curve is over a certain value, the signal enters in this case.
7. The two graphs represent graphically how the neural networks think. Thus, they are used to add security to the neural network results since the user can easily understand if they make sense.

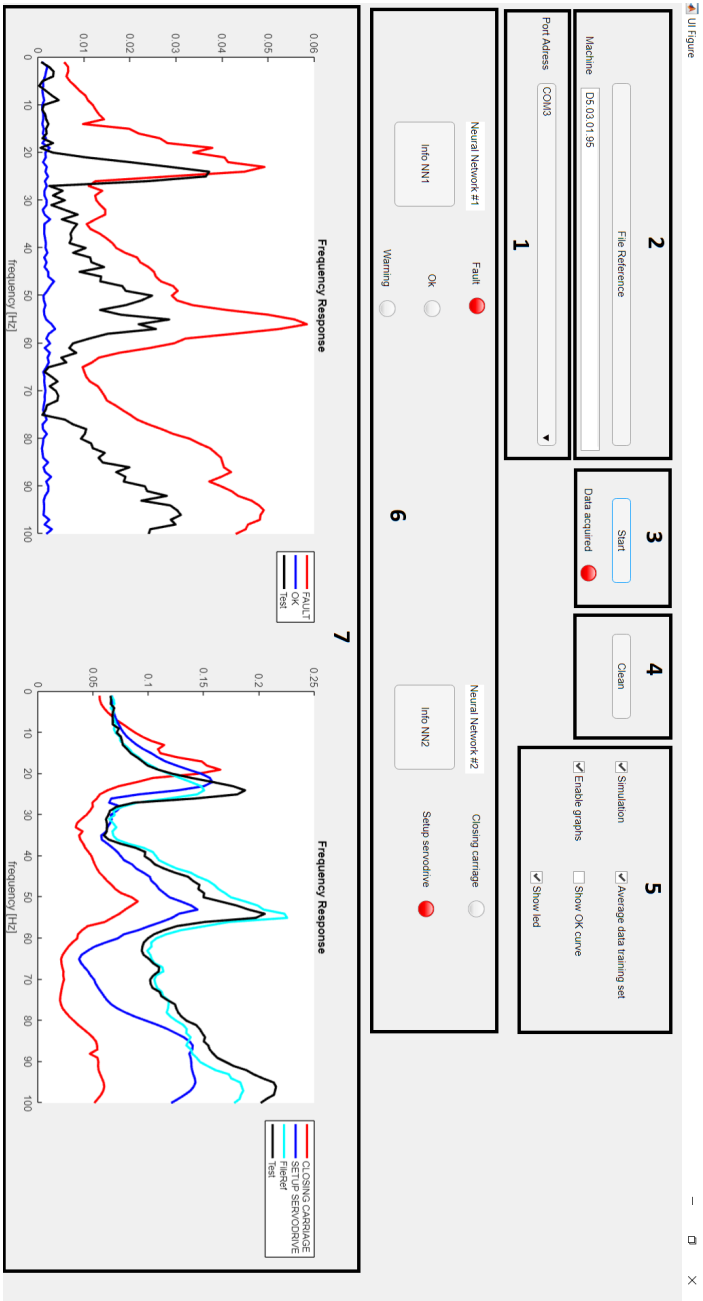


Figure 5.19: APP

5.1.3 Single-phase

Another solution is to consider independently each machine and, therefore, create a model that is valid only for that machine. In this way the potential machine failure can be detected in only one phase and without adding operations.

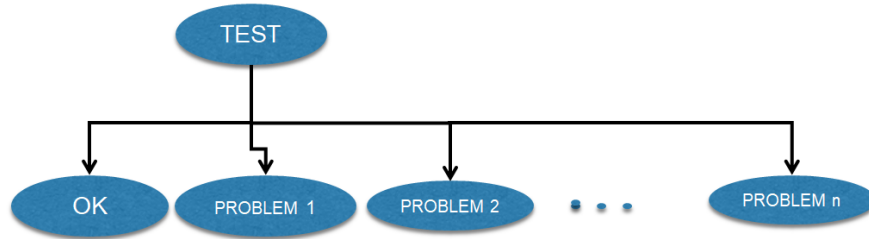


Figure 5.20: Single-phase general scheme

Neural Network

In this case, a dataset for each machine has been created: each of them included tests about all situations - normal behavior, setup servodrive anomaly and closing carriage problem - put together.

The corresponding graph was the following:

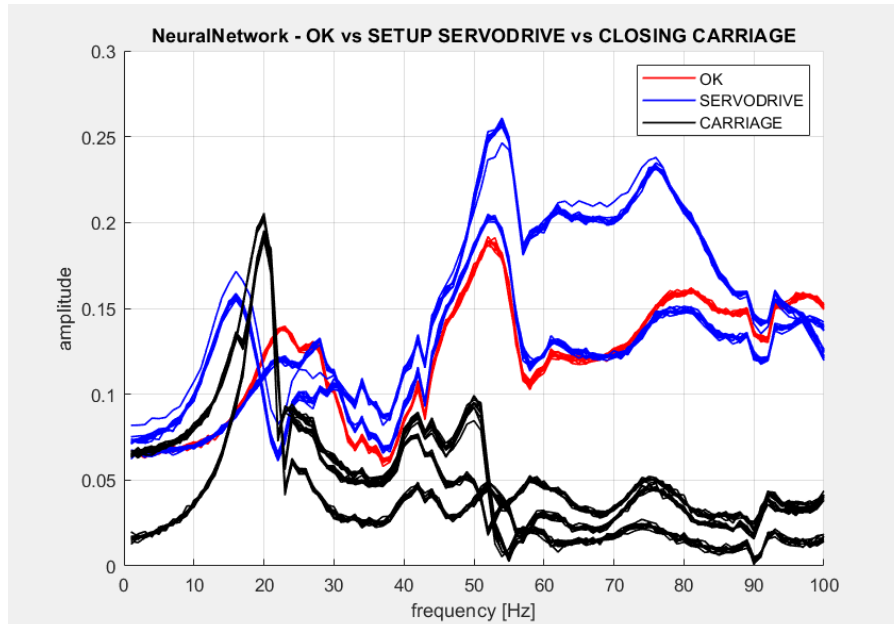


Figure 5.21: Neural Network dataset

For all machines, the neural network, which had the same properties of the previous one, reached good accuracy as emerged from the Confusion matrix and ROC:

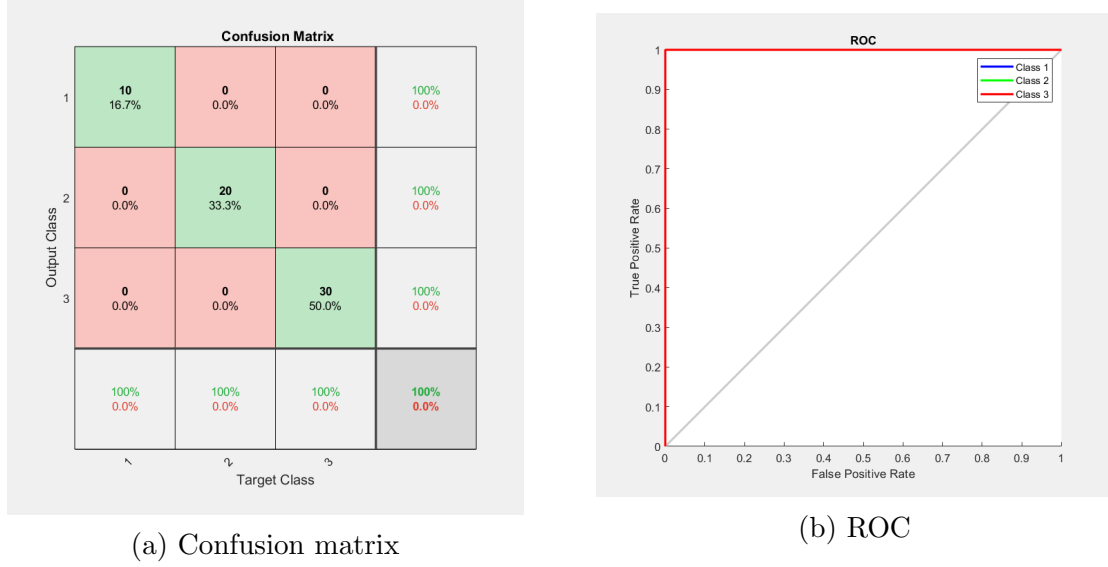


Figure 5.22: Confusion matrix and ROC

Test

As last procedure, a MATLAB script for the testing phase has been implemented in order to verify the validity of each neural network. The main difference from the previous one is the necessity to select the machine on which to do the analysis. Once you do this selection, the system will automatically recall the corresponding neural network and, then, it will require to the user where it has to take the test: either it has to acquire in real-time from the machine or take from a folder a file already present. Finally, a message containing the selected machine and the actual condition will appear.

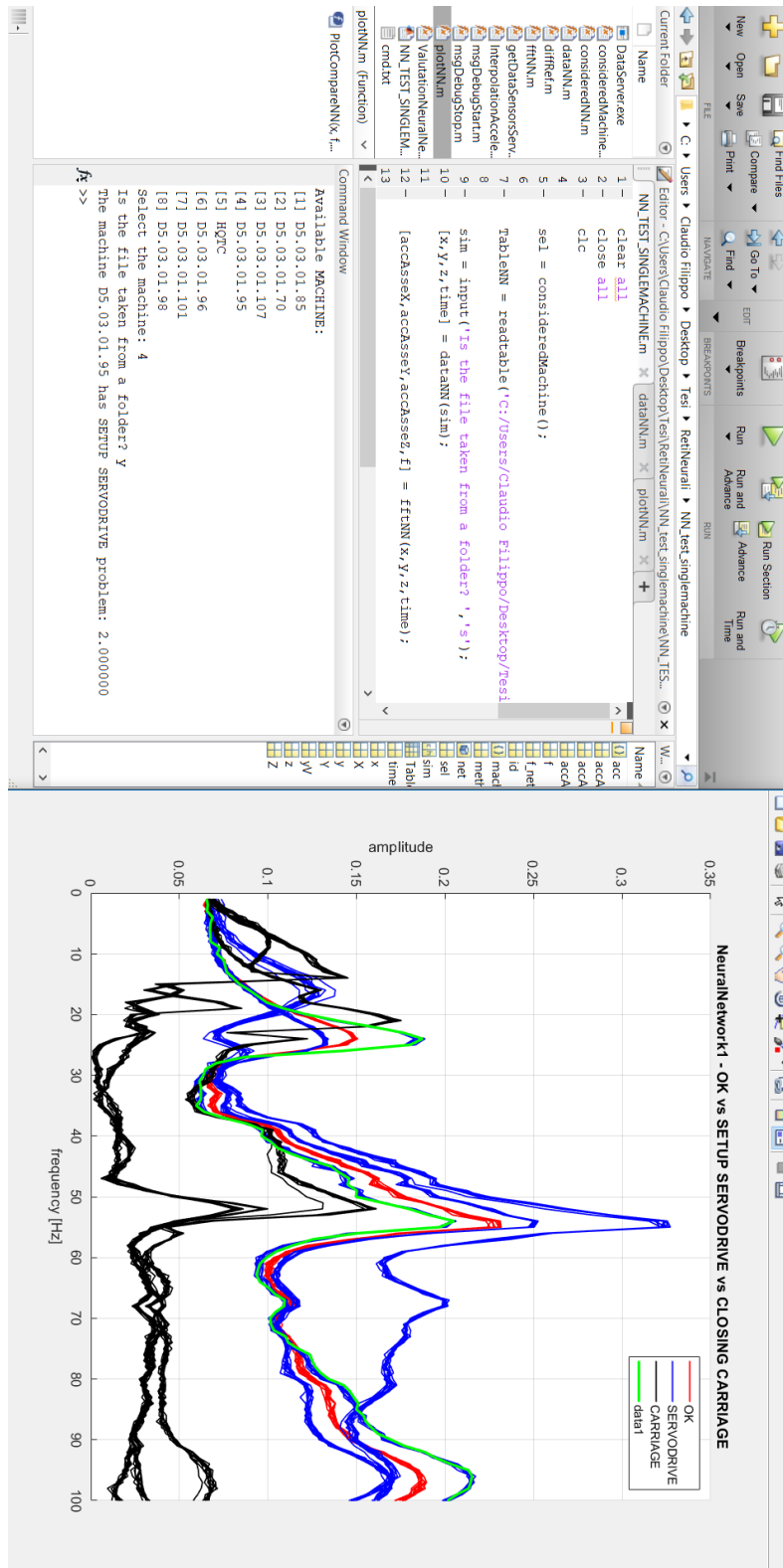


Figure 5.23: Test - single phase

5.1.4 Conclusion

This single-phase solution seems better than the multi-phases one because does the same job in a faster and easier way, since has only one phase and no further operations on data.

However, for each machine, it requires all necessary tests for training the corresponding neural network: it is not so much useful for the purpose one wants to reach since the degree of autonomy is strongly limited to the selected machine.

For this reason, at this stage of analysis, the first solution is preferable.

5.2 Case study 2 : encoder

In this second case, the focus has been switched on axes servo position error. Using the virtual oscilloscope, the servo position error of all axes - X,Y,Z,a,b,c - and the acceleration have been plotted. Since to make the process faster each time 15 signals have been recorded consecutively, the acceleration signal, whose characteristic is similar to a pulse as evidenced by figure (5.24), has been used as reference signal in order to split each of them and standardize all in such a way as to have all the other components of the signal singularly starting at the same time. In particular, since each component had to be of 1s sampled every 1.2ms, each time the acceleration exceeded the threshold of $-8m/s^2$, the system considered 83 data before and 750 after this event.

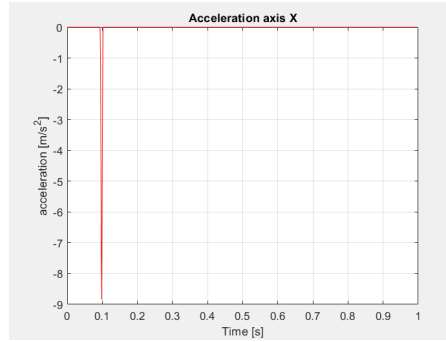


Figure 5.24: Acceleration signal

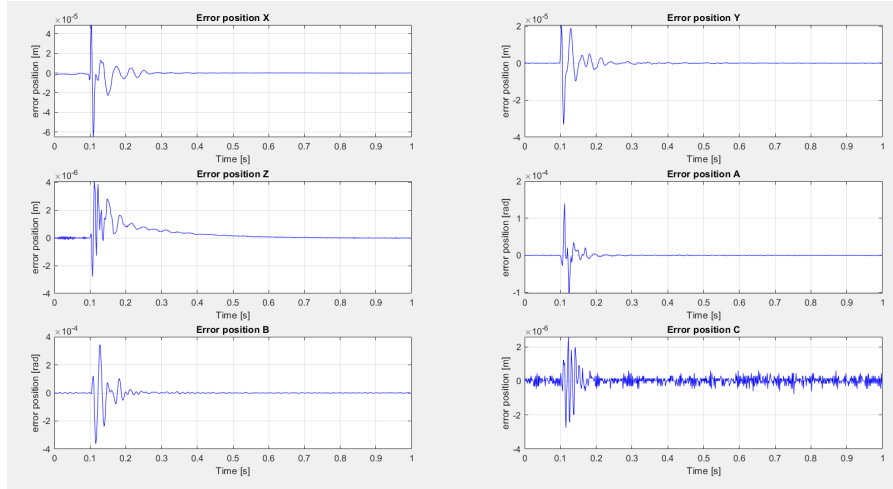


Figure 5.25: Time responses - normal condition

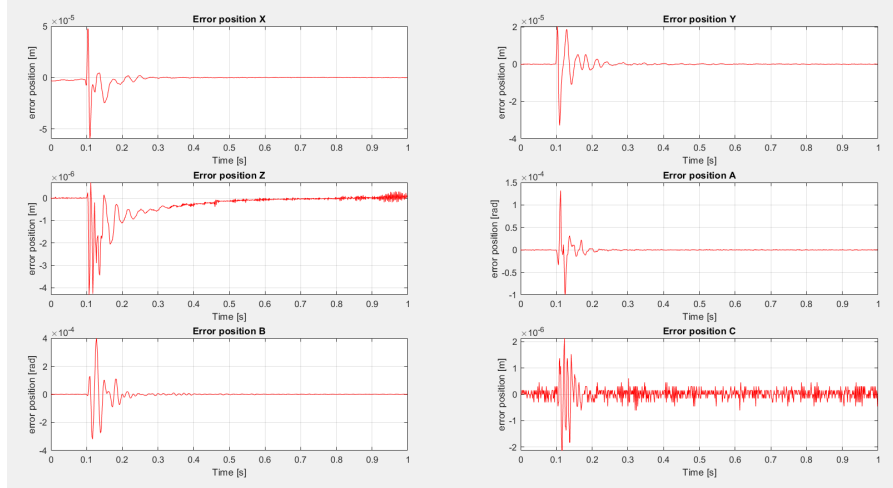


Figure 5.26: Time responses - servo drive anomaly

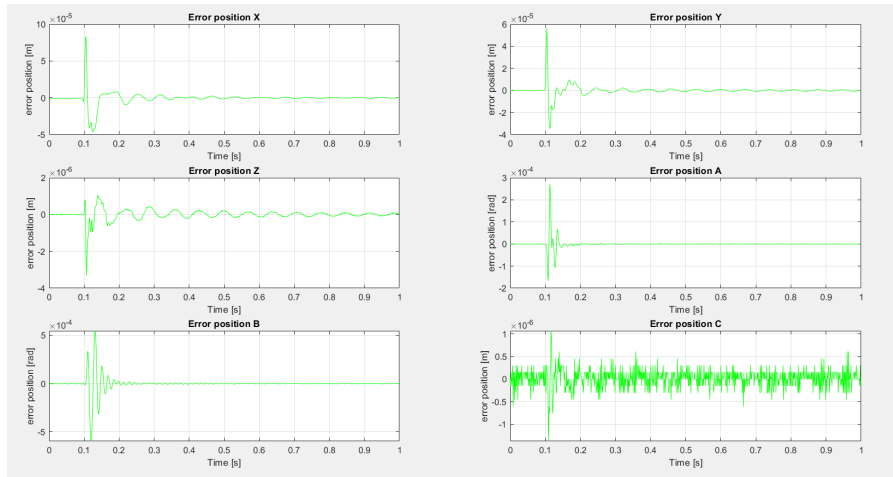


Figure 5.27: Time responses - carriage anomaly

As in the previous case study, the move from time to frequency domain has been necessary to notice the differences among the cases.

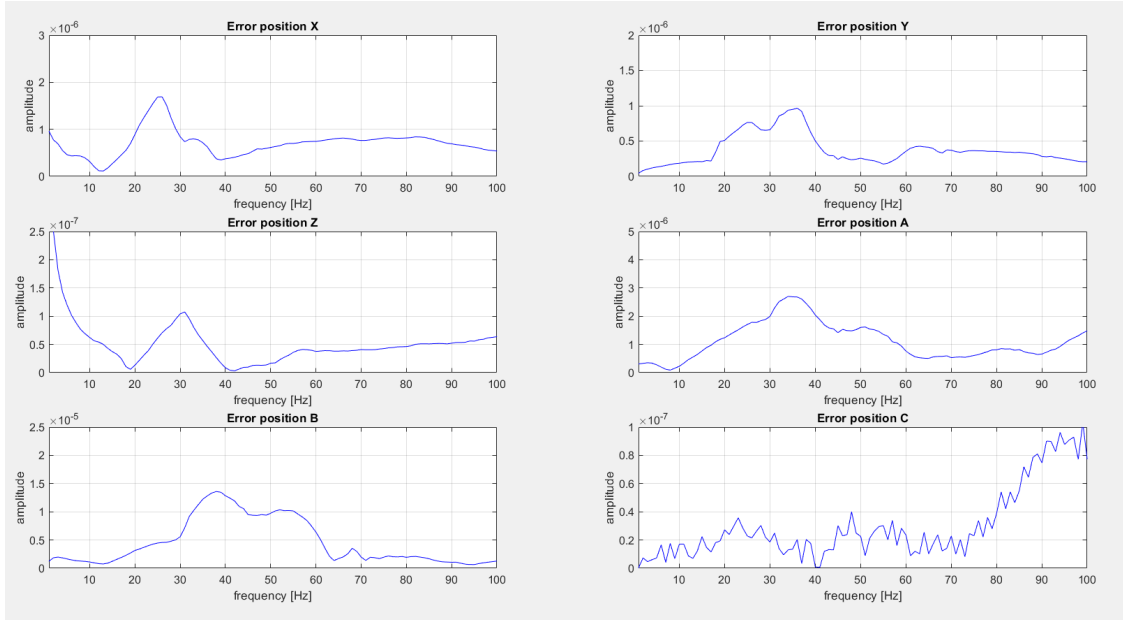


Figure 5.28: Frequency responses - normal condition

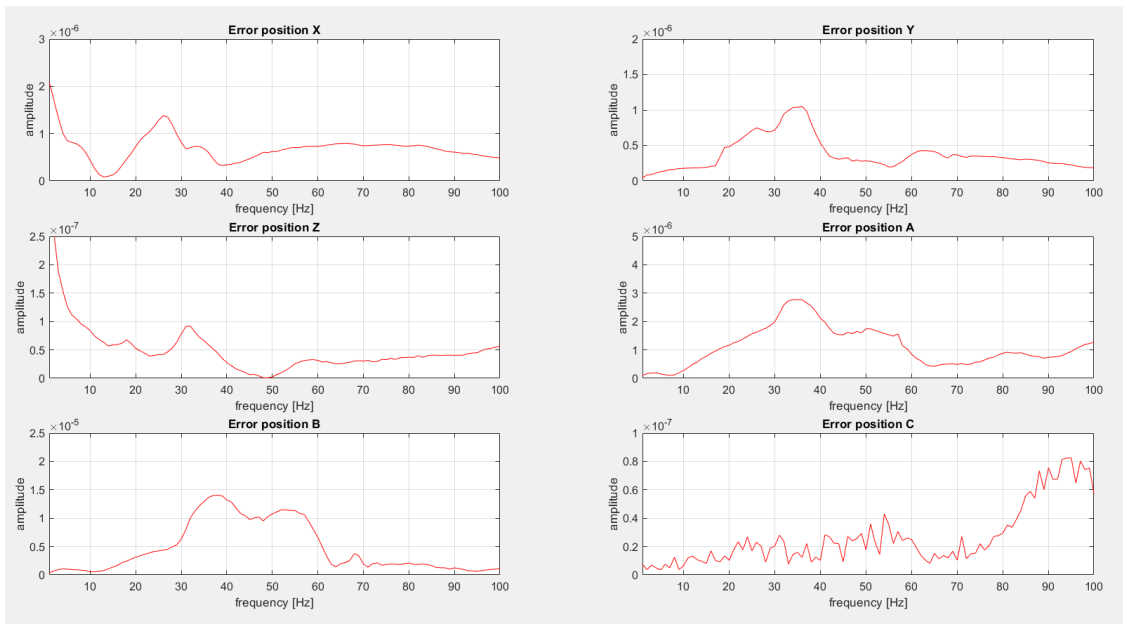


Figure 5.29: Frequency responses - servo drive anomaly

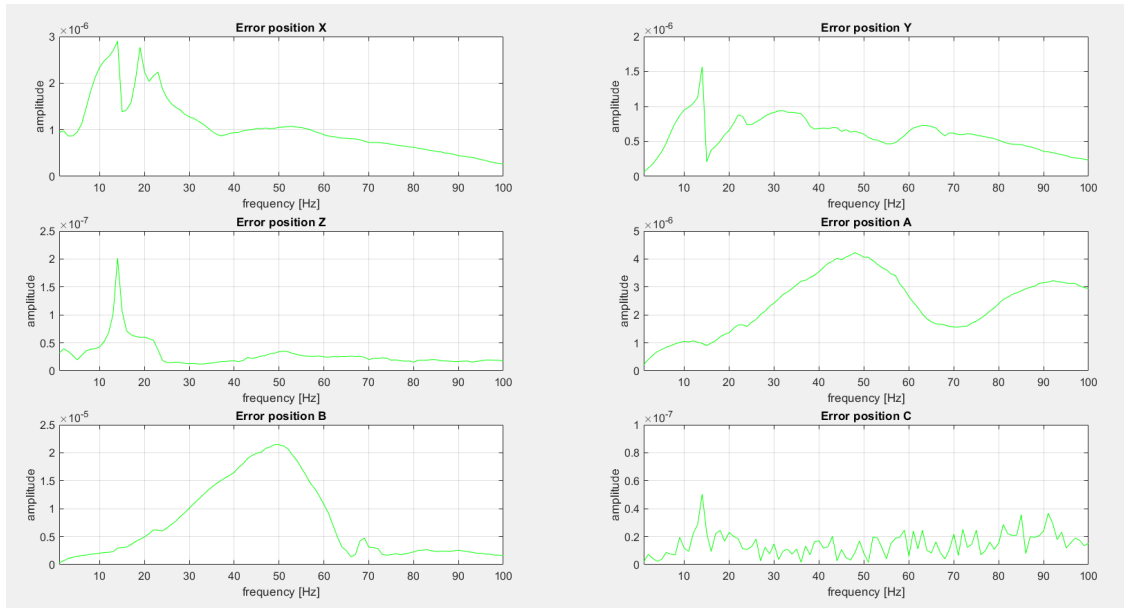


Figure 5.30: Frequency responses - carriage anomaly

The interpolation process has been carried out respecting the same hypothesis and procedure described earlier and the selected axis to carry out the analysis from now on is the X one.

5.2.1 Data analysis

The problematics that emerged previously also here were present. In order to refresh the mind, here are illustrated again with reference to the actual case study:

1. the machines in normal condition had a behavior that was similar but not identical between each other.

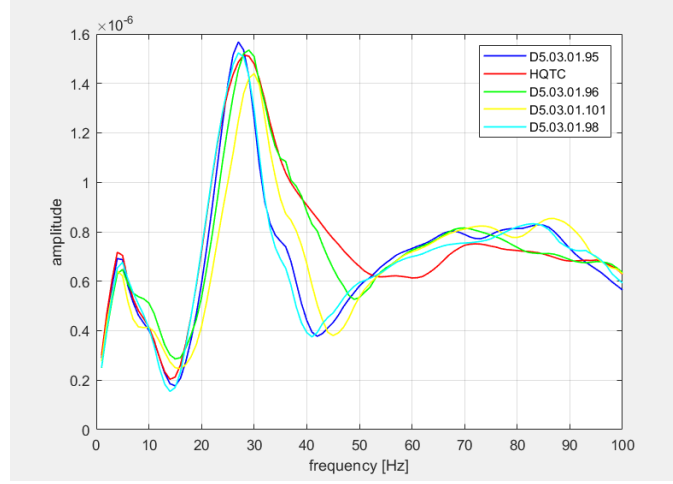
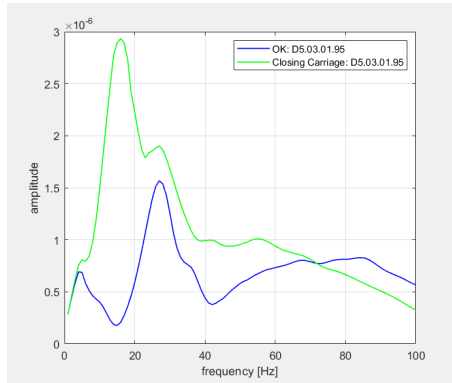
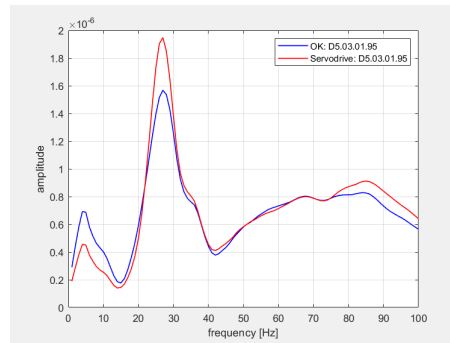


Figure 5.31: Correct responses of more machines overlapped

2. the machine in anomaly conditions had a frequency response whose can be very different (closing carriage problem) but also similar (setup servo drive anomaly), in most of the considered frequencies, when compared with the response in normal condition.



(a) ok vs closing carriage



(b) ok vs setup servo drive

Figure 5.32: Anomalies responses - comparison

5.2.2 Multi-phases

To solve these situations, the same solution of multi-phases accelerometer has been adopted: the entire problem has been splitted in two phases introducing the reference signal.

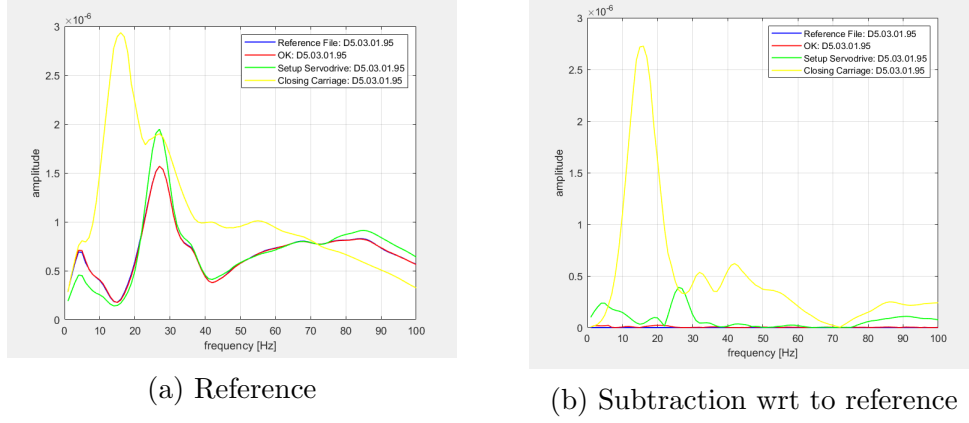


Figure 5.33: Reference file

So the aim, as before, is primarily to identify whether and, only after, which problem is occurring:

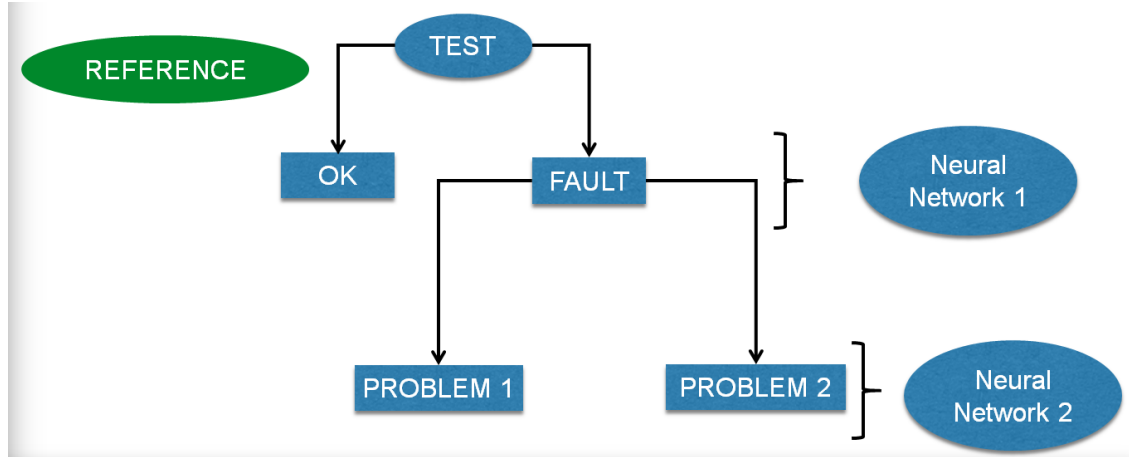


Figure 5.34: General scheme

Neural Networks

For this purpose, two datasets and as many neural networks have been created:

- the first one included data corresponding to correct condition, painted in blue,

and anomaly one (mix of closing carriage anomaly and servodrive anomaly) painted in red.

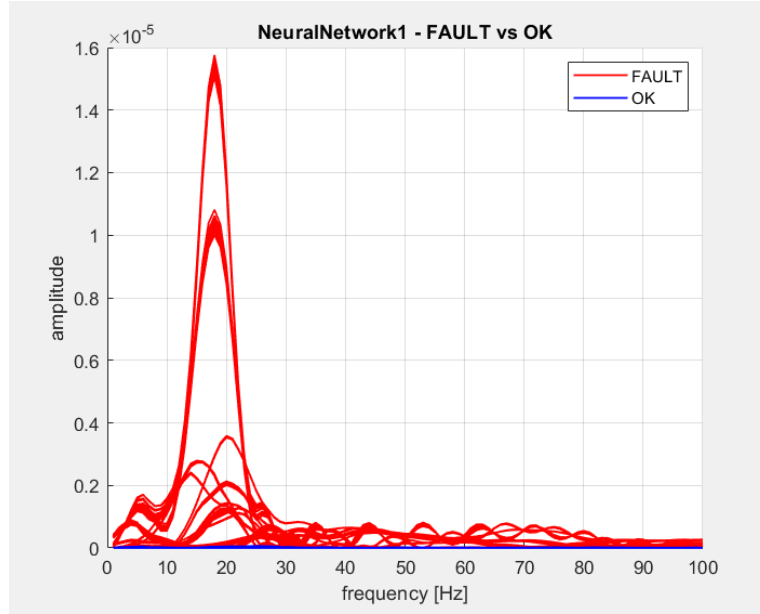


Figure 5.35: Neural network dataset - first phase

- the second one is used to distinguish the two anomaly cases: the servo drive expressed in blue and closing carriage in red.

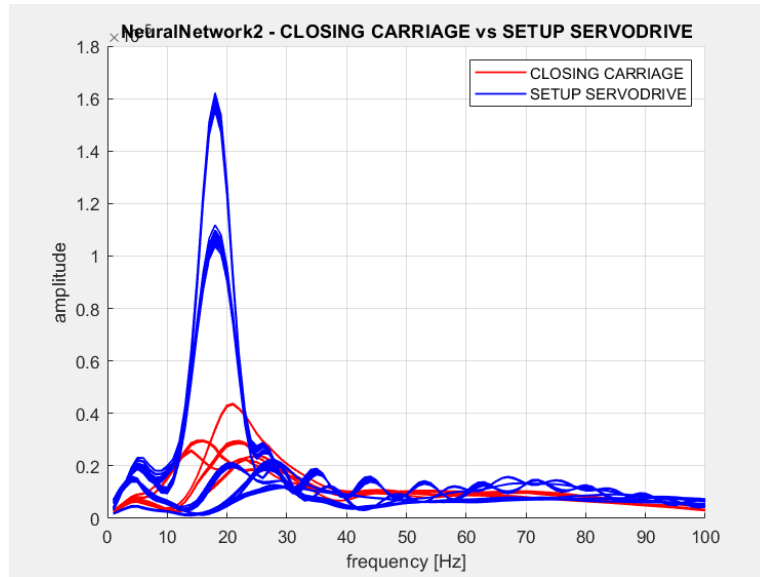


Figure 5.36: Neural network dataset - second phase

The respective neural networks had good accuracy and precision as emerged from the respective Confusion matrix and ROC:

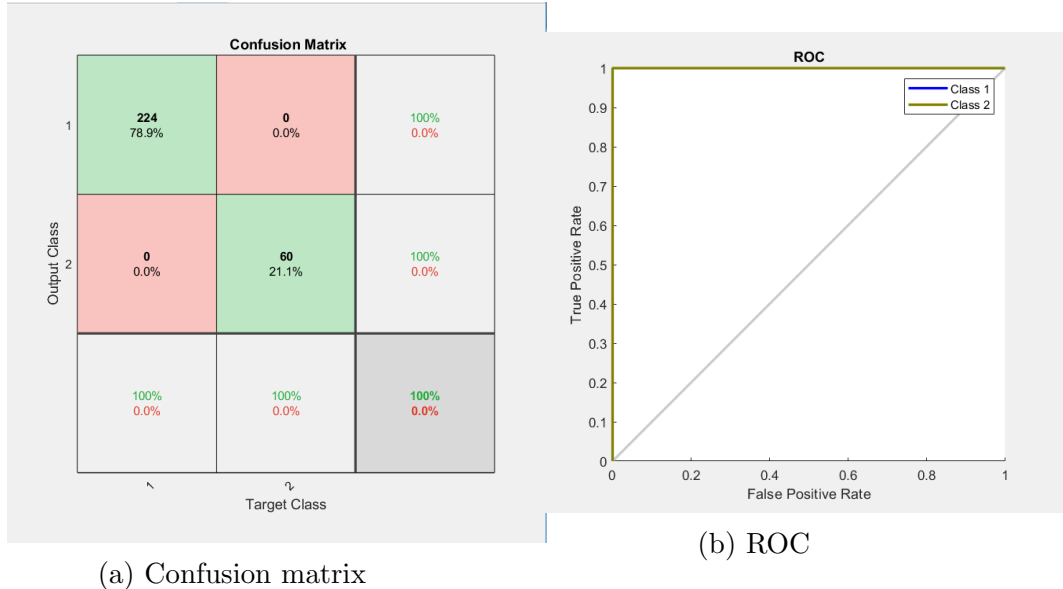


Figure 5.37: Confusion matrix and ROC - first phase

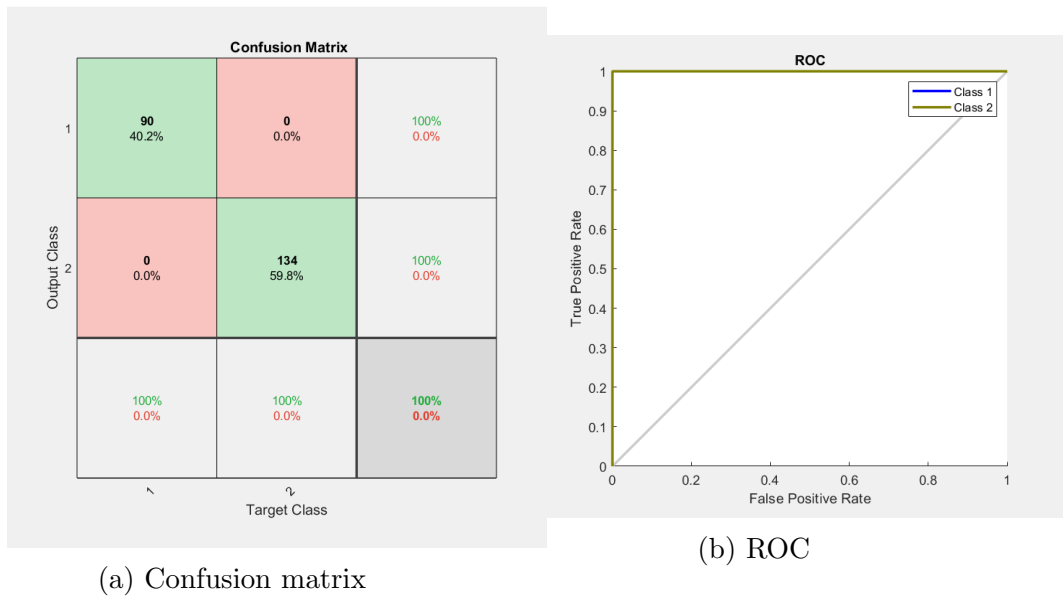


Figure 5.38: Confusion matrix and ROC - second phase

Test

The testing script implementation has followed the same procedure of the multi-phases accelerometer one.

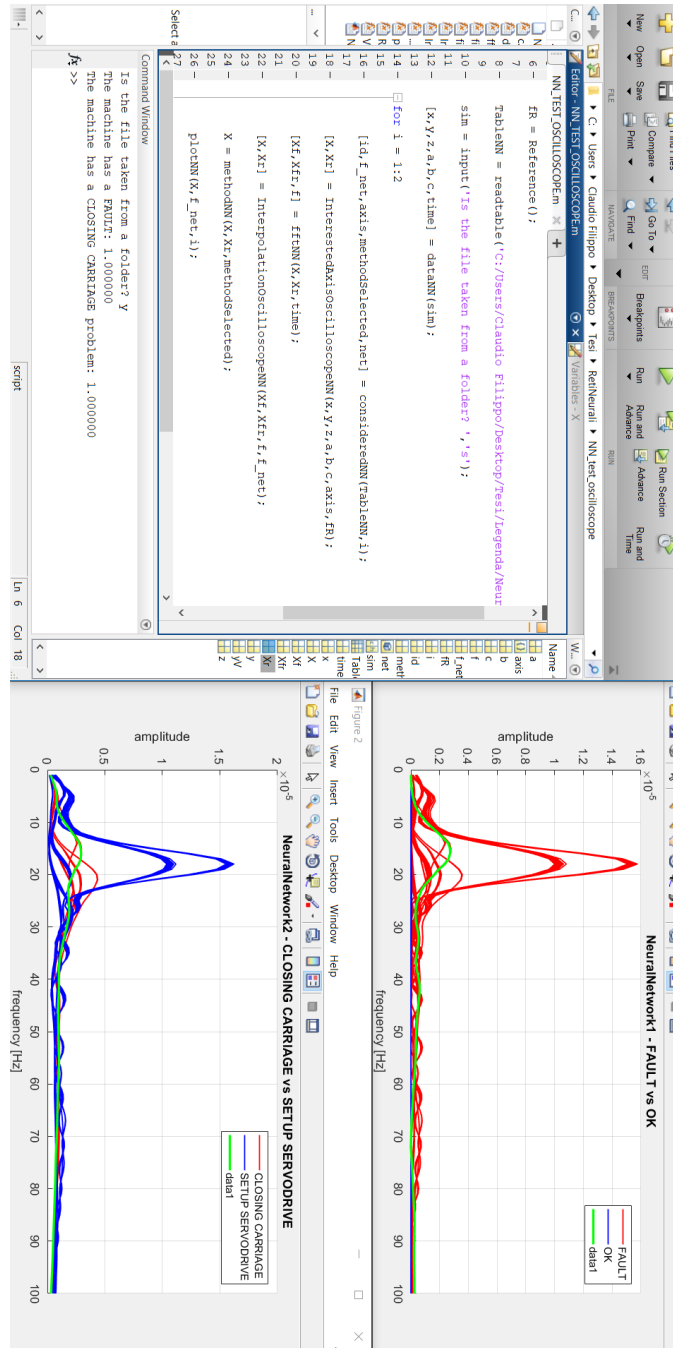


Figure 5.39: Test - case study 2

Chapter 6

Conclusion

The predictive maintenance is something born in the last few years and, surely, it is a very powerful tool if developed correctly and continuously. For this reason, not finding specific references in literature, this thesis had as main target to guarantee the feasibility of the whole method.

Therefore, some considerations and choices have been taken a-priori during the discussion, as result of theoretical hypothesis without really know if they would work in practice: for example the axes starting position, types of sensors and the accelerometer position are only some of the assumptions considered.

In the end, the obtained results look very positive since this tool, in both case studies, is able to recognize all situations addressed so far.

Obviously, this thesis represents only the first step of implementation of the predictive maintenance and, now, will be illustrated the next steps and possible scenarios.

6.1 Next steps

In the short and long time the following achievements can be obtained:

- Run intensive training and testing phases to add accuracy and complexity to our classification: until now, only two problems have been considered but, in the future, add new cases must become a priority. Moreover, add tests for each single case can help the classifier to become more efficient and more independently, with the possibility to avoid further mathematical operations that slow the model.
- Develop industrial setup: from company point of view, this tool allows to develop computational unit, install new sensors on machine and implement new application for the CNC program in order to obtain a better and more efficient item.

- Intellectual property protection: this project is subject a patent and, therefore, it goes to increase the resources and the knowledges owned by the company.
- New application fields: once this tool produce a result, it is necessary to close the loop and to find a system in order to fix eventually the machine. In this field, several scenarios can be taken into account: for instance, if the occurred problem is possible to solve via software, a solution can be implemented that instructs the system to automatically make the correct changes to come back to the normal situation. Otherwise, if a mechanical problem occurs, then the augmented reality field can enter in the process showing to the technical which is the best procedure to fix the situation.

Appendix A

JAVA application - code

A.1 JavaAccelerometer

```
1 package javaaccelerometer;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.io.PrintWriter;
6 import java.nio.file.Files;
7 import java.nio.file.Paths;
8 import java.nio.file.StandardCopyOption;
9 import java.util.ArrayList;
10
11 public class JavaAccelerometer {
12
13     static String fileData;
14     static String namePort;
15     static int timeout;
16     static double triggerLevel;
17     static int trSel;
18
19     static Thread comThread;
20
21     static boolean stopThread = false;
22     static int timeOutStep = 1000;
23
24     public static void main(String[] args) throws IOException,
25         InterruptedException{
26         String filePath = "cmd.txt";
27
```

```
28
29     try{
30         ArrayList<Command> commands = FileHelper.readFile(filePath);
31         for (Command c : commands){
32             fileData = c.getFileData();
33             namePort = c.getPortName();
34             timeout = c.getTimeout();
35             triggerLevel = c.getTriggerLevel();
36             trSel = c.getTriggerType();
37
38
39             System.out.println("File already present: " + fileData);
40         }
41
42     }catch(IOException e){
43         System.err.println(e);
44     }
45
46
47     if (fileData!=null)
48     {
49         Files.copy(Paths.get(fileData), Paths.get("data.csv"),
50             StandardCopyOption.REPLACE_EXISTING);
51     }
52     else
53     {
54         System.out.println("Port name: " + namePort + "; TriggerLevel: "
55             + triggerLevel + "; trSel: " + trSel);
56
57         comThread = new Thread(new Runnable() {
58
59             @Override
60             public void run() {
61                 try{
62
63                     Sequoia.runSequoia(triggerLevel,trSel - 1,namePort);
64
65
66                 }catch(Exception ex){
67                     System.out.println("Errore sull'esecuzione di
68                         sequoia: " + ex);
69                 }
70             });
```

```
71
72     comThread.setPriority(Thread.MAX_PRIORITY);
73     comThread.start();
74
75     boolean wellExit = false;
76
77     for (int i = 0; i < timeout; i++)
78     {
79
80         if (timeout - i < 6)
81         {
82             System.out.println(String.valueOf(timeout - i) + "
83                 Seconds before timeout exit...");
84         }
85
86         Thread.sleep(timeOutStep);
87
88         if (!comThread.isAlive())
89         {
90             wellExit = true;
91             break;
92         }
93     }
94
95     if (!wellExit)
96     {
97         System.out.println("Time expired...aborting process...");
98
99         stopThread = true;
100
101         System.out.println("Writing dump data to file...");
102
103         File file = new File("data.csv");
104         PrintWriter outputFile = new PrintWriter(file);
105
106         for (int h = 0; h < 10; h++)
107         {
108             outputFile.println("0;0;0;0;");
109         }
110
111
112         System.out.println("Dump data writed to file...");
113     }
114 }
115 }
```

116 }

A.2 Sequoia

```
1
2 package javaaccelerometer;
3
4 import java.io.FileNotFoundException;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import jssc.SerialPortException;
8
9
10 public class Sequoia{
11
12     static SerialPortCommunication port = new SerialPortCommunication();
13
14     public static void runSequoia (double triggerLevel, int trSel, String
        PortName) throws FileNotFoundException, InterruptedException,
        IOException, SerialPortException {
15
16         try
17     {
18         ArrayList<String> sel = new ArrayList<>();
19         sel.add("X");
20         sel.add("Y");
21         sel.add("Z");
22
23
24         System.out.println("Trigger Direction = " + sel.get(trSel) + " --> "
            + trSel);
25         System.out.println("Trigger Level = " + String.valueOf(triggerLevel)
            + " m/s^2");
26         System.out.println("Waiting for trigger...");
27
28         //inizializzazione del thread di lettura
29         CalibrationAccelerometer calAcc = new
            CalibrationAccelerometer();
30         calAcc.CalibrationMatrix();
31         calAcc.CalibrationVector();
32         calAcc.Sensivity();
33
34         try{
35             port.WriteReadSerialPort(trSel, triggerLevel, PortName);
```

```
36         } catch (NullPointerException ex){
37             System.out.println("Non lavora: " + ex);
38         }
39
40     }
41     catch (RuntimeException exc)
42     {
43         System.out.println(exc);
44     }
45 }
46
47 }
```

A.3 SerialPortCommunication

```
1 package javaaccelerometer;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5 import java.io.PrintWriter;
6 import java.time.LocalDateTime;
7 import java.util.ArrayList;
8
9 import jssc.SerialPort;
10 import jssc.SerialPortException;
11
12
13 public class SerialPortCommunication {
14
15     static boolean stopThread = false;
16     static boolean triggerOn = true;
17     static boolean triggerCatch = false;
18     static int preTriggerCount = 192;
19     static int postTriggerCount = 8000;
20     static int bufferSize = 81920;
21     static int cutSize = 8192;
22     static int triggerPos = 0;
23     static int triggerCount = 0;
24     static double sampleTime = 0.0001220703125;
25
26     static SerialPort port;
27     static CalibrationAccelerometer calibAccel = new
        CalibrationAccelerometer();
28     static AccelerometerComponents accComp = new AccelerometerComponents();
```

```
29  static ArrayList<ArrayList<Double>> accBuffer = new ArrayList<>();
30  static ArrayList<Double> accBufferList = new ArrayList<>();
31
32
33
34  public SerialPortCommunication(){
35
36      this.port = null;
37  }
38
39
40  public void WriteReadSerialPort (int trSel, double triggerLevel, String
    PortName)
41      throws SerialPortException, InterruptedException,
        FileNotFoundException {
42
43      port = new SerialPort(PortName);
44
45      port.openPort();
46
47      int baudRate = 921600;
48      port.setParams(baudRate,
49                    SerialPort.DATABITS_8,
50                    SerialPort.STOPBITS_1,
51                    SerialPort.PARITY_NONE);
52
53      port.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
54
55      port.writeString("AC");
56      byte[] floatAccel_ = new byte[2];
57
58      floatAccel_ = port.readBytes(2);
59
60
61      if ((char)floatAccel_[0] == 'A')
62      {
63
64      }
65      else
66      {
67          port.closePort();
68          Thread.sleep(1000);
69          port.openPort();
70
71          port.writeString("AC");
72          floatAccel_ = new byte[2];
```

```
73     floatAccel_ = port.readBytes(2);
74 }
75
76 while (true)
77 {
78     if (stopThread)
79     {
80         break;
81     }
82
83     int sizeRead = 0;
84
85     byte[] floatAccelBuffer = new byte[sizeRead*5];
86
87     byte[] floatAccel = new byte[5];
88
89     byte[] dumpStep = new byte[1];
90
91     ///lettura dei dati dalla porta seriale
92
93     while (true)
94     {
95
96         sizeRead = port.getInputBufferBytesCount();
97         sizeRead = sizeRead / 5;
98         sizeRead = sizeRead * 5;
99
100        if (sizeRead < 5)
101        {
102            continue;
103        }
104
105        floatAccelBuffer = new byte[sizeRead];
106
107        floatAccelBuffer = port.readBytes(sizeRead);
108
109        byte XH = (byte)(floatAccelBuffer[0] & 0x0F);
110
111        byte XL = (byte)(floatAccelBuffer[1] & 0x0F);
112
113        byte YH = (byte)((floatAccelBuffer[2] & 0xF0) >>> 4);
114
115        byte YL = (byte)((floatAccelBuffer[3] & 0xF0) >>> 4);
116
117        byte ZH = (byte)(floatAccelBuffer[3] & 0x0F);
```

```
118
119     byte ZL = (byte)(floatAccelBuffer[4] & 0x0F);
120
121
122     byte chkSumCalc = (byte)(XH + XL + YH + YL + ZH + ZL);
123
124     chkSumCalc = (byte)(chkSumCalc & 0X0F);
125
126     byte chkSum = ((byte)((floatAccelBuffer[0]& 0xF0) >>> 4));
127
128
129
130
131     if (chkSumCalc == chkSum)
132     {
133         break;
134
135     }
136     else
137     {
138         System.out.println("Checksum Fault during Triggering -> " +
139             LocalDateTime.now().toString());
140         dumpStep = new byte[1];
141         dumpStep = port.readBytes(1);
142     }
143
144     ///fine lettura
145
146
147     for (int i_ = 0; i_ < sizeRead; i_ = i_ + 5)
148     {
149
150         for (int j_ = 0; j_ < 5; j_++)
151         {
152             floatAccel[j_] = floatAccelBuffer[i_ + j_];
153
154         }
155
156         ///inizio scorporo delle accelerazioni dai bytes ricevuti
157         e li inserisco nei buffer delle 3 accelerazioni
158
159         accBuffer = accComp.getValue(floatAccel, calibAccel);
160
161         accBufferList = accBuffer.get(trSel);
```



```
162
163     ///fine scorporo
164
165
166     ///verifico se avviene il trigger
167     if ((triggerOn) && (triggerCatch == false))
168     {
169         if (accBufferList.size() > preTriggerCount)
170         {
171             if (accBufferList.get(accBufferList.size() - 2) <=
172                 triggerLevel)
173             {
174                 if (accBufferList.get(accBufferList.size() - 1)
175                     > triggerLevel)
176                 {
177                     triggerCatch = true;
178                     triggerPos = accBufferList.size() - 1;
179                     triggerCount = 0;
180                     System.out.println("Trigger Catch...!!!");
181                 }
182             }
183         }
184
185
186     ///fine trigger
187
188
189     ///inizio gestione dei dati da quando viene catturato il
190     trigger
191
192     if (triggerCatch)
193     {
194         triggerCount = triggerCount + 1;
195
196         if (triggerCount > postTriggerCount)
197         {
198             int a = triggerPos;
199
200             int c = preTriggerCount + postTriggerCount;
201             int d = accBuffer.get(0).size();
202
203             System.out.println("Data Ready...writing to
204                             file....");
```

```
204
205     ArrayList<Double> xd = accBuffer.get(0);
206     ArrayList<Double> yd = accBuffer.get(1);
207     ArrayList<Double> zd = accBuffer.get(2);
208
209     ArrayList<Double> xData = new
        ArrayList<>(xd.subList(triggerPos -
            preTriggerCount, triggerPos +
                postTriggerCount));
210     ArrayList<Double> yData = new
        ArrayList<>(yd.subList(triggerPos -
            preTriggerCount, triggerPos +
                postTriggerCount));
211     ArrayList<Double> zData = new
        ArrayList<>(zd.subList(triggerPos -
            preTriggerCount, triggerPos +
                postTriggerCount));
212
213     File file = new File("data.csv");
214     PrintWriter outputFile = new PrintWriter(file);
215
216     for (int h = 0; h < xData.size(); h++)
217     {
218         outputFile.println(String.valueOf((new Double(h
            * sampleTime)).toString() + "," +
            xData.get(h).toString() + "," +
            yData.get(h).toString() + "," +
            zData.get(h).toString()));
219     }
220
221     outputFile.close();
222
223     System.out.println("Data has been writed to file!
        Exit process...");
224
225     stopThread = true;
226
227     triggerPos = 0;
228     triggerCatch = false;
229
230
231     triggerOn = false;
232
233     triggerCatch = false;
234
235 }
```

```
236         }
237
238
239         ///fine gestione dati
240
241     }
242
243     ///inizio rimozione dal buffer dei campioni in eccesso
244
245     if (accBuffer.get(0).size() > bufferSize - 1)
246     {
247         for (int i = 0; i < cutSize + 1; i++) {
248
249             ArrayList<Double> a = accBuffer.get(0);
250             a.remove(i);
251
252             ArrayList<Double> b = accBuffer.get(1);
253             b.remove(i);
254
255             ArrayList<Double> c = accBuffer.get(2);
256             c.remove(i);
257
258         }
259
260         if (triggerCatch)
261         {
262             triggerPos = triggerPos - cutSize;
263         }
264     }
265
266     ///fine rimozione
267
268 }
269
270 }
271
272 }
```

A.4 CalibrationAccelerometer

```
1
2 package javaaccelerometer;
3
4 public class CalibrationAccelerometer {
```

```
5
6     private float[] calibVector;
7     private float[] calibMatrix;
8     private float sensivity;
9
10
11     public CalibrationAccelerometer() {
12         this.calibMatrix = null;
13         this.calibVector = null;
14         this.sensivity = 0;
15     }
16
17     public float[] CalibrationVector () {
18
19         calibVector = new float[3 + 2];
20
21         calibVector[0] = (float)(32.15645 * 2);
22         calibVector[1] = (float)(8.031253 * 2);
23         calibVector[2] = (float)(8.000053 * 2);
24
25         return calibVector;
26     }
27
28
29
30
31     public float[] CalibrationMatrix() {
32
33         calibMatrix = new float[9];
34
35         calibMatrix[0] = (float)(0.5000134 * 2);
36         calibMatrix[1] = (float)(-0.001953125 * 2);
37         calibMatrix[2] = (float)(0.001953125 * 2);
38         calibMatrix[3] = (float)(-0.001955064 * 2);
39         calibMatrix[4] = (float)(0.5000114 * 2);
40         calibMatrix[5] = (float)(-0.001953136 * 2);
41         calibMatrix[6] = (float)(-0.00781264 * 2);
42         calibMatrix[7] = (float)(0.03134206 * 2);
43         calibMatrix[8] = (float)(0.5027562 * 2);
44
45         return calibMatrix;
46     }
47
48
49     public float Sensivity(){
50
```

```
51     sensivity = (float)(26.8);
52
53     return sensivity;
54 }
55
56
57
58
59 }
```

A.5 AccelerometerComponents

```
1
2 package javaaccelerometer;
3
4 import static java.lang.Math.pow;
5 import java.util.ArrayList;
6
7 public class AccelerometerComponents {
8
9     static ArrayList<ArrayList<Double>> accBuffer = new ArrayList<>();
10
11     public AccelerometerComponents(){
12         accBuffer.add(new ArrayList<>()); //acc X
13         accBuffer.add(new ArrayList<>()); //acc Y
14         accBuffer.add(new ArrayList<>()); //acc Z
15     }
16
17
18
19     public ArrayList<ArrayList<Double>> getValue(byte[] floatAccel,
20         CalibrationAccelerometer calAcc){
21
22         short[] acc = new short[5];
23         short xAcc_;
24         short yAcc_;
25         short zAcc_;
26
27         float xAcc;
28         float yAcc;
29         float zAcc;
30
31         float[] calibMatrix = calAcc.CalibrationMatrix();
```

```
32     float sensivity = calAcc.Sensivity();
33     float[] calibVector = calAcc.CalibrationVector();
34
35
36     for (int i=0; i<5; i++)
37     {
38
39         if (floatAccel[i] < 0)
40         {
41             acc[i] = (short)(floatAccel[i] + 0xFF +1);
42         }
43         else
44         {
45             acc[i] = (short)(floatAccel[i]);
46         }
47     }
48
49
50
51
52     xAcc_ = (short)((short)((short)((acc[0]) << 8)) |
53              (short)(acc[1]));
54     yAcc_ = (short)((short)((short)((acc[2]) << 4)) |
55              (short)((acc[3]) >> 4));
56     zAcc_ = (short)((short)((short)((acc[3]) << 8)) |
57              (short)(acc[4]));
58
59
60
61
62     if (xAcc_ <0)
63     {
64         xAcc = (float)(pow(2,16)+ xAcc_);
65         xAcc = (float)(xAcc - 2048) + calibVector[0];
66     }
67     else{
68         xAcc = (float)(xAcc_ - 2048) + calibVector[0];
69     }
70
71     if (yAcc_ <0)
72     {
73         yAcc = (float)(pow(2,16)+ yAcc_);
74         yAcc = (float)(yAcc - 2048) + calibVector[1];
```

```
75         }
76         else{
77             yAcc = (float)(yAcc_ - 2048) + calibVector[1];
78         }
79
80
81         if (zAcc_ <0)
82         {
83             zAcc = (float)(pow(2,16)+ zAcc_);
84             zAcc = (float)(zAcc - 2048) + calibVector[2];
85         }
86         else{
87             zAcc = (float)(zAcc_ - 2048) + calibVector[2];
88         }
89
90
91
92
93         accBuffer.get(0).add((double)(((xAcc * calibMatrix[0] + yAcc *
94             calibMatrix[1] + zAcc * calibMatrix[2]) / sensivity)));
95         accBuffer.get(1).add((double)(((xAcc * calibMatrix[3] + yAcc *
96             calibMatrix[4] + zAcc * calibMatrix[5]) / sensivity)));
97         accBuffer.get(2).add((double)(((xAcc * calibMatrix[6] + yAcc *
98             calibMatrix[7] + zAcc * calibMatrix[8]) / sensivity)));
99
100         return accBuffer;
101     }
102
103
104 }
```

A.6 Command

```
1 package javaaccelerometer;
2
3 public class Command {
4     private String portName;
5     private String triggerType;
6     private double triggerLevel;
7     private int timeout;
8     private String fileData;
```

```
9
10 public static Command parseData(String data){
11     String[] splittedData = data.split(";");
12
13     Command c = new Command();
14     c.setPortName(splittedData[0]);
15     c.setTriggerType(splittedData[1]);
16     c.setTriggerLevel(splittedData[2]);
17     c.setTimeout(splittedData[3]);
18
19     if (splittedData.length == 5){
20         c.setFileData(splittedData[4]);
21     }
22
23     return c;
24 }
25
26 private Command(){
27 }
28
29 public Command(String portName, String triggerType, int triggerLevel,
30     int timeout, String fileData) {
31     this.portName = portName;
32     this.triggerType = triggerType;
33     this.triggerLevel = triggerLevel;
34     this.timeout = timeout;
35     this.fileData = fileData;
36 }
37
38 public String getPortName() {
39     return portName;
40 }
41
42 public void setPortName(String portName) {
43     this.portName = portName;
44 }
45
46 public int getTriggerType(){
47
48     if (triggerType.equals("X"))
49     {
50         return 1;
51     }
52     else if (triggerType.equals("Y"))
53     {
```



```
54         return 2;
55     }
56     else if (triggerType.equals("Z"))
57     {
58         return 3;
59     }
60     else
61     {
62         return 1;
63     }
64 }
65
66
67
68
69 public void setTriggerType(String triggerType) {
70     this.triggerType = triggerType;
71 }
72
73
74 public double getTriggerLevel() {
75     return triggerLevel;
76 }
77
78 public void setTriggerLevel(double triggerLevel) {
79     this.triggerLevel = triggerLevel;
80 }
81
82 public void setTriggerLevel(String triggerLevel) {
83     this.triggerLevel = Integer.parseInt(triggerLevel);
84 }
85
86
87
88 public int getTimeout() {
89     return timeout;
90 }
91
92 public void setTimeout(int timeout) {
93     this.timeout = timeout;
94 }
95
96 public void setTimeout(String timeout) {
97     this.timeout = Integer.parseInt(timeout);
98 }
99
```

```
100
101
102     public String getFileData() {
103         return fileData;
104     }
105
106     public void setFileData(String fileData) {
107         this.fileData = fileData;
108     }
109
110     @Override
111     public String toString() {
112         return "Command{" + "portName=" + portName + ", triggerType=" +
            triggerType + ", triggerLevel=" + triggerLevel + ", timeout=" +
            timeout + ", fileData=" + fileData + '}';
113     }
114 }
```

A.7 FileHelper

```
1 package javaaccelerometer;
2
3 import java.io.BufferedReader;
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.io.FileReader;
7 import java.io.IOException;
8 import java.util.ArrayList;
9
10 public class FileHelper {
11
12     public static ArrayList<Command> readFile(String filePath)throws
        FileNotFoundException, IOException{
13         ArrayList<Command> fileContent = new ArrayList<>();
14
15         File file = new File(filePath);
16         FileReader fr = new FileReader(file);
17         BufferedReader br = new BufferedReader(fr);
18         String buffer;
19
20         while ((buffer = br.readLine()) != null) {
21             if(!buffer.isEmpty()){
22                 fileContent.add(Command.parseData(buffer));
23             }
24         }
25     }
26 }
```

```
24     }  
25     br.close();  
26     fr.close();  
27  
28     return fileContent;  
29 }  
30  
31 }
```

Bibliography

- [1] Intelligenza artificiale (blog). *Reti Neurali*. URL: <http://www.intelligenzaartificiale.it/reti-neurali/>.
- [2] A.Magnani. “Perché si parla tanto di industria 4.0: che cos’è e quanti lavori può creare”. In: *Il Sole 24 ore* (2017).
- [3] A.Moawad. “Neural networks and back-propagation explained in a simple way”. In: (2018).
- [4] A.Pedrazzini. “L’industria 4.0 è un’occasione per valorizzare persone e competenze”. In: *Il Sole 24 ore* (2018).
- [5] F.Baena et al. “Learning Factory: The Path to Industry 4.0”. In: (2017).
- [6] K.Santos et al. “Opportunities Assessment of Product Development Process in Industry 4.0”. In: (2017), pp. 1358–1365.
- [7] M.A.A.K.Bahrin et al. “Industry 4.0: A review on industrial automation and robotic”. In: *Jurnal Teknologi* (2016).
- [8] B.Capehart. *Automated Diagnostics and Analytics for Buildings*. Ed. by Fairmont Press. 2014.
- [9] M.H. Beale, M.T. Hagan, and H.B Demuth. *Neural Networks Toolbox: Getting Started Guide*. MathWorks, 2018.
- [10] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1996.
- [11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.
- [12] C.M.Florkowski. “Sensitivity, Specificity, Receiver-Operating Characteristic (ROC) Curves and Likelihood Ratios: Communicating the Performance of Diagnostic Tests”. In: *The Clinical Biochemist Reviews* 4 (2008).
- [13] C.Souza. “Discriminatory Power Analysis by Receiver-Operating Characteristic Curves”. In: (2009).
- [14] R.E.Schapire E.L.Allwein and Y.Singer. “Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers”. In: *J. Mach. Learn. Res.* 1 (2001), pp. 113–141.

- [15] Jimin He and Zhi-Fang Fu, eds. *Modal Analysis*. Butterworth-Heinemann, 2001.
- [16] C.Niezrecki J.Baqersad P.Poozesh and P.Avitabile. “Comparison of Modal Parameters Extracted Using MIMO, SIMO, and Impact Hammer Tests on a Three-Bladed Wind Turbine”. In: *Topics in Modal Analysis II, Volume 8*. Ed. by Allemang R. Springer International Publishing, 2014.
- [17] K.Smith. “Reactive vs Preventive vs Predictive Maintenance”. In: *VIZIYA (blog)* ().
- [18] L.Li L.D.Xu E.L.Xu. “Industry 4.0: state of the art and future trends”. In: *International Journal of Production Research* 56.8 (2018), pp. 2941–2962.
- [19] M.Bartoloni. “I robot distruggono posti di lavoro? In Germania è vero il contrario”. In: *Il Sole 24 ore* (2018).
- [20] M.Beyeler. “How to choose the right algorithm for your machine learning problem”. In: (2017).
- [21] M.Cifalinò. “La manutenzione negli impianti industriali”. In: *Impianto.it* (). URL: <https://www.impianto.it/manutenzione-impianti-industriali/>.
- [22] M.Gori. “Introduzione alle reti neurali artificiali”. In: *Mondo digitale* 4 (2006).
- [23] M.Mijwel. “Artificial Neural Networks Advantages and Disadvantages”. In: (2018).
- [24] M.Sanjeevi. “Different types of Machine learning and their types”. In: (2017).
- [25] MathWorks. URL: <https://www.mathworks.com/help/deeplearning/ref/plotconfusion.html>.
- [26] S Ryszard Michalski, G Jaime Carbonell, and M Tom Mitchell, eds. *Machine Learning an Artificial Intelligence Approach Volume II*. Morgan Kaufmann Publishers Inc., 1986.
- [27] Thomas M. Mitchell. *Machine Learning*. 1997.
- [28] N.Boldrini. “Cos’è il Machine Learning, come funziona e quali sono le sue applicazioni”. In: *AI4Business* (2017).
- [29] N.M.M.Maia and J.Silva. “Modal analysis identification techniques”. In: 359 (Jan. 2001), pp. 29–40.
- [30] N.M.Živković. *How do Artificial Neural Networks learn?* 2018. URL: <https://www.codeproject.com/Articles/1225385/How-do-Artificial-Neural-Networks-Learn>.
- [31] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.
- [32] P.Louridas and C.Ebert. “Machine Learning”. In: *IEEE Software* 33.5 (2016), pp. 110–115.

- [33] R.K.Mobley. *An Introduction to Predictive Maintenance*. Ed. by Elsevier. 2002.
- [34] S.Mousa R.Morarr H.Arman. “The Fourth Industrial Revolution (Industry 4.0): A Social Innovation Perspective”. In: *Technology Innovation Management Review* 7 (2017), pp. 12–20.
- [35] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag, 1996. Chap. 7: The backpropagation algorithm.
- [36] S.Bhatt. “5 Things You Need to Know about Reinforcement Learning”. In: (2018).
- [37] S.Lawrence and C.L.Giles. “Overfitting and Neural Networks: Conjugate Gradient and Backpropagation”. In: *International Joint Conference on Neural Networks* (2000), pp. 114–119.
- [38] S.Selcuk. “Predictive maintenance, its implementation and latest trends”. In: *Journal of engineering manufacture* (2017).
- [39] T.Fawcett. “An introduction to ROC analysis”. In: *Pattern Recognition Letters* 27.8 (2006). ROC Analysis in Pattern Recognition, pp. 861–874.
- [40] T.Foxworth. “Using IoT and Machine Learning for industrial Predictive Maintenance”. In: (2017).
- [41] DataFlair team. *Introduction to Artificial Neural Network Model*. 2017.
- [42] V.Gupta. “Understanding Feedforward Neural Networks”. In: (2017).
- [43] V.Jha. “Machine Learning Algorithm - Backbone of emerging technologies”. In: *TechLeer* (2017).
- [44] TED Institute - YouTube. “Markus Lorenz: Industry 4.0: how intelligent machines will transform everything we know”. In: (2015).