



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

On-site process monitoring and autonomous control of a CNC milling machine using CNN and Bayesian Optimization

TESI DI LAUREA MAGISTRALE IN
MECHANICAL ENGINEERING - INGEGNERIA MECCANICA

Author: **Filippo Danilo Michelacci**

Student ID: 964723

Advisor: Professor Annoni Massimiliano Pietro Giovanni

Co-advisors:

Academic Year: 2021-2022

Jarvis, sometimes you have to run, before you can walk.
- Tony Stark

Abstract

In this thesis, the feasibility of implementing a cheap and reliable monitoring system to check the quality of the machined surfaces was researched. The proposed system only needs a small camera mounted next to the tool. Different monitoring CNN architectures were trained and compared in order to choose the most suitable one. Additionally, an optimisation system using Bayesian optimisation was implemented to choose the best machining parameters a priori and use that knowledge to take actions when the machining results were not as expected, testing the model's performance as an online optimiser directly connected to the monitoring system. The created system, if further improved, will be capable of performing in an industrial machining environment and can easily be implemented, opening the possibility of introducing autonomous and automated systems even in the small to medium companies' reality.

Keywords: CNC machine, on site monitoring, online decision making, Convolutional Neural Networks, Bayesian Optimisation

Abstract in lingua italiana

In questa tesi è stata analizzata la fattibilità e possibilità di realizzare un sistema di controllo affidabile ed economico in grado di esaminare la qualità delle superfici fresate. Il sistema proposto è semplicemente formato da una telecamera USB montata vicino all'utensile e collegata a un computer. Differenti architetture di reti neurali convoluzionali sono state analizzate e testate al fine di sceglierne la migliore. In aggiunta, è stato realizzato un ottimizzatore Baesiano in grado di scegliere la miglior combinazione di parametri di taglio prima d'iniziare la lavorazione e capace d'intervenire durante il processo di fresatura, in quanto direttamente collegato al sistema di monitoraggio, qualora la qualità della superficie non fosse sufficiente. Il sistema, realizzato, se ulteriormente migliorato, potrà essere implementato in un contesto industriale offrendo la possibilità, anche per le piccole e medie imprese, di un allacciamento al paradigma dell'industria 4.0.

Parole chiave: Macchine Utensili CNC, controllo in tempo reale, controllo diretto, Reti Neurali Convoluzionali, Ottimizzazione Baesiana

Contents

Abstract	iii
Abstract in lingua italiana	iv
Contents	v
Introduction	1
0.1 A milling overview	1
0.1.1 Orthogonal cutting	1
0.1.2 From Orthogonal cutting to Milling	3
0.1.3 An introduction to milling tools	5
0.1.4 How can we measure the quality?	9
0.2 What is a CNC milling machine?	10
0.3 Why do we need CNC milling machines?	11
0.4 Issuing the problem	11
0.4.1 The machining parameters tuning problem	11
0.4.2 The monitoring problem	12
0.4.3 Previous related work	13
0.5 Thesis overview	16
0.5.1 Thesis's target	16
0.5.2 Thesis Structure	17
1 Computer Numerical Control machines & Artificial Neural Networks	18
1.1 Computer Numerical Control (CNC) milling machines	18
1.1.1 The G-Code	19
1.2 Artificial Neural Network (ANN)	21
1.2.1 Introduction	21
1.2.2 Basics Math and architectures	22
1.2.3 Convolutional Neural Networks (CNN)	24
2 Proposed Solution and its Application	27
2.1 Estimation method	27

2.1.1	Dataset creation	27
2.1.2	CNN estimation models	34
2.1.3	Training of the models	36
2.2	Optimisation method	39
2.2.1	Introduction to Bayesian Optimisation	39
2.2.2	Choosing the statistical model	40
2.2.3	Choosing the acquisition function	43
2.2.4	Assembling the optimiser	43
2.3	Assembling the final system	45
2.3.1	Modifying the G-Code	47
2.3.2	Parameters modification strategy	48
3	Results & Discussion	49
3.1	CNN Training	49
3.1.1	Xception using MSE loss function	49
3.1.2	Xception using MAPE loss function	51
3.1.3	Xception using Huber loss function	53
3.1.4	ResNet50 using MSE loss function	56
3.1.5	ResNet50 using MAPE loss function	58
3.2	CNN Models comparison	60
3.2.1	Xception models comparison	61
3.2.2	ResNet50 models comparison	63
3.2.3	Final model test performances	65
3.3	Static Optimisation	67
3.4	Combined system	70
4	Closing the Circle	71
4.1	Summary	71
4.2	Conclusions	72
4.3	Future Work	73
	Bibliography	74
A	Appendix A: Main Scripts	79
A.1	Machining Parameters Combination	79
A.2	Data Enhancement	81
A.3	Model Training	82
B	Appendix B: CNC Specifications	88

List of Figures	89
List of Tables	94
List of Most Important Symbols	95
List of Abbreviations	96
Acknowledgements	97

Introduction

Since the beginning of times, humans have always tried to overcome their limitations by pushing the innovation boundaries to the extreme, constantly challenging themselves. Observation was always one of the main characters in this evolution and most of the time represented the starting point of any newness. Even this thesis, which aims to improve and assess some of the major problems present in a manufacturing environment, was ignited after recalling many times spent observing CNC machines working in factories and in the laboratory. In detail, this work is focused on monitoring a CNC milling operation using Artificial Neural Networks (ANN), in particular Convolutional Neural Networks (CNN) which are part of the family of Machine Learning (ML). This constant inspection is then used by a controller unit that, using a Statistical Optimization algorithm called Bayesian optimization, automatically corrects the system to guarantee a final constant quality of the machined product.

This Introductory section was written to provide the reader with a general overview of the milling environment, giving an answer to some of the most common questions that may arise in his or her mind while imagining the process; the focus will then be shifted to some of the challenges that must be overcome to conduct an efficient and reliable machining operation, combined with the current solutions and their respective limitations, paving the way to the proposed approach.

0.1. A milling overview

Even during the Neanderthal Era, we realised that to create valuable objects, raw materials had to be processed with different techniques to gradually obtain the finished products. Among those, subtractive ones, in particular chopping, cutting, and chipping ..., were always intensively used. As technology evolved subtractive methods have become more and more accurate, efficient and reliable. This section is focused on introducing the reader to the basics of the milling paradigm which represents one of the most ancient and employed techniques among the whole family of subtractive manufacturing processes.

0.1.1. Orthogonal cutting

Imagine a knife cutting through butter or scraping the surface of an ice block. What do these processes have in common with a tool cutting its way through a block of steel on a milling

machine? All of them are paving their way through a solid mean that is opposing to be sheared. In both cases, the material is contrasting the action of the tool that is trying to detach some mass, called chips, from the main body. The simplest and most intuitive theory to model this process is called the *The Orthogonal Cutting Model* [1].

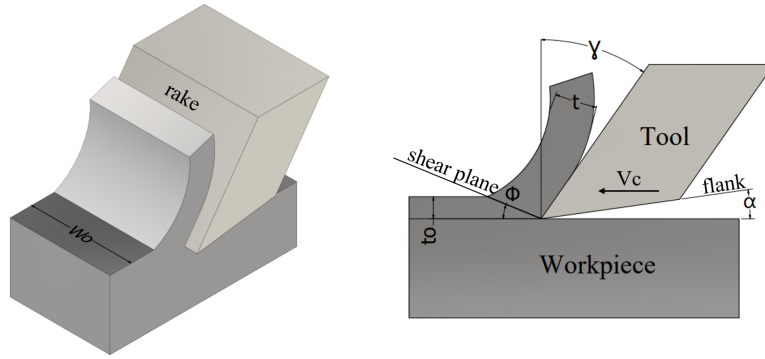


Figure 1: Sketch of an Orthogonal cutting model. The tool is moving and it is perpendicular to the piece along the cutting line.

As it can be seen from Figure 1 the chip is formed and moves along the *rake* of the tool. Compression and shear are both present during the cutting operation; the first can be noticed because the shear plane is formed and scales are generated along its length due to the deformations induced on that plane. The compression load is instead generated by the *flank* of the tool that is scraping the machined surface. In orthogonal cutting the *cutting speed*, V_c , is linear and is possessed by the tool, if this is moving, or by the piece, in case the other is still. This cutting parameter is very important because influences the chip formation and the power loss due to the friction generated during the cutting operation, that leads to a temperature rise changing the material properties, consequently it has to be chosen wisely according to the tool and workpiece materials. Another important parameter is the *feed*, f , which in orthogonal cutting is usually fixed because measured along the uncut chip thickness (t_0). This is correlated to the feed rate (also called feed in the machining environment), V_f , which usually identifies how fast is the tool moving along the feed direction in contact with the workpiece. Last but not least there is the depth of cut which in Figure 1 is identified with the width of cut W_0 . Even if the tool used in this orthogonal model possesses simple geometries and cutting profiles it is very useful to make the reader understand the most important angles. As it can be noticed from the picture two of the main were represented: the rake angle, γ , and the flank one or clearance angle, α . In order to identify these two parts of the tool the chip formation must be observed so that the *flank* and the *rake* can be identified. While the chip is formed its crystalline planes are sliding on each other because of the shear action that is induced by the tool (Figure 2). This process starts on the *shear plane* identified with the shear angle, ϕ . The observation of the chip formation is very important because it is a rapid and effective way to determine how the tool is cutting. Its colour [2], length and shape are key features that must be considered and can give qualitative information regarding the condition of the tool, the temperature and friction that is

generated on the contact point and the quality of the material that has been cut. The formation of long chips must always be avoided because can lead to injuries and bad final quality as they can get stuck inside the machine and cause additional scratches and marks on the machined surface. Although much more can be said about the cutting phenomena, including the whole force discussion, what was explained before is enough to prepare the reader to understand the basics of a milling process and the core identity of this work.

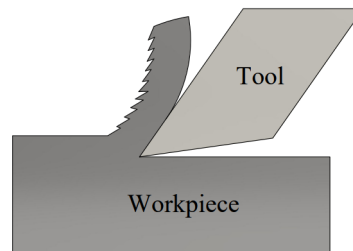


Figure 2: Sketch of a more realistic Orthogonal cutting model. The sliding between planes can be noticed and the chip deformation is added.

0.1.2. From Orthogonal cutting to Milling

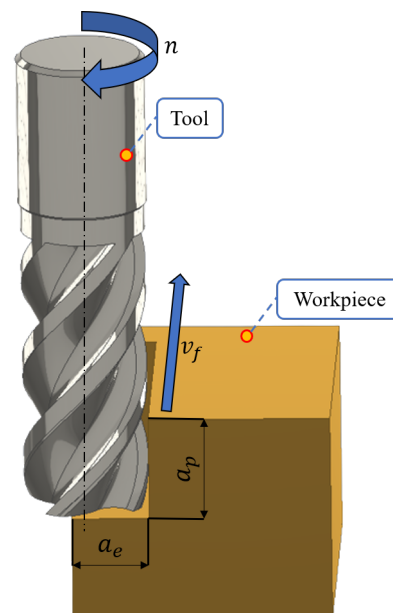


Figure 3: Representation of a basic milling operation. The main milling parameters have been included.

In a milling machine, the tool is mounted on a spindle that spins and moves along the three main axes, if the machine is a traditional one, or additional ones if the machine is more complex and advanced. The Z axis is always the spindle's rotation axis while the X and Y can be determined using the right-hand rule.

During a milling operation, the tool is spinning at a fixed angular speed, n , (measured in RPM in the industry) around the rotation axis, which generates the cutting action. Because of the revolutions, the motion is no more linear but uniform circular and the tangential speed of the tool must be calculated considering n .

$$V_c = \frac{\pi n D}{1000} \quad \left[\frac{m}{min} \right] \quad (1)$$

Where n represents the RPMs of the tool and D represents the diameter of the tool expressed in mm . Due to the different geometry of the cutting part, the feed must take into consideration the number of teeth of the tool; this facilitates the calculations and helps to understand how much the cutting edges are stressed while operating. In a milling operation the feed, f , is measured in mm per tooth and the V_f can be directly calculated knowing n , f and the number of teeth, z , present on the tool:

$$V_f = n f z \quad \left[\frac{m}{min} \right] \quad (2)$$

A higher feed rate leads to less machining time and higher productivity but can be detrimental for the quality. The engagement of the tool is set by choosing the depths of cut; these are two in milling: the radial depth of cut, a_e and the axial one, a_p , that can be identified from Figure 3.

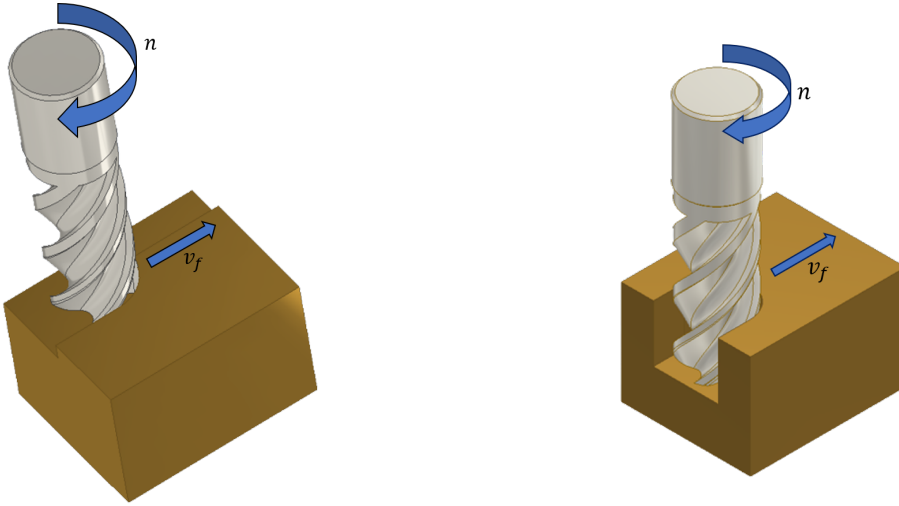


Figure 4: Facing and slotting operations. During slotting the minimum width of the channel that can be created is the diameter of the tool.

There are unthinkable operations that can be done with a milling machine by changing the relative position of the piece to the Z axis, combined with specific tools; in this work however, only facing and slotting were taken into consideration (Figure 4). These are among the most used and common ones in the machining industry.

The trajectory of the tool during a machining process is called *tool path*. During a facing operation, the tool path is often linear and big diameter tools are used so to reduce the number of passes, increasing the planarity of the machined surface and the smoothness. When machining a slot instead, various machining strategies can be used. One of the fastest and most adopted ones is called trochoidal slotting. A trochoidal tool path is a combination between a uniform circular motion with a uniform linear motion characterised by a continuous trajectory radius [3] (Figure 5). This motion generates less cutting forces and allows to increase the productivity by having a combination of high a_p and low a_e extending the tool life, increasing the heat dissipation [4]. Pay attention that this strategy can only be used when the width of the slot is bigger than the diameter of the tool.

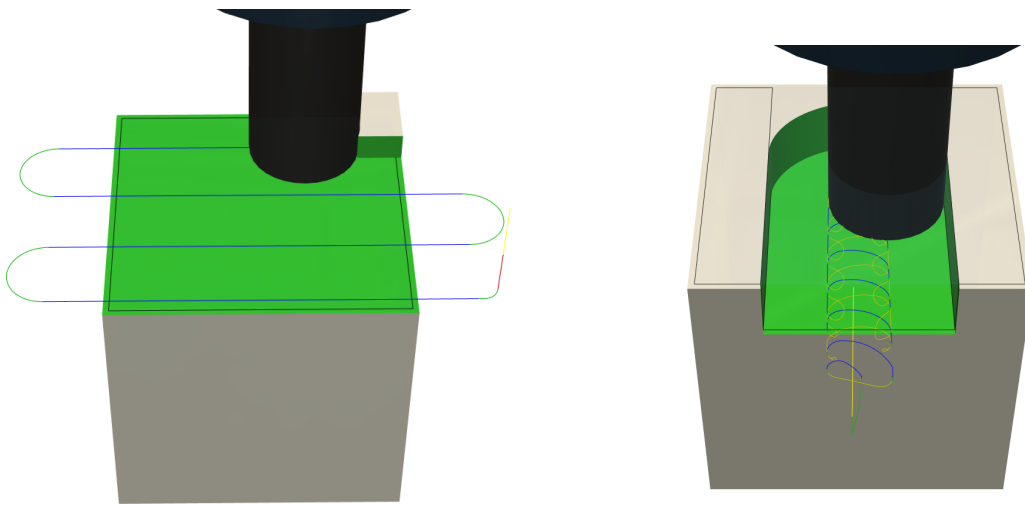


Figure 5: Conventional linear facing path and trochoidal one. The blue lines represent when the tool is cutting while the yellow ones when the tool is not engaged with the material.

0.1.3. An introduction to milling tools

We cannot neglect that inside the machining scenario tools are among the main characters. No matter the type of operation performed tools are always present and have to perform in the most extreme environments guaranteeing high performances and high resistance. There is a wide variety of milling tools that are specifically designed according to different operations and to cut several types of materials. However, the angles introduced before in the *Orthogonal* cutting environment can still be identified in each of them, with some additional ones derived from the basic ones, no matter their complexity. There are two big families of cutting tools: solid tools and indexable tools. The first ones are unique piece that is usually extruded or cast and then machined to obtain the cutting edges and the final shapes. These are usually made out of tungsten carbides, high-speed steels (general information for details please read the tool catalogues) or advanced ceramic materials, and can have advanced coatings to reduce their wear. The latter are usually made in two parts: the body of the tool, which often is made out of high alloyed tempered or hardened steels and more rarely of tungsten carbide, and the cutting inserts

which are secured on the body and provide the cutting edges. High Speed Steel (HSS) solid tools are usually less fragile than inserts because these, are made out of extremely hard materials that cannot be cast and must be sintered starting from powders. One of the biggest advantages of using inserts is that they are easy to change and each insert usually has more than one available cutting edge, so it can be switched once it becomes dull.

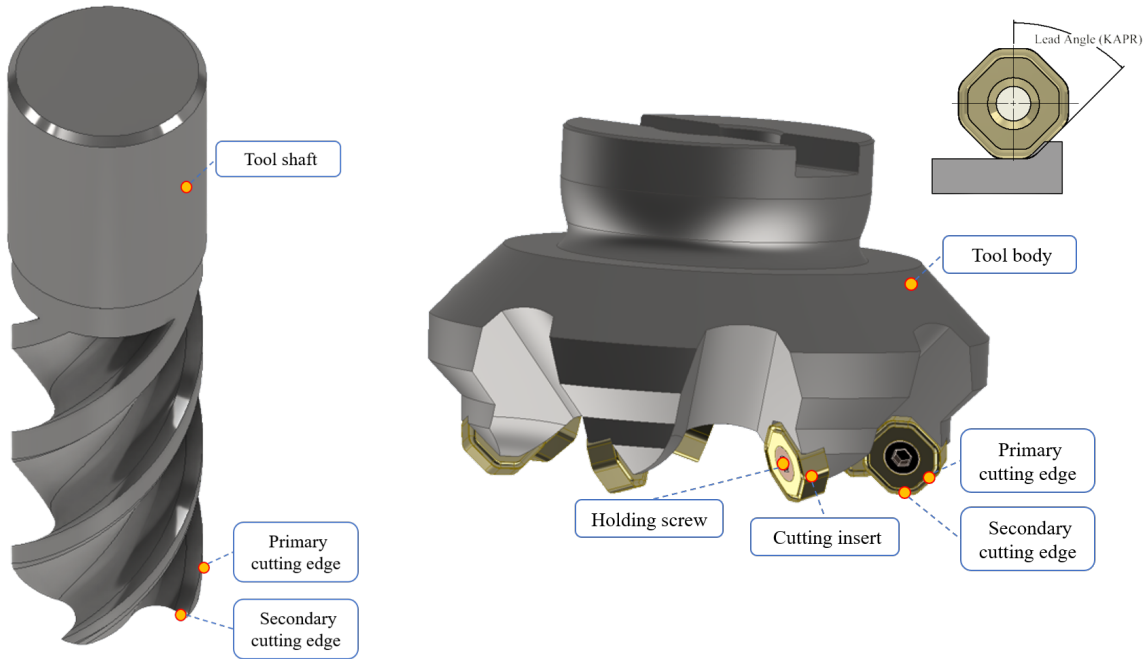


Figure 6: Solid tool and tool body with insert descriptions and comparison. Credits to [5].

As stated before the same angles introduced in the *Orthogonal* cutting environment can be used also in a milling one. Considering a solid tool for example we can identify the rake of the tool as the surface on which the chips flow. As a consequence, the main angles can be defined. Because in a milling tool there are usually two cutting edges then the *rake angles* are two: *axial* and *radial*. Moreover, usually, the cut is not orthogonal anymore and the main cutting edge is not perpendicular to the horizontal machined surface but has its own angle called *Lead* angle. *Clearance* angles can still be identified and are fundamental to avoid the cutting edges from scraping the machined surface and most importantly they avoid the cutting edges from stopping into the bumps that can be formed due to the accumulated local compressions that are generated under the surface while carving the chips. Many more angles can be identified and called with various names according to the different types of tools, tool companies and uses; however, only the most characteristic and general ones were introduced in this introduction.

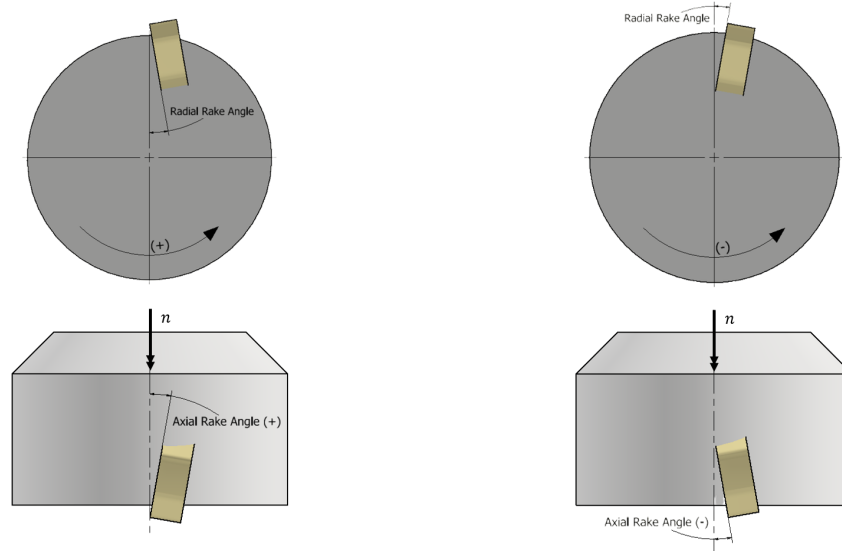


Figure 7: Scheme representing the bottom and side view of a single cutting insert and the tool body in which the radial and axial rake angles can be seen (pay attention to the fact that the rotation directions are different for each view).

During a milling operation, many agents and factors are contributing to attack the structure and integrity of the tool. Among those heat, abrasion, high local pressures and friction have a major role. Each time the cutting edge is engaged with the material it has to resist becoming dull, reducing its cutting ability and productivity of the overall process. This is why a correct tool wear study and prevention is very important in a machining environment. The most famous equation that rules the wearing process is the *Taylor's equation*:

$$V_c T^n = C \quad (3)$$

Or its extended version [6]:

$$V_c T^n f^a d^b = C \quad (4)$$

Where V_c is the cutting speed, T is the life of the tool, f is the feed per revolution and d is the depth of cut, in case of milling the axial can be used, and C is a machining constant that must be determined experimentally or using manuals and numerically represents the cutting speed that makes the tool last one minute when both the feed and depth of cut are numerically equal to one. All the exponents of the equation must be calculated experimentally solving a linear system that is obtained by applying the properties of the logarithm to the equation:

$$\log(V_c T^n f^a d^b) = \log(C) \rightarrow \log(V_c) + n \cdot \log(T) + a \cdot \log(f) + b \cdot \log(d) = \log(C) \quad (5)$$

But how can we measure the progressive deterioration of the tool's cutting edge? The ISO 8688-

ISO 6669:1989 standard regulates exactly how tool wear can be identified and quantified on milling tools. In detail, it considers the *flank* of the end mills as the area on which the wear must be considered. When uniform wear is present the limit on the width of the wear land, VB , is set to be a maximum of 0.3 mm on the mean value [7]. In the case of non-uniform wear is set to 0.5 mm on any of the flutes' maximum wear values.

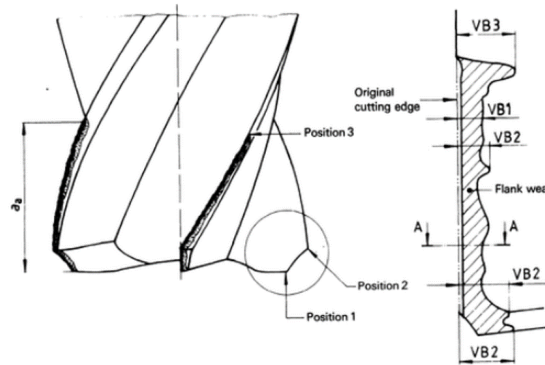


Figure 8: Tool wear representation according to the the ISO standards. [7].

In general, the tool's wear progression graph has a sigmoid shape that can be divided into three main phases. During the first one, all the sharpest edges are rounded off because those are the areas in which the local pressure is the maximum; in the second phase the main deteriorating agents and effects start to work on the tool *rake* and *flake* fighting against the coating, if present, or the hardened layer of the tool. Once this layer is broken the third phase is identified in which there is a rapid increase of the wear and the tool can break or become dull and as a consequence unserviceable any more for cutting operations.

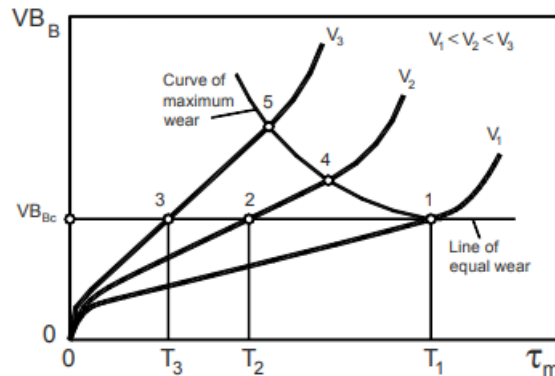


Figure 9: Typical tool wear evolution according to different cutting speeds. [6].

It is crucial to correctly identify these three regions, when analysing the wear progression of the tool, to consequently calibrate the correct cutting parameters depending on the machining operation and the final process output. For example, if during a batch lot production the tool wear assessment identifies the condition to be in the third region, the most crucial one, the decision to change the tool can be taken if many more pieces need to be produced and a failure of the tool during a machining operation can cause a bigger time loss and production loss caused

by the damage of the machined piece combined with the tool change process. If instead, the batch is nearly done, we can change the cutting parameters and be more precautionary and finish the production without the need of changing the tool. Please notice that these are two general qualitative examples that are only useful to show the reader how important the tool monitoring process is and how variable the decision-making aspect can be.

0.1.4. How can we measure the quality?

Quality inspection represents a crucial step in any manufacturing process. Although both the qualitative and quantitative approaches can be used to inspect the final machined products, the industry mostly relies on quantitative ones especially when parts need to be post-processed or assembled. Many quality indicators are used to evaluate a finished product but in the machining environment, without question, roughnesses are among the most utilised and important ones. The measured ones are always present in any quality report while their tolerated range is always imposed in the technical drawings upstream of the manufacturing processes. They guarantee the correct surface finish. Even if a considerable amount of literature and normative is present about the subject and much more can be documented, in this work only profile and surface roughnesses were taken into consideration but from a machining point of view and the full theoretical knowledge behind was not fully investigated as all the modern measuring machines are compliant to the corresponding ISO standards. However, in this section, an introduction to the basic theory and principles will be given to have a better understanding of the overall work.

'Surface roughness refers to microscopic geometric features of tiny peaks and valleys on a machined surface ...' [8], these features determine the final quality of the machined piece and affect also the mechanical, physical and chemical properties of the part itself. But how can we identify them efficiently and synthetically? To correctly diagnose them we need to find a numerical indicator that can be directly associated with a characteristic of those. In the literature, in particular inside the ISO 4287:1997 [9], there are plenty of indicators that can be used to correctly characterise these peaks and valleys. Because their distribution is random all of those values are extrapolated using statics and maths. Among those, the most used one is the *Arithmetical mean height of the assessed profile*, R_a :

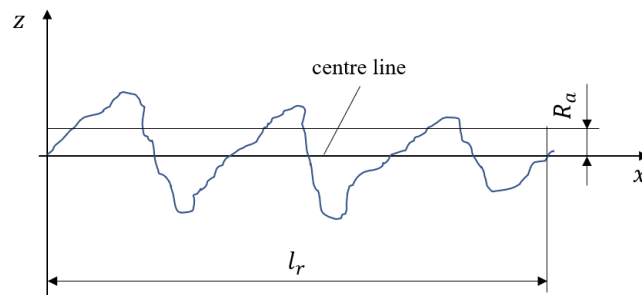


Figure 10: R_a definition according to ISO 4287:1997. l_r represents the sampling or inspection length considered for the evaluation and the centre line is such that the aggregate of the zones over the line is equivalent to the aggregate of the regions beneath the line. Credits to [9].

$$R_a = \frac{1}{l_r} \int_0^{l_r} |z(x)| dx \quad [\mu m] \quad (6)$$

As it can be seen, this type of measurement is a one-dimensional one because only considers the distribution over one line. The location and the parameters of the measurement must be chosen according to the standards. This indicator is the most known one and gives a rapid indication of the state of the surface and its quality of it. In the industry, experienced workers can have a rough estimation of it just by passing their nails over the machined piece or looking at the marks left by the cutting tool. There are also tables that can be used to understand the roughness requirements based on the field of application of the part (Figure 11).

This theory can be easily extended to a surface instead of a single line in order to have a more specific and accurate quantification. This evolution was possible due to the advent of optical machines that can scan a whole surface by taking a single picture. Although machinists still prefer to use profile parameters like R_a and the literature is as not well documented as for the others, surface parameters are getting more and more used, usually in combination with profile ones. This led to the definition of a similar parameter called *Arithmetical mean height of the assessed surface*, S_a :

$$S_a = \frac{1}{A} \int \int_A |z(x, y)| dx dy \quad [\mu m] \quad (7)$$

Even though the two look like they can be compared by some scaling factors or equation, performing so is theoretically wrong and will lead to a wrong analysis; however, we can use them together to extrapolate even more quality related information [8].

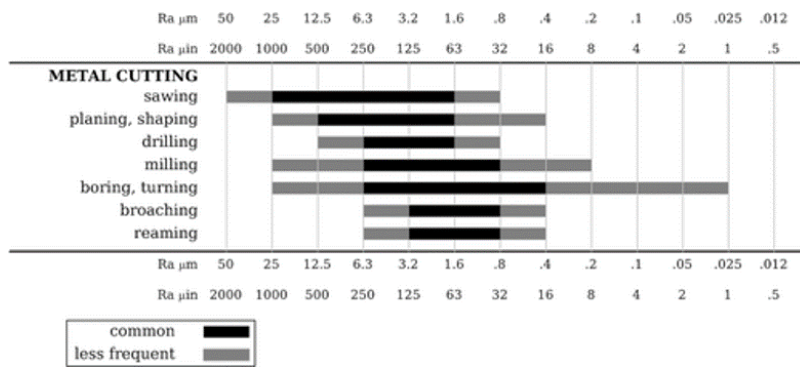


Figure 11: Example of a machinist table that can be found in any machinist or mechanical manuals. Credits to Wikipedia.

0.2. What is a CNC milling machine?

In traditional milling machines the human presence is necessary to operate them, moving the tool manually according to the specifications. This makes them slow and hugely reduce the

accuracy and repeatability of each manufacturing operations. To solve these and increase the productivity *James Parsons*, an Air force computer pioneer [10], decided to control a boring machine with data stamped on punched cards creating the first Numerical Controlled (NC) machine (1949). However the first ever Computer Numerical Control (CNC) machines started to make chips during the cold War in which computers gain more attention and became fundamental elements among the research equipments. Nowadays, with the advancement of Computer Aided Manufacturing (CAM) and Computer Aided Design (CAD) softwares is possible to create a 3D model of the desired finished piece and convert it into lines of code, the G-Code, that are fed directly into the machine and guide the tools to realize the final shapes. Closing the circle, a CNC machine is such that a computer directly controls the tools through a programming code. This thesis directly targets CNC milling machines which are intensively used in the modern manufacturing environments to produce a wide range of products with a wide range of materials.

0.3. Why do we need CNC milling machines?

Although today's Industry 4.0 is focusing on additive technologies such as 3D printing, subtractive processes are still the beating heart in most manufacturing companies and CNC milling machines are no less. The ability to work with a huge variety of materials with different hardnesses and mechanical properties, combined with a high level of automation, repeatability and precision are some of their key features. Moreover, as the research progresses and innovation grows, they are gradually becoming more accessible to small to medium businesses that are replacing the old traditional machines with more advanced ones. Tool companies are creating tools that when put to the test withstand extreme forces and loads to such levels that machines are becoming the bottleneck of the process because they are not powerful and stiff enough, suffering from vibrations and instabilities [11].

0.4. Issuing the problem

Even if these machines are beautiful, fascinating, extremely advanced and efficient they still possess downfalls as every other creation of humans including them-self. These are related to their complexity and the nature of the cutting process itself which is hard to be changed but still offers much space for improvements and developments. Furthermore, the human factor cannot be neglected, which still plays a big role inside the whole manufacturing chain.

0.4.1. The machining parameters tuning problem

Whenever a cutting operation starts the correct machining parameters must be chosen. These are crucial elements that directly affect the output of the process. Although in a milling machine there are many of those, all of them can be derived from the four cardinal ones (V_c , V_f , a_p , and a_e). These were detailed in the Introduction in which an insight into the milling environment

was given. How to choose those to obtain an effective and efficient process is still a challenge that companies and researchers have to face and constantly solve. The adaptability of CNC machines is indeed a great property but at the same time introduces complications when the machining environment must be configured; because each time, new aspects of the process must be considered before proceeding with the cutting; especially when choosing the best combination of the previously mentioned machining parameters. Most industries rely on the understanding generated by decades of chips produced and by experienced workers that can identify the perfect blending through a trial and error process. These acknowledged manufactures are able to listen and look at the machine while is cutting, process the outputs and change the parameters based on what they have already experienced. Although this method is effective and widely used, it lacks responsiveness and adaptability to new materials like composited and advanced alloys; in addition, possesses high variance, because is strictly related to the operator that is performing the monitoring so it is also difficult to hand down between new workers. New solutions have been developed using more sophisticated approaches that include the use of finite element simulations, theoretical calculations and the tool-manufacturer tables that are provided for specific cutting equipment. However, all of these are implemented before the machining operation starts and do not have accessibility and control over the process as the tool is cutting. They are approaches that are completed at priori and, the first two, are based on scientific models that have to be validated and adapted to every scenario.

0.4.2. The monitoring problem

As stated before the nature of the milling process is a subtractive one. This means that once the chips are produced the material is removed permanently. Even if additive methods, like welding, forging or casting, can be used to restore the removed volume it is rarely convenient to apply those, because the amount of time and resources needed will lead to higher costs. This implies that if the material is wrongly cut the quality of the final piece will be irreversibly affected and most of the time the machining operations must be restarted on a new chunk. Extreme care must be taken during the machining phases because the corresponding environment has a strong dynamic nature in which many phenomena can arise while the tool is engaged with the material. Besides that, modern CNCs have travelling accelerations that are in the order of magnitude of the gravitational one and in most industries, the monitoring of the procedures is still carried out by human operators. As known, individuals are very versatile but lack reproducibility and reaction time compared to robots, this leads to poor risk prevention because actions are taken once the damage or the problem occurs. It is not rare to see operators pressing the red emergency bottom to shut down a process after a scary noise has been heard. Moreover, most of the time the visibility inside the machine is limited due to the presence of additional components like fixing devices and additional tools combined with the chips that are produced which fly all around. A human eye stationed outside the machine has a low possibility of clearly seeing what is really happening inside, and as a consequence to take action when needed.

0.4.3. Previous related work

Improving the machining process has always been the focus of any research. Even if the targets could change, according to the different fields of interest, still, the priorities have consistently been the same. Because milling is an articulated dynamic operation, in which many components are working together, the improvement development cannot be focused only on one of those but must also consider the synergies and relationships. As with all industrial processes, the advancements are always guided by the four fundamental core values: cost, quality, rate and flexibility. As a matter of fact, all the late and past research in the field is trying to improve at least one of those four [12], [13]. The monitoring aspect is mainly targeting three of those: the reduction of production costs, the increase of the production rates and the enhancement of the production quality. Supervising a complicated machine requires deep knowledge of the environment and of the phenomena that can arise during machining. There is no unique best solution but instead, each one possesses strengths and weaknesses. Moreover, the adoption of one strategy over the other can also be induced by the specific tasks or final outputs that need to be obtained in addition to the specific conditions and environment in which we are operating. The monitoring topic has always been a crucial and delicate aspect of the machining environment. When traditional machines were dominating the industry humans relied on their senses to supervise. Eyes, ears, nose and hands are and were our most important sensors that could extrapolate information during the process, then processed by our brain and at the end decision were taken. With the advent and advancements of new technologies, powerful sensors have been developed together with sophisticated control units. Ears can now be replaced by very sensitive microphones that can analyse the sound spectrum far beyond the human range and with those pieces of information, the machining condition can be identified. Y.D. Chethan et al. [14] optimized the machining parameters of a lathe machining process by identifying the tool wear with a combination of machine vision and acoustic emission that can be directly correlated to the state of the tool. The combination of the two allows to obtain a better estimation and to extend the adaptability of the method to different machining environments. As we know sounds and vibrations are correlated and important information such as the surface quality and the tool state can be extrapolated from them. T. Y. Wu et al. [15] obtained an estimation of the surface roughness using an Artificial Neural Network that receives as input the post-processed vibration signals obtained from the accelerometers placed on the vice and on the spindle of the milling machine and is able to predict the R_a of the machined surface with an overall *Mean Absolute Percentage Error (MAPE)* of 25 %. Vibrational approaches are very powerful and useful but require intensive data post-processing and feature identification. Moreover, the signal can be spoiled from many environmental elements and also can vary according to the different materials and tools used. In addition, the other vibrating elements of the machine can interact with the signals and create additional noise or resonance peaks that can ruin the measured data. During quality inspection, eyes always play a very important role, especially when trained ones are used. These can easily recognise imperfections, spots or traces left by worn-off tools or damaged ones. The industry technological equivalent of those are cameras that, combined

with machine vision systems, can extrapolate information from acquired images and videos. In the manufacturing environment, this is nothing new. Machine Vision for monitoring purposes has been already implemented for nearly half of a century and has enabled to reach production speeds far beyond the human eye tracking possibilities. Regarding milling, research has been done both in the monitoring and estimation domains. Before the advent of CNNs S. Palani et al. [16] used a camera connected to a computer to take pictures of the machined surfaces; these were fed into a computer to extrapolate the relevant features needed to perform the evaluation. In the work, the authors chose to analyse the images using the *Fourier Transform (FT)*, due to the repetitive nature of the texture created by the tools. The R_a was estimated using an ANN that took as inputs the machining parameters, the average Grayscale and the *Major peak frequency (F1)* and the *Principal component magnitude squared (F2)*, which are calculated from the *FT*. They managed to obtain an average error of 2.47 % which proved the feasibility of the system in an industrial environment. However, this method requires feature extraction which can be challenging if the machining pattern changes and the ANN must be calibrated every time the material or the tool changes. As stated before CNNs are part of the Machine Learning family and extrapolate information from the images by convoluting them with different filters that each time try to focus on particular features. One of the advantages of these is that those filtering layers can be calibrated based on a set of pictures used as reference (the so-called training set) and do not need any features estimation to be performed; in Chapter 2 a more complete explanation will be given about the whole families of ANN and CNN. Achmad P. Rifai et al. [17] applied these to evaluate the R_a of both milled and turned surfaces. They created their own architecture and trained it using 5 different loss functions. Images were preprocessed by applying different filters and also enhanced by splitting and rotating them. The prediction time of each model was also measured and they were able to obtain a prediction accuracy between 88 % and 91 % depending on the different machining operations. Yonglun Chen et al. [18], [19] compared 3 different well know CNN architecture: the ResNet50 [20], Xception [21] and DenseNet121 [22] to classify different families of R_a . They took the roughness range and divided it into 6 and 12 categories. The problem was shifted from a regression one to a classification one. In addition, they try to assess the performance of the models in different light environments. They were able to generate a dataset of 2840 images after the data enhancement step which was split between training, validation and testing. A total of 100 epochs of training were executed and a final accuracy of maximum 98.24 % was achieved in a controlled lighted environment. As it can be understood another possible approach could be to estimate the roughness directly by knowing the machining parameters and tool wear. This solution can be really effective when coolant is used and so images cannot be taken properly; moreover, the machining parameters are always known a priori without using any sensor because are inputs that are given to the machine and are directly present in the G-Code. When machining composites for example it can be hard to take the correct pictures because the light absorption and reflection can be problematic, also the machined surface is usually composed of many particles and elements laying at different layers that can trick the CNN due to their repetitive nature. As matter of fact, Cam Boga et al. [23]

developed an ANN that, taken as inputs the type of cutting tool, the spindle angular speed and the feed rate, is able to predict the final R_a of the machined surfaces of high-strength carbon fiber composite plates. The development of the perfect architecture (for example the number of hidden neurons) of the network was conducted using a genetic algorithm and the final trained ANN was able to make predictions with an overall *coefficient of correlation* (R^2) of 0.96 and a *Mean Squared Error* (*MSE*) of 0.076. Once the quality is estimated actions must be taken as a consequence of the estimation. As stated before, in the manufacturing field action are taken mainly on the machining parameters that must be changed or set based on specific conditions. The optimisation of those before starting a full machining cycle can be a really complicated challenge that most of the time can only be overcome using trial and error approaches. However, with the advent of machine learning and with the advancement of computational power more and more optimisation methods and algorithms have been investigated and proven to be suitable for such tasks. Even here, we have to consider that there is not a unique solution that stands up among all the others but different alternatives are present with their negative and positive properties. Besides that, other factors like the computational resources and the optimization nature can deeply influence the choice. Reinforced Learning (RL) definitely represents one of the most important machine learning paradigms of nowadays and has been widely implemented in many engineering fields such as robotics and manufacturing being capable of dealing with multi-objective optimisation tasks. Zhenhui Wang et al. [24] used this to optimize the machining parameters of a milling operation in order to obtain the minimum R_a with the maximum *MRR*. The roughness of each combination of variables was estimated using DDQN-improved support-vector regression which allowed them to use a small batch of raw data to train the regression model and the prediction results were better than those obtained using a genetic algorithm. After performing the optimisation the system was able to identify the Pareto curve and suggest the best combination of machining parameters based on the initial input space and constraints. In the machining environment, the benefits of a well-developed optimisation system can be appreciated both before starting the machining operations (at priori) and especially while performing them (online). The latter can be a more sophisticated topic due to its dynamics and tool material complex iterations. The benefits of combining an online monitoring system with a real-time optimisation module were already investigated one decade ago. Ghassan Al-Kindi and Hussien Zughaer [25] mounted two cameras on a CNC milling machine to constantly monitor the process and extrapolate the surface roughness in real time of the last machined areas. The system was intelligent enough to switch between the two to get the perfect picture and also was able to analyse the G-Code to understand the coordinates of the machined area. In addition, the system was able to directly modify the G-Code and put the machine in a waiting state until the new code was uploaded. The R_a was estimated by pre-processing the images and feature extraction based on the grey-scale intensity. Results proved that the machining operation and the machine can benefit from this monitoring system; however, they did not validate their results with additional experiments and they did not investigate the applicability of the system to industrial machines. Ravi Sekhar et al. [26] created a control loop control system that was able to control the final R_a

of *Al-Mg* matrix composites reinforced with carbon nanotubes and other carbides that made the camera estimation very difficult to be implemented and so an ANN was implemented to predict the final roughness and adjust the machining parameters as a consequence to meet the initially set requirements using a PID controller. Results proved that the method was actually able to control the quality oscillations and keep them in a limited range. As it can be noticed, once the AI paradigm was been able to be developed all the research moved towards that direction implementing more and more advanced algorithms to solve sophisticated problems. Tien-Dung Hoang et al. [27] were able to optimise high-speed milling processes by creating an ANN that was able to predict the final tool wear based on the initial one, the machining parameters, the cutting force and time; this information was then fed, together with the *depth of cut* and the textitcutting speed, in another ANN that was able to optimise the V_f to keep the R_a in a limited range, which was modelled using an empirical exponential function whose coefficients were estimated using the machined samples. The ANN was able to predict the tool wear with a *Root-Mean-Squared-Error (MSE)* of 0.023 mm and was able to keep the roughness in the allowed range even with global tool wear of 0.06 mm.

After going through most of the research conducted so far we can understand that the monitoring and optimisation fields are still of interest. However, most of the presented works are either using feature extraction algorithms, which possess low adaptability and they need to be calibrated every time through a trial and error process and by an expert that fully knows the theory that stands behind the mathematics and the procedure, or rely on ANN that are trained using many samples and try to predict the output after many epochs of training and validation. Related to optimisation instead [28], most of the techniques used in research always need an estimator that is able to predict the outcome of the process so that its parameters can be estimated. In addition, this phase is mostly conducted before starting the machining operations and only a few were able to successfully change the parameters as the machine was working. In the presented work the monitoring phase was realised using a camera that is able to predict the output using CNNs without the need for feature extraction and elaborated pre-processing of the acquired image, which is helpful to perform the operation in real-time; in addition, the optimisation module was developed using the so-called 'black box optimisation' family of methods with do not care about identifying the model that connects the outputs and the inputs but tries to generate this link using stactical methods. This ensures that only a limited amount of samples are needed to create an optimiser that can fully respond and take action when needed.

0.5. Thesis overview

0.5.1. Thesis's target

As it can be imagined reading the previous sections of this thesis the aim of this work is to create a quasi-autonomous machine that is capable of self monitoring and control itself with a low human presence. This can be achieved combining different algorithms of machine learning

that have been trained on a small number of samples created using the same milling machine. The controlling of the system is realised using *Python* because the CNC software is a closed one that was realised by the manufacturing company and this needed specific attention because of its architecture. Different machines have different controlling units so each time the controller must be calibrated specifically. The overall system, once a machining operation is concluded, takes a picture of the machined surface and the tool, identifies the surface roughness and optimises the *feed rate* and *rpm* based on those inputs to guarantee the same surface quality with the highest *MRR*. The created system is low cost and requires parts that can be purchased online and are widely available on nowadays market, showing the possibility of creating a solution that can be adopted also in the medium to small manufacturing companies.

0.5.2. Thesis Structure

After the introduction, the attention will be brought to Chapter 1, which gives the reader more information about the methodologies and machines used to complete this thesis and details the terminologies and parts present in each machine. In addition, a final description of a CNC machine is given and the basic concepts behind Machine Learning and Artificial Neural Networks are presented including a deeper characterization of the algorithms used in this work, some of the Math behind the performance evaluation metrics, the network architectures and the connections between different code parts. This part, combined with the Introduction Chapter and Chapter 2 which analyses the approach used and the choice made, gives the reader the possibility to close the knowledge circle about machining, understand all the parts of the final system and develop his or her own critical opinion about the results and performances. These are illustrated in Chapter 3 which discusses and comments on them, giving also an answer to most of the why questions that can arise while reviewing the previous chapter, especially regarding the choices that were made. All the work is summarized in Chapter 4 in which the crucial achievements are outlined and a door on future developments and also feasible dreams is widely opened.

1 | Computer Numerical Control machines & Artificial Neural Networks

Even if in the Introduction a brief description was given about both the CNC machines and ANN (Artificial Neural Networks), still more information can be given about these two important topics that compose the groundings of this work, giving the reader a better view and understanding of the overall project. However, not all the specific details will be given and only the basics will be discussed. If a deeper understanding is desired, readers may consider reading and studying the related cited documents.

1.1. Computer Numerical Control (CNC) milling machines

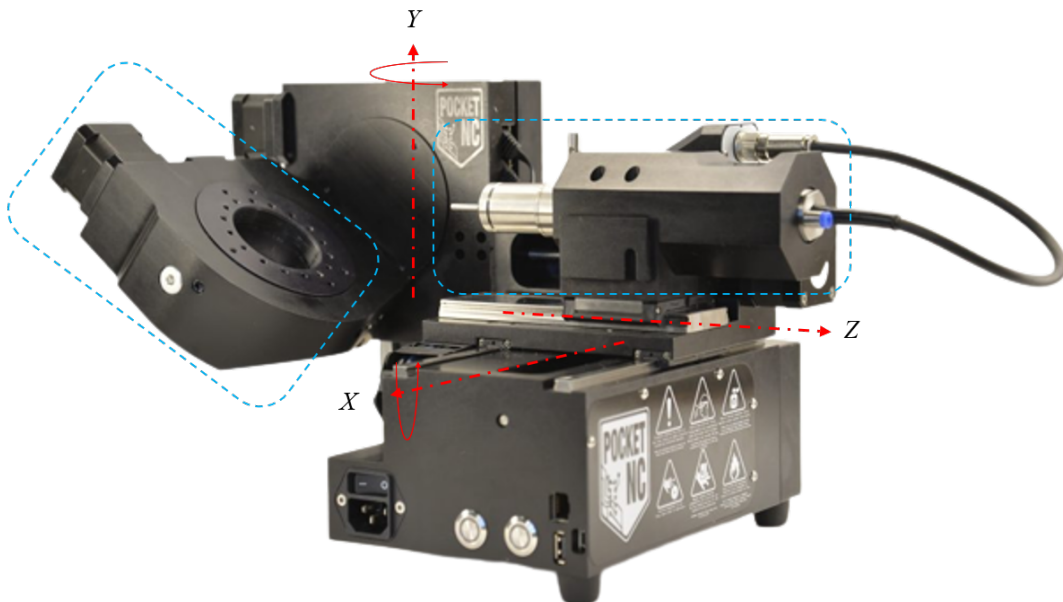


Figure 1.1: The CNC machine used in this work (Pocket NC V2-50 CHB, see the Appendix B: CNC Specifications, to look for the technical specifications).

Figure 1.1 shows a CNC used in this work which is a 5-axis CNC milling machine, and it will be used in this chapter to provide a general characterization of these pieces of machinery. As it can be noticed from Figure 1.1, it has 3 linear axes: the Z one, which in CNC machines is always the tool axis, the X one and the Y one. The orientation and position of these are decided by the manufacturer of the machine but they are always orthogonal to each other in order to form a reference system. In addition, this machine has 2 rotational axes that are called A and B that are centred around the two linear axes X and Y respectively. In this equipment, the X axis is fixed with the tool holder but in bigger machines, the tool holder usually only possesses the Z one while the other two are fixed with the machined piece. In every CNC each axis is controlled by a different motor and is independent of the others, although in some machines motors can be linked together using a master to slave hierarchy. In most milling machines we can divide them into two main areas: the tool one and the piece one. The first one, represented by the dotted box on the right part of Figure 1.1, regards all that is needed to make the tool cut. In this case, the spindle, the cutting tool and the motor that provides the torque. The second area regards what is related to the machined piece (dotted box on the left part of Figure 1.1). In this case, is a platform that has holes in it to address the multiple clamping systems that can be installed to hold the material in place while the tool is cutting. This is a crucial element of the machine because has to be able to clamp down different materials and shapes and hold them steadily as the tool is removing the chips from them, resisting the vibrations and forces that are generated. In addition, it must also facilitate the switching between one piece and the others in order to reduce the overall processing time.

1.1.1. The G-Code

How do we control these machines? As stated before once the 3D model of the final piece is completed and the tool path is generated, these are converted into the machine language creating a G-Code. This stands between the human and the equipment and controls its functions and motors. The code is automatically generated by the CAM software through the use of a post processor, that is usually present directly inside the software, itself once all the tool paths and processing are defined.

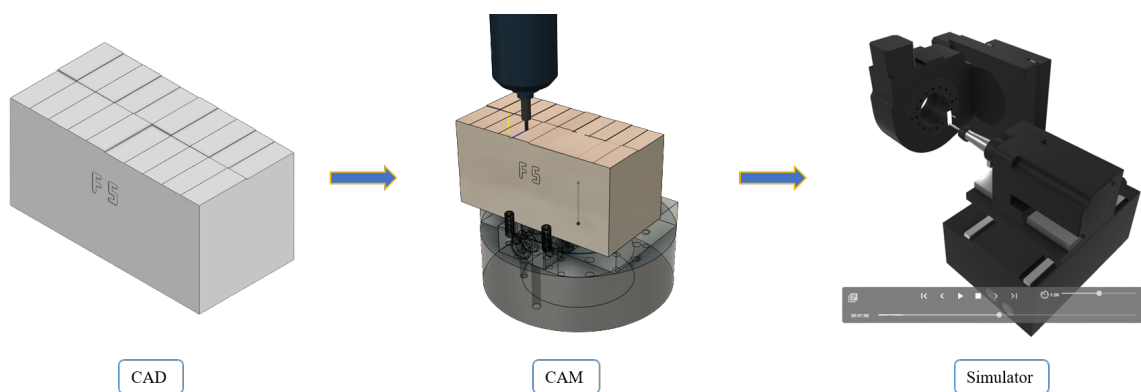


Figure 1.2: Development phases of a machined piece before actually entering the CNC machine.

As it can be seen from Figure 1.2, once the CAD model has been finished it is handled by a CAM software that, as stated before, generates the final G-Code. Before proceeding with the actual machining a simulator is usually used to check the running of the code and identify any collisions between the different moving elements or any not allowed movements that are requested by the G-Code [29]. These simulators are also implemented inside the CAM software but sometime they are also provided by the CNC company or third parts companies, possessing the best fidelity in terms of environment and machine components. Modern simulators can also check and calculate the instantaneous cutting forces and power that is generated during the cutting process and even predict the tool wear or the heat generated during the process. However, these are used only in specific applications in which the machining tolerances and accuracies must satisfy strict requirements and as a consequence, every phenomenon must be considered and counter-measures must be rapidly taken. This is usually done by making the simulation environment and the real one interact with each other so that the software is constantly updated with real-time signals and predictions can be performed with better accuracies; this family of models is called *Digital Twin* [30] and usually requires big efforts from the economical and data gathering points of view.

```

%
(Axis, stop)
(BAE_UP1)
N10 G21
N15 G90 G94 G40 G17 G91.1
N20 G53 G0 Z0.
(2D ADAPTIVE13)
N25 G49
N30 M5
N35 G53 G0 X63.5 Y63.5
N45 T1
N50 S11500 M3
N55 G54 G0
N60 G53 G0 X63.5 Y63.5
N65 A90. B0.
N70 G1 X-44.412 Y-8.464 F10000.
N75 G43 Z47.617 H4
N80 G1 Z37.617
N85 Z32.817 F10000.
N90 Z32.417 F500.
N95 X-44.405 Y-8.468 Z32.338
N100 X-44.386 Y-8.479 Z32.263
N105 X-44.354 Y-8.498 Z32.194
N110 X-44.311 Y-8.523 Z32.134
N115 X-44.259 Y-8.554 Z32.084
N120 X-44.2 Y-8.59 Z32.047
.
.
.
N11225 X-32.635 Y0.21 Z32.047
N11230 X-32.698 Y0.181 Z32.084
N11235 X-32.753 Y0.156 Z32.134
N11240 X-32.798 Y0.136 Z32.194
N11245 X-32.832 Y0.12 Z32.263
N11250 X-32.853 Y0.111 Z32.338
N11255 X-32.86 Y0.108 Z32.417
N11260 Z47.617 F10000.
N11265 G53 G0 Z0.
N11270 G49
N11275 G53 G0 X63.5 Y63.5
N11280 A0. B0.
N11285 M30
(Axis, stop)
%
```

Figure 1.3: Example of a G-Code produced by the post processor of the CNC machine used in this work.

In Figure 1.3 it can be seen that the G-Code taken into consideration can be divided into three main parts. The first one is the opening which usually defines the global variables, reference systems that are used and the units of measurement which are crucial and care must be put into choosing the correct ones, to avoid damages that can occur inside the machines as the tools move. The second part defines the milling operations. In this case, only one was conducted ("2D ADAPTIVE13"). Even here can be identified an opening section, in which the system coordinates are defined, the tool is assigned and some of the machining parameters and coordinates are set; this is followed by the machining section in which, for each line, the coordinates at which the tool must be positioned are given in order to remove the chips. Finally, the closing part positions all the axes in a reference position and stops the tool and all the other movements and speeds. Care must be put each time because different machines can have some modifications of the code functions based on the manufacturer choices and language used.

1.2. Artificial Neural Network (ANN)

Machine learning (ML) is a subfield of artificial intelligence that gives computers the ability to learn without explicitly being programmed (MIT definition). Artificial Neural Network (ANN) is a technique of the ML family that is inspired by the learning mechanisms of biological organisms [34]. This section was specifically created to provide to the reader the basic terminologies and theory related to the field. If further studies and understating are needed we suggest the read of the related scientific work used as benchmark to create this section [31], [32], [33].

1.2.1. Introduction

Imagine opening the head of a student while he is listening to a Math lesson and seeing what is happening inside his brain. What we will notice is that there are a big number of neurons that are connected to each other through the *dendrites* and *axon*, exchanging electrical and chemical signals of different intensity among themselves (*synapses*). If, pushed by curiosity, we take and compare a normal student's brain to Einstein's one we will see that, even if the scientist one is heavier, what makes the real difference is the number of connections. Einstein's brain definitely has a more complicated and entangled network of connections between neurons that allows him to think, elaborate data and generate stimuli involving more neurons. Indeed, the training of a biological neural network consists in creating new connections and strengthening or weakening the existing ones. As stated in the opening part of the thesis observation is the key element to innovation. In fact, since these biology mechanisms became clearer (formulation of Hebbian Learning by D.O. Hebb in 1940), it did not take much for humans to develop the first ANN, *the perceptron* by Frank Rosenblatt in 1957 [35], inspired by the learning process of the human brain.

1.2.2. Basics Math and architectures

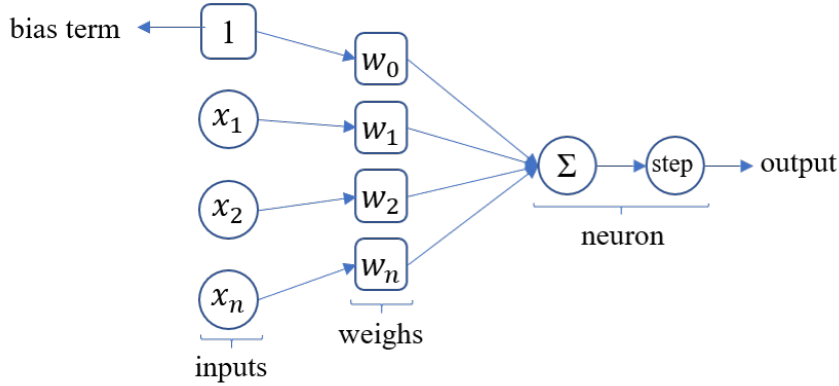


Figure 1.4: General structure of the perceptron. Inspired by the connections between biological neurons [31], [35].

The perceptron is the simplest ANN because it only has one neuron. As it can be seen from Figure 1.5 the network takes as input n inputs plus the *bias* and gives a single output. In mathematical terms let's consider a number of m examples each one containing n elements. This can be represented by a matrix \mathbf{X} which is a $m \times n$ matrix. If we consider a single experiment, this can be represented by the vector $\mathbf{x}^{(i)}$ which represents the i -th row of the matrix. Now in order to train the network we have to create a training set which is composed by the matrix $\bar{\mathbf{X}}$ that is modified to consider the *bias* and the target values of each experiment that can be grouped in the vector $\mathbf{y} = [y_1, y_2, \dots, y_m]^T$. Each time one of the i -th experiment, $\bar{\mathbf{x}}^{(i)}$ is fed into the neuron, a weighted sum with the relative weights $\mathbf{w} = [w_0, w_1, \dots, w_n]$ is performed and then the scalar result is processed by an *activation function* Φ and so the output, $\hat{y}^{(i)}$, is created:

$$\hat{\mathbf{y}} = \Phi(\bar{\mathbf{X}} \times \mathbf{w}) \quad \rightarrow \quad \hat{y}^{(i)} = \Phi \left(\sum_{j=0}^{j=n} w_j \bar{x}_j^{(i)} \right), \quad \forall i \in [1, \dots, m] \quad (1.1)$$

There are many activation functions that can be used, and each of them has its own characteristics and peculiarities that must be considered depending on the nature of the problem. For example, in this network the *step* function was used, but others like the *identity*, $\Phi(x) = x$, or the *Relu*, $\Phi(x) = \max(0, x)$ and others can be used. The output of the network must be compared, $\hat{y}^{(i)}$, to the real corresponding value, $y^{(i)}$, in order to "teach" the ANN how good its prediction was and its weights must be updated based on the error of prediction, in other words the ANN must be trained. There are many *loss functions* that can measure this error among them two of the most intuitive and most popular ones are the *mean squared error (MSE)*:

$$L(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \quad (1.2)$$

and the *mean absolute error (MAE)*:

$$L(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m |\hat{y}^{(i)} - y^{(i)}| \quad (1.3)$$

To train the network the weights must be updated following a certain criteria that is dictated by the first derivative of those loss functions. Because it is a minimization problem we can update the weights following the direction provided by the negative of the derivative but scaled by a factor, *learning rate* α , that determines how much to move each time.

$$w_j = w_j - \alpha \frac{\partial L(\mathbf{w})}{\partial w_j} \quad (1.4)$$

The choosing of α is a crucial task because determines the efficiency of the training. A too-high value will speed the training process but there is a higher chance of getting stuck into local minimums and the solution will be more unstable because during training jumps between different minimums. On the other side, a too-low learning rate will require a lot of training time and there is the risk that training will be stopped before the optimum solution is reached.

Although the perceptron has only one neuron and it is a linear model, so has all its limitations, it paved the way to modern neural networks in which the linear limitations are overcome gaining together more neurons to form a fully connected network. However, the same principles described before are applied even to more complicated ANN with little differences.

The structure of a more advanced ANN extends the perceptron concept. There are still n inputs nodes but then more than one neuron is present. These are the so-called *hidden layers* that are agglomerate of neurons with different weights and, sometimes, different activation functions. Lastly, there is the output layer which processes the data and obtains the output or the outputs.

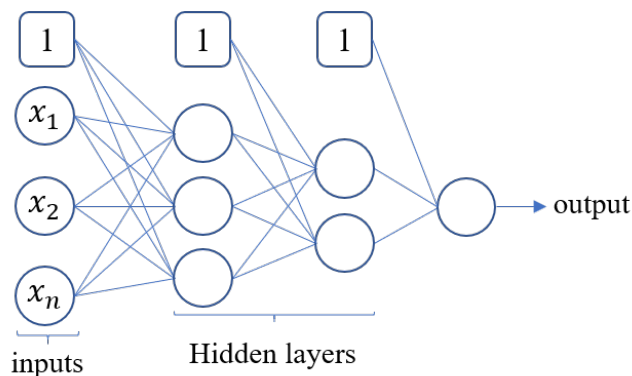


Figure 1.5: Example of ANN with n inputs nodes, two hidden layers (one with three neurons the other with two) and a single output node.

The same notation is used to explain the maths behind this more complex networks but this

time the weights are grouped into weight matrices [36]. To keep track of each layers the variable l is used to indicate the hidden layer number that we are referring starting from the first one, while p_l is the variable used to indicate the size of the l -th layer. Let's consider a neural network with n inputs nodes, L hidden layers and p_L outputs. \mathbf{W}_{l+1} represents the matrix of weights that links the layer l with the $l+1$ one and \mathbf{h}_{l+1} the vector of the outputs generated by the neurons of the $(l+1)$ -th layer. As a consequence the \mathbf{W}_1 matrix has a size $(n+1) \times p_1$, while \mathbf{W}_{L+1} matrix $(p_L+1) \times p_{L+1}$:

$$\begin{aligned} \mathbf{h}_1 &= \Phi(\bar{\mathbf{X}} \times \mathbf{W}_1) \\ \mathbf{h}_{l+1} &= \Phi(\mathbf{H}_l \times \mathbf{W}_{l+1}) \quad \forall l \in [1, \dots, L-1] \\ \hat{\mathbf{y}} &= \Phi(\mathbf{H}_L \times \mathbf{W}_{L+1}) \end{aligned} \tag{1.5}$$

Where \mathbf{H}_{l+1} represents the matrix of the outputs generates the neurons of the $(l+1)$ -th layer and is a $p_l \times p_{l+1}$ one. The computations explained by Eq. 1.5 are referred to the *forward* in which the data is passed through the network to be evaluated and the estimation are generated. In the *backward* pass instead the errors and gradients are propagated in the opposite direction to calibrate the weights layer by layer. The combination of the two operations is the core value of any neural network and the algorithm that performs such tasks is called *backpropagation* algorithm. This can be computed after all the dataset has been passed forward or by dividing the inputs into smaller *batches* and propagate the errors each time a batch is analysed; once all the batches have been processed, and so all the input data, an *epoch* is completed. Usually it takes many to train a whole ANN.

A more detailed analysis of the gradient calculations and the algorithm body can be found in the cited literature at the opening of the sub-chapter.

1.2.3. Convolutional Neural Networks (CNN)

When we think about ANN we think about the inputs in form of a column vector. But what if those are multidimensional matrices? Imagine that we want to process black-and-white images inside our networks. In that case, we maybe need many input nodes, one for each pixel so as the resolution increases our network would become massive and this would slow the training process and the estimation part. Moreover, do we really need to evaluate each time a single pixel on its own or it is better to have a more global view of the image by a group of pixels? The answers to these questions pushed the research to create a different type of network that could process RGB images with a low number of parameters needed but still impressive results. These neural networks use the so-called *convolutions*. In pure mathematics, this is an operator that takes two functions and produces a third one that shows how the second one modified the first one, CNN is a mapping operation in which a pixel is mapped in an output pixel using a mathematical operator (*filter*) that consider also the other pixels surrounding it or the only pixel itself.

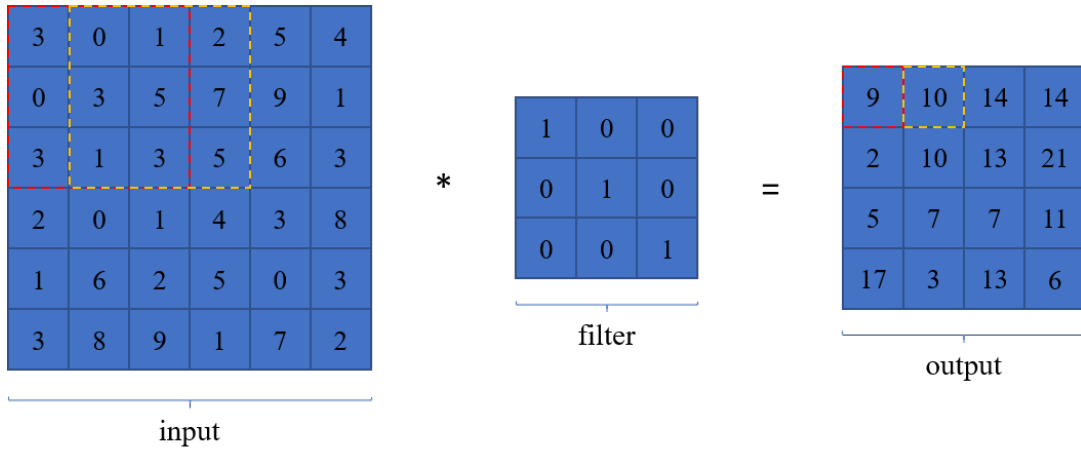


Figure 1.6: A 2D convolution operation explained. In this case the filter extracts the trace of each sub-matrix to make it easier to understand.

In Figure 1.6 a simple convolution is explained. In reality, images are 3-dimensional matrices because they have 3 matrices, one for each fundamental colour. As a consequence, the convolution is extended to the third dimension by summing the results of each 2D one. Each cell of the matrix represents the intensity associated with the pixel of that colour channel. This requires that also filters are 3 dimensional so there is one filter for each channel. The filter size is usually expressed as $f \times f \times n_f$ and the number of channels is omitted because it is equal to the input. n_f represents the number of filters so if for example the number of channels is three and n is ten that means that there ten filters that are $f \times f \times 3$ so the output of the convolution will have ten channels and no more three.

Not only one single type of mapping operator exists but others like the *maxpooling* or the *averagepooling* are used too. These, instead of performing a sum between elements, act differently: the first extract the maximum value of each sub-matrices while the second performs the mean for each one. In addition not always the filter moves inside the matrix by one cell at time and it can also jump by two or three. These amounts are represented by the so-called *stride* (s) that is an integer bigger than zero. In addition, *padding* (p) of zeros can be added to the input matrix borders to increase its size. In the end, if the input is a $n \times n$ matrix the output size will be :

$$\left[\frac{n + 2p - f}{s} + 1 \right] \times \left[\frac{n + 2p - f}{s} + 1 \right] \quad (1.6)$$

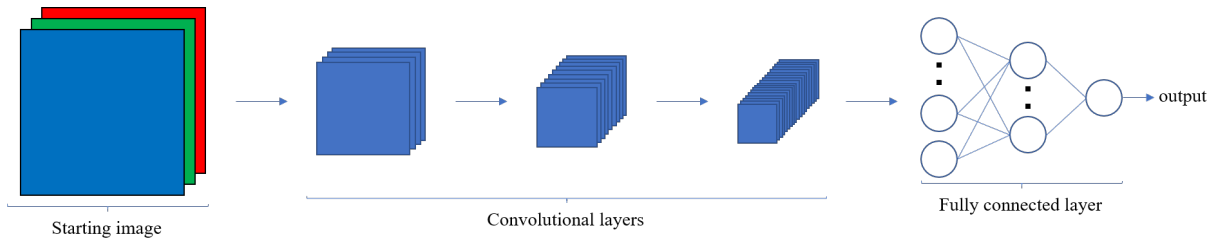


Figure 1.7: Scheme of a complete convolutional neural network.

As it can be seen from Figure 1.7 a fully constructed CNN is composed of many different parts. Once the information is extrapolated from the image through the different filters, the final prediction is computed using a tailor-made ANN. Images are usually preprocessed before being processed by the network to reduce the noise and guarantee maximum performance. The training of these networks consists in uploading the filters and the weights of the fully connected layer. This is done using the same algorithm introduced before but with images as inputs combined with their respective true outputs.

CNNs have improved by far the machine vision field by introducing advanced architectures that can operate in many different fields and carry out different tasks. They have the advantage of feature sharing because they can share the feature detection among different parts of the image and each output of the layer is influenced only by a fraction of the inputs. In addition, they can be compact and faster than traditional ANN because they require fewer parameters and knowledge can be transferred from one network, which has been trained on the same task but on a different dataset and outputs, to another so they can be reused and adapted.

2 | Proposed Solution and its Application

The overall concept of this thesis can be divided into three main parts. The first one regards the estimation of the roughness from the taken pictures during the machining operations. The second is founded on the optimisation strategy used to get the best machining parameters for the specific machining operation. The last one closes the circle by combining the previous two in order to create an autonomous system that is capable of monitoring and correcting a manufacturing process. This section explains in detail how each task was mastered, using specific equipment and procedures. The CNC mill used to perform all the machining operations was the one represented in Figure 1.1 with three independent linear axes and two independent rotational axes. This machine has a limited power of 600 *W* and mounts only tools that have a 4 *mm* shaft diameter. In addition, it has a limited clamping surface of maximum nearly 100 *mm* by 80 *mm*. This device is controlled by a company developed software (Penta Machine, Pocket NC [37]) that is accessible through the Web and has an integrated tool calibration system.

2.1. Estimation method

To get the CNN to perform a proper estimation, training had to be performed. In this case, however, no transfer learning could be used because the task was completely different from an image classification one. In addition, no database was available so, it had to be built from scratch.

2.1.1. Dataset creation

To build the dataset to train and test the CNN many samples had to be machined and also a good distribution of pieces per roughness range had to be achieved. Because the machine power is low, aluminium was chosen to make most of the samples because of its mechanical properties of being lighter and easier to cut compared to steel. In addition, no coolant was used so to avoid spoiling the pictures and the machined surfaces. Steel samples were also created but in a limited number. Billets of aluminium 6061 of size 40*mm* by 40*mm* by 80*mm* were used. The manufacturer specifications indicated that all the six faces of the billets were face milled guaranteeing a dimensional error from 0 to -0.2 *mm*; however, once arrived, one's dimensions were measured and stored in an Excel file, to account for the machining error of the producer.

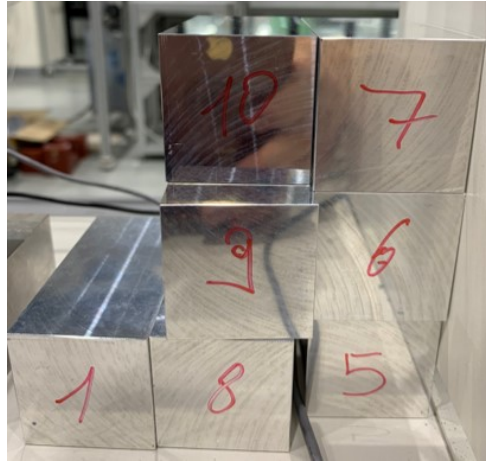


Figure 2.1: Aluminium billets after being measured and catalogued.

To obtain the maximum number of samples per billet all the top and bottom surfaces were machined. As it can be seen from Figure 2.2 the machined surface is full of little rectangles that have different depths.

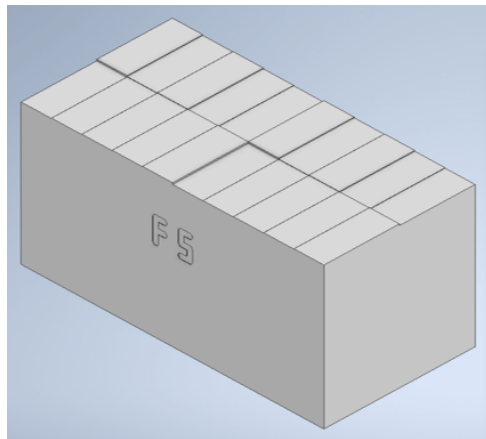


Figure 2.2: 3D model of a machined billet. On the top surface each rectangle at a different height represents a different sample. The deepest ones will have rounded edges in the real billets, generated from the milling operation.

Each of these was machined with diverse machining parameters (at least one among ap , ae , f and n see Table 2.1 for more details) allowing to put twenty samples per surface. Moreover, the front was engraved to keep track of the sample's number and its orientation. It was fundamental to precisely catalogue each piece to later match the images with the machining parameters and the measured roughnesses.

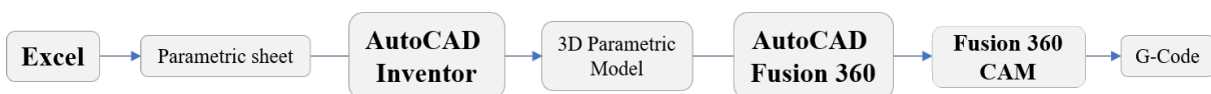


Figure 2.3: Step by step procedure to machine a single billet. Inside the CAM software the CNC company post processor was installed.

Figure 2.3 shows the procedure to obtain the final G-Code and consequently machine a billet. Once the machined parameters were chosen an Excel file was created. This was linked to the billets file so that the exact shape of the starting block could be identified. Afterwards a 3D parametric model was created using the software Autodesk Inventor and was linked with the sample Excel file. This allowed to speed the modelling phase because, once the link between the documents was set, all was needed to generate a new machined billet was to copy an exiting one and link it with another sample's Excel file and all the parameters would have automatically been uploaded. The 3D model was then exported into a CAM software and the machining phases were defined. Finally, the G-Code could be generated. But how were the machining parameters chosen? Choosing the admissible ranges of machining parameters was a big challenge. A good guess could be elaborated from the machining tables like the one shown in Figure 2.4 that was elaborated directly by the tool manufacturer. However, the machine itself represented a huge obstacle due to its low power and reduced size which was very prone to vibrate while machining with aggressive depth of cuts and so the upper bounds had to be changed. In addition, the cutting tool head diameter varied from 1.5 mm up to 4 mm creating the need of developing a MATLAB script (reported in Appendix A: Main Scripts, Section Machining Parameters Combination) that could calculate all the possible combinations, according to the different materials and tool sizes, and save them into an Excel file.

被削材 切削条件 刃径 D	機械構造用炭素鋼 (S45C~S55C)				合金工具鋼 (SKD, SCM, SUS)				調質鋼 (35~40HRC) (HPM, NAK)				鋼合金・アルミ合金			
	送り速度 (mm/min)		回転速度 (min ⁻¹)		送り速度 (mm/min)		回転速度 (min ⁻¹)		送り速度 (mm/min)		回転速度 (min ⁻¹)		送り速度 (mm/min)		回転速度 (min ⁻¹)	
	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面
0.1	70	150	33,600	60	130	33,600	30	60	33,600	110	210	52,500				
0.2	90	190	33,600	80	160	33,600	40	70	33,600	150	290	52,500				
0.3	120	230	33,600	80	170	33,600	60	120	33,600	180	360	52,500				
0.4	130	250	33,600	90	190	33,600	60	130	28,880	200	400	52,500				
0.5	130	250	32,550	90	190	26,250	60	130	23,100	210	420	52,500				
0.6	130	250	28,350	90	190	20,480	60	130	17,850	240	480	52,500				
0.7	130	250	25,200	90	190	17,850	60	130	15,750	260	530	52,500				
0.8	130	250	22,580	90	190	16,280	70	140	14,180	300	610	52,500				
0.9	130	250	19,950	90	190	14,180	70	140	12,600	340	670	51,450				
1	130	250	18,380	90	190	13,130	70	140	11,550	370	740	49,880				
1.2	130	250	15,750	90	190	11,030	70	140	9,770	360	710	42,530				
1.3	130	250	14,700	90	190	10,400	70	140	9,140	350	690	39,900				
1.5	130	250	13,130	90	190	9,350	70	140	8,300	340	670	33,600				
1.8	140	270	11,030	90	190	7,880	70	140	7,140	340	670	29,400				
2	140	270	10,190	90	190	7,350	70	150	6,620	330	650	25,200				
2.5	170	340	8,610	90	190	6,410	70	150	5,570	370	740	21,000				
3	180	360	7,250	110	210	5,570	80	160	4,620	420	840	16,800				
3.5	200	400	6,300	130	250	4,830	80	170	3,990	440	880	14,700				
4	220	440	5,670	130	260	4,410	90	190	3,680	470	950	12,600				
刃径D	Ad		Rd		Ad		Rd		Ad		Rd		Ad		Rd	
	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面	溝	側面
D<1	≤0.02D		≤0.05D	≤0.02D		≤0.05D	≤0.02D		≤0.05D	≤0.1D		≤0.1D				≤0.1D
1≤D<3	≤0.05D		≤0.07D	≤0.05D		≤0.07D	≤0.05D		≤0.07D	≤0.2D		≤0.2D				≤0.2D
3≤D<6	≤0.15D	≤1.5D	≤0.1D	≤0.15D	≤1.5D	≤0.1D	≤0.15D	≤1.5D	≤0.1D	≤0.2D	≤1.5D	≤0.2D				≤0.2D
6≤D	≤0.2D		≤0.15D	≤0.2D		≤0.15D	≤0.2D		≤0.15D			≤0.15D				

Figure 2.4: Cutting table of a family of tools based on the operation and material. Credit to MISUMI corp.

In Table 2.1 it can be noticed that to achieve a big range of roughness, the parameter space boundaries were very large; needless to say that it was not possible to machine using high radial and axial depth of cuts at the same time or use the maximums of these with small diameter tools. Before choosing the machining parameters a rough estimation of the maximum power needed was calculated using the interface created from *Iscar* [38] implemented directly online, which offered a quick solution to check the feasibility of the process.

Diameter	Parameter	Minimum	Maximum	δ increment
1.5 mm	a_p [mm]	0.1	0.5	0.1
	a_e [mm]	0.2	1.2	0.2
	n [rpm]	20000	25000	5000
	V_f [$\frac{mm}{min}$]	50	550	50
3 mm	a_p [mm]	0.1	0.6	0.1
	a_e [mm]	0.2	1.4	0.2
	n [rpm]	20000	25000	5000
	V_f [$\frac{mm}{min}$]	50	550	50
4 mm	a_p [mm]	0.1	2	0.2 up to 1 then 0.5
	a_e [mm]	0.2	0.8	0.2
	n [rpm]	16000	16000	0
	V_f [$\frac{mm}{min}$]	50	550	50

Table 2.1: Machining parameter ranges divided by tool diameter and relative to the aluminium billets. Please notice that the choice of one parameter influences the range of the others due to the limited power of the machine.

The CAM software automatically developed four different machining strategies according to the position and depth of the each rectangle.

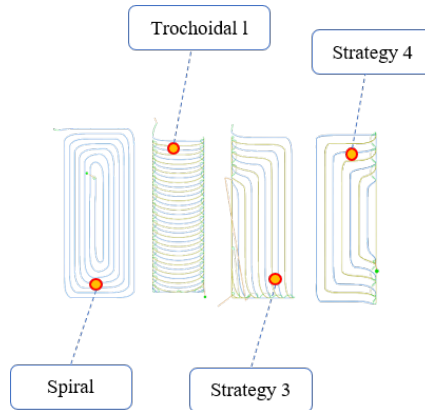


Figure 2.5: The different machining strategies developed by the CAM software.

These strategies are created to optimise the tool load, the machining time and to reduce the vibrations as the tool starts to bite into the material. Each of them was directly elaborated from the software without the possibility of directly choosing which one to adopt. The reason behind these choices, was not further investigated as it was embedded inside the software algorithms. Moreover, different strategies created different surface patterns, increasing the variety of the

samples and, as a consequence, of the overall dataset used to train the CNNs enhancing their adaptability and flexibility.



Figure 2.6: Two machined aluminium billets. The four different adopted machining strategies can be noted on the surface.

Once all the samples were machined the roughness of each one was measured. This process was conducted using an optical machine that was able to scan an entire surface every single time, speeding up the process. The measuring equipment, Keyence VR-500 [39], uses a white patterned led light to illuminate the surface and captures the reflection of this structured beam to recreate a 3D model of the scanned area. This method is able to achieve a precision of $\pm 2.5 \mu m$ when measuring heights and $\pm 0.1 \mu m$ regarding roughness measures. Both R_a and S_a were measured for each sample considering the widest possible rectangular area from the centre of each sample. While for the R_a only a single line parallel to the longest side of each rectangular areas and passing through the middle, was used.

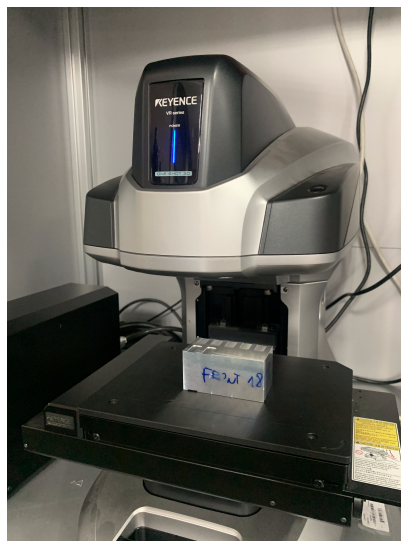


Figure 2.7: The measuring of the S_a and R_a of a machined billet. The optical machine can be noticed with the sample laying on the measuring table. The markings to identify the billet (blue letters) can also be noticed.

For each machined sample, a picture was taken. This was possible using a USB Camera (Bysameyee USB Digital Microscope [40]), characterised by a resolution of 640 by 480 and maximum zooming capability up to 1000, mounted directly next to the tool and fixed to the moving chuck.

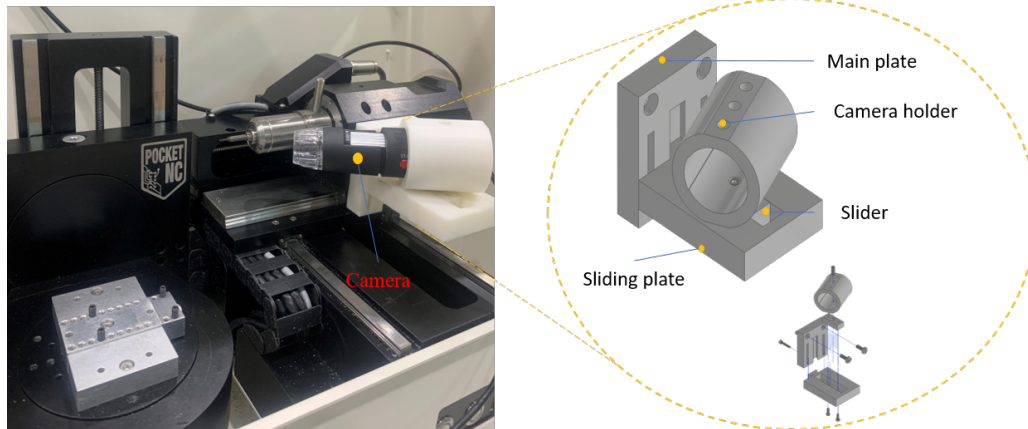


Figure 2.8: On the left is shown the global view of the camera mounted on the machine. On the right the camera holding device.

As it can be noticed from Figure 2.8 the holder was designed to allow the correct positioning of the camera by guaranteeing two linear degrees of freedom and a rotational one. The 3D-printed structure is light, and fast to be produced and assembled. Two pictures were taken for each sample: one with the brightness of the led light of the camera at 100% and the other one with about 50%. This was performed to mimic the conditions of an ideal testing environment, in which the light is well calibrated with the surroundings and a real working ambient that has light noises and shadows that can spoil the final images.

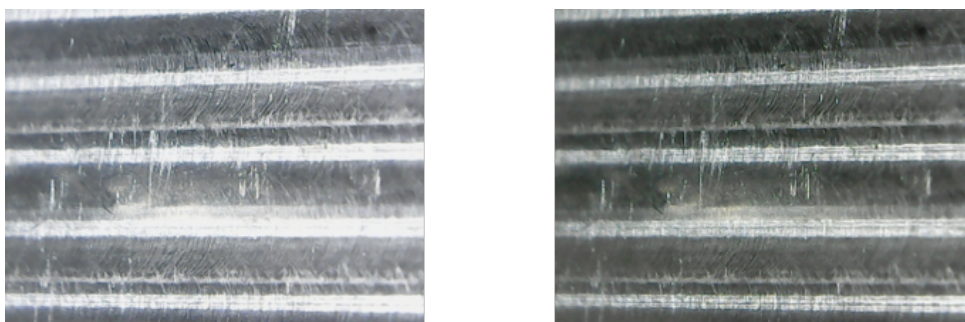


Figure 2.9: On the left is portrayed the full bright taken picture. On the right the half bright one.

One of the problems of this dataset was its dimension. Datasets like the MNIST [41] or CIFAR-10 [42], which are the benchmarks used to test most of the newly developed CNN, have 60000 images for the training phase and 10000 for the testing. This is achieved after years and years of data gathering and analysis. However, is very hard to achieve such sizes in a machining environment, especially with a small machine like the one used in this work. As a consequence image enhancement is used. This technique allows to generate additional data by modifying

the existing one, changing some of the features and parameters. In this work it was applied six times for every image, drastically increasing the number of samples. In detail, the *ColorJitter*, *Sharpness*, *Orientation* and *Position* were changed. In particular, the *Orientation* was randomly changed from 0 to 90 degrees and the *Sharpness* was adjusted using a *sharpness_factor* up to 20 (see Appendix A: Main Scripts, Section Data Enhancement, to find the code and more details).

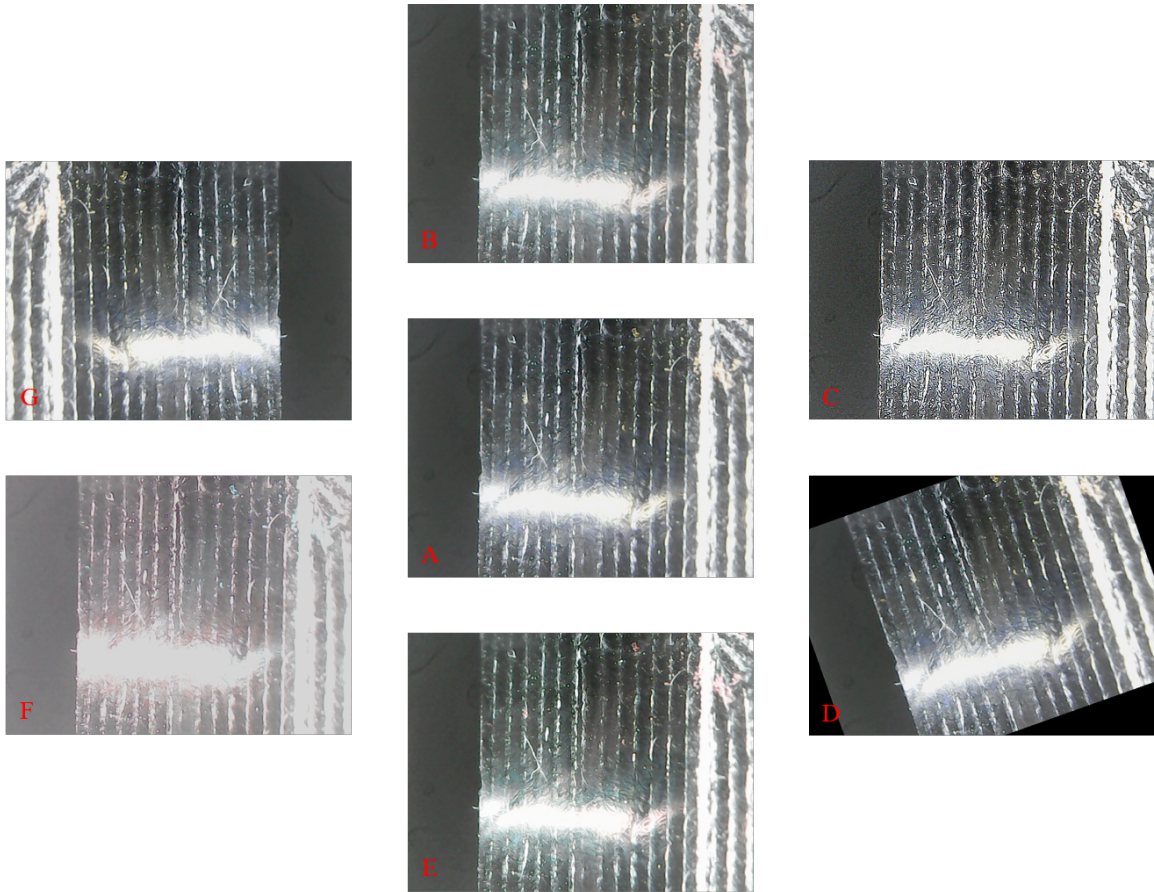


Figure 2.10: Applied enhancements. [A] Real image, [B] Jitter change, [C] Sharpness change, [D] Rotation, [E] Jitter change, [F] Jitter change and [G] Mirror.

All of this was computed using a Python script (Appendix A: Main Scripts, Section Data Enhancement) that was developed using the PyTorch package [43] which is specifically designed to create datasets for machine learning applications. In the end, the dataset is composed of 4886 pictures that are linked to their roughness. Even if R_a is still more used and known in the machining industry, in this work, surface parameters were chosen instead of linear because the measuring optic instrument is better suited to measure such parameters. In addition, the nature of the machining operations is mainly related to generate flat surfaces and consequently the whole quality of the area is relevant instead of multiple lines. Nevertheless, creating a dataset that uses a different roughness metric is a trivial operation in this case because the data were already gathered.

2.1.2. CNN estimation models

There are many available models that have been developed by researchers that could be used to perform this regression task of estimating the S_a directly from the images of the machined surfaces, however, most of them are born as classification ones instead. In a classification task, the trained model usually predicts the probability of an instance or image belonging to a class, in a regression one instead, the output of the model is continuous as the architecture is trying to predict continuous values. This complicates the choice because all the architectures are specifically designed and optimised to perform such tasks. However those models' structures are able to decompose images at different levels to extract, firstly the simplest features, like horizontal and vertical lines, and in the deeper layers of the CNN the most complex ones. Because of this, they can still be adapted to perform regression tasks by changing the *Fully Connected* layer from a logical one too, as in this case a *Linear Regression* one. This condenses the outputs (whose size changes according to different architectures but in this case was always a row vector) of the CNN filters into a single value that represents the actual prediction of the model. In this work only two models were used and compared: the ResNet50 [20] and Xception [21]. The first one was chosen because it is one of the most used ones, was adopted before in the same field of estimation and represents the benchmark when testing the developed architectures.

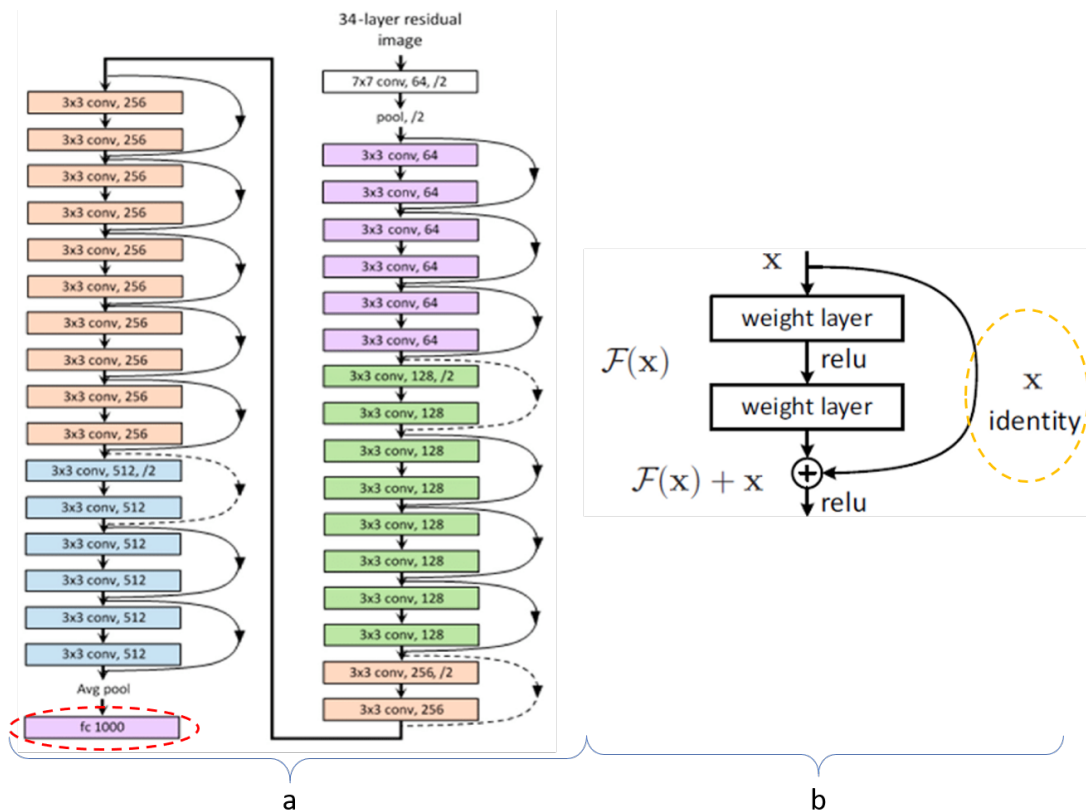


Figure 2.11: On the left (a) it is represented the general structure (not the complete one) of the ResNet50 with the fully connected layer highlighted by the red dotted circle. On the right (b), the structure of a residual block with the skip connection is highlighted by the orange dotted circle. [20].

The ResNet50 has fifty layers of convolutions and a total number of training parameters of more than 25 million, making it part of the family called *Deep Convolutional Neural Networks*. These architectures are able to detect low to high-level features but suffer from the problem of the vanishing gradients in which, during the backpropagation step, the gradient is too low and is not able to upload the weights and so the network is not trained anymore. This can cause a rapid convergence of the training session but with poor results or a bad performance of the network. The deeper the model the higher the chance of this happening because the information is not able to penetrate into the deeper layers. To reduce the chances of this happening the developers introduced residual layers that, combined with the *Relu* (presented in Chapter 1) activation function, allowed to create deeper architectures keeping a low computational time and no increase in the number of parameters to be trained. The skip connection (see Figure 2.11) convolutes the input with identity filters to resize its last dimension, making it compatible with the output. This connection jumps over all the convolutional filters and arrives directly at the output of the block, propagating the information to the deeper layers of the network. If one layer gets a zero gradient upload then, that particular residual block, has only to learn an identity function and will not hurt the performances of the overall architecture.

Deep networks used to suffer from the problem of overfitting. This happens when the network is too good at predicting the training dataset but lacks accuracy when tested on a new dataset never seen before. The Xception model tries to avoid this problem by reducing the number of parameters needed per layer but still extracting the features at most levels of definition.

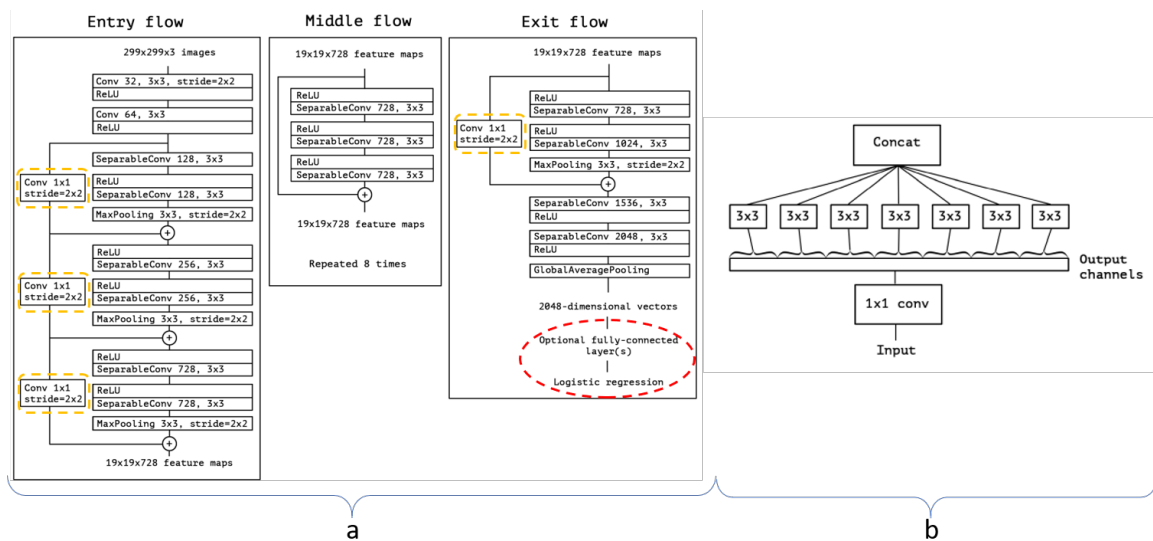


Figure 2.12: On the left (a) is shown the general structure (not the full one) of the Xception with the residual layers highlighted by the orange dotted rectangular boxes and the fully connected layer by the red dotted circle. On the right (b), the structure of a Depthwise Separable Convolution block is schematised. [21].

The problem with traditional convolution operations is that they require a lot of memory and time because the filters are convoluting on three dimensions. To reduce the use of resources

extreme Depth Convolutions were introduced. The input is firstly convoluted with n_f 1 by 1 by n_c convolution filters (point-wise convolution) that work only on the number of channels and so the third dimension. Consequently, 2D filters are applied to each channel separately and the results are concatenated along the channels' direction (depth-wise convolution). The first step allows the network to compress the information of each channel into a single output per cell. The last convolution instead allows us to consider each channel separately from the others gathering information from also the surrounding elements. The Xception model proved to be more accurate compared to the ResNet50 regarding classification tasks and still less sensible to the vanishing gradient phenomena due to the presence of residual layers that can be seen in Figure 2.12. In addition, it has a lighter architecture with fewer connections between each block (22.8 million parameters).

Both models were implemented using the Pytorch platform on the Python base. In addition, the fully connected layer of each was modified with linear activation functions and only one output node because initially they were provided with logic regression activation functions and 10000 output nodes that are suitable for classification purposes.

2.1.3. Training of the models

To train each model the dataset was split into three main parts. Eighty percent was used for training purposes, ten percent for validation and the remaining ten percent for testing. The division was based on similar studies ([16], [17], [18]) and based on the size of the dataset. Because the latter was limited, most of the samples were used to train the models while only a small amount was used to test them. If the size of the training dataset is not big enough, under-fitting can occur and it had to be avoided, especially in a regression task like this in which the output was used as a control input. Validation is used to tune the hyperparameters (in this case it was only the learning rate, that was changed according to Table 2.3) and to control that overfitting does not occur while training. The mini-batch descent algorithm [44] was used with a size of sixteen pictures per batch. This is a variation of the gradient descent algorithm that trains the model over small batches of the overall training dataset instead of using the whole one, uploading the weights of the networks gradually instead of once per epoch. Its choice was mainly driven by the limitations of the GPU which was an Nvidia GEFORCE RTX with a capacity of 12 Gb. A bigger batch size would have required fewer iterations per epoch but would have occupied more memory and a slower convergence would be verified. In theory, the validation and testing tasks can be fused into a single one performed on a single database; however, to increase the accuracy it is better to tune the hyperparameters of the model on a single database and then test the final performance of the model on a new dataset that the network has never seen before (testing phase).

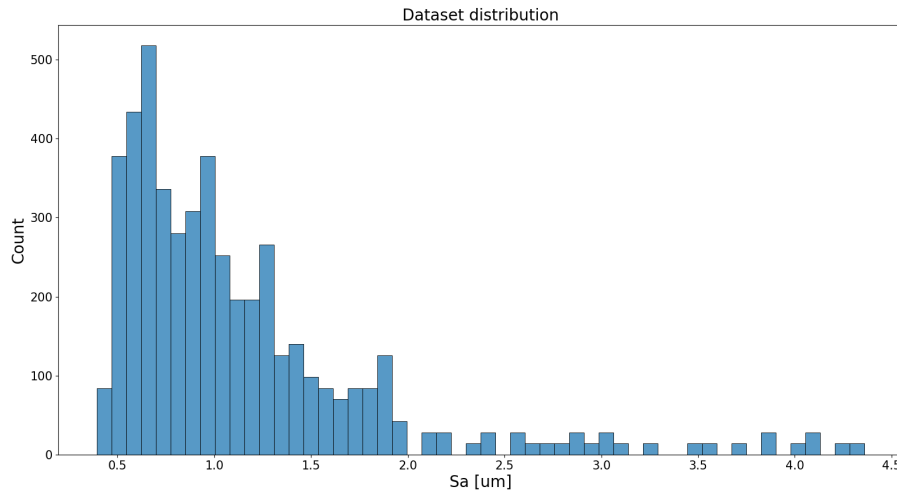


Figure 2.13: Total dataset distribution among different roughness values.

As shown in Figure 2.13 the samples' distribution was not balanced. This was due to the fact that the machine had some limitations in terms of power and tool size so the machining was not aggressive compared to an industrial milling machine. To assess this issue the three datasets were created by gathering the samples by ranges of S_a and then, for each range, elements were picked to from the three datasets with the same percentages reported before. So for example, if in the range between 2.5 micron and 3 μm there were 100 specimens, 80 were added to the training database, 10 to the validation and 10 to the test. To optimise even more the performance of the training, the images had to be processed before being given as input into the network. In detail the *ResNet50* and *Xception* architecture require the images to be pre-processed differently because of the networks' architectures. Specifically, the pictures had to be cropped from the centre and normalized with different parameters (Table 2.2).

Model	Operation	Operation Parameters
ResNet50	CenterCrop	224
	Normalize	mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
Xception	CenterCrop	229
	Normalize	mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]

Table 2.2: Pipeline of pre-processing transformations.

The *Adam* optimiser [46] was chosen because it is a more advanced one compared to *Gradient Descent*. The learning rate varied from 0.0005 down to 0.00005 as the number of epochs increased to avoid the saturation of the errors.

Epoch number	1	2	3	4	5	6	7	8	9	10
Learning rate	0.0005	0.0005	0.0005	0.0005	0.0003	0.0003	10^{-4}	10^{-4}	10^{-4}	5×10^{-5}

Table 2.3: Learning rate evolution over 10 epochs.

Three different *loss functions* were tested to find out which one was more suitable to perform the regression task and all models were trained for at least 64 epochs. In particular the *MSE* was adopted, which was introduced in Chapter 1, the *Mean Absolute Percentage Error (MAPE)*:

$$MAPE = \frac{1}{m} \sum_{i=1}^m \frac{|y^{(i)} - \hat{y}^{(i)}|}{|y^{(i)}|} \quad (2.1)$$

and the *Huber Loss*:

$$l^{(i)} = \begin{cases} \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2 & , \text{if } |y^{(i)} - \hat{y}^{(i)}| < \delta \\ \delta \cdot (|y^{(i)} - \hat{y}^{(i)}| - \frac{1}{2} \cdot \delta) & , \text{if } |y^{(i)} - \hat{y}^{(i)}| \geq \delta \end{cases} \quad (2.2)$$

The *Huber Loss* is effective in minimizing the outliers and uses the advantage of the *MAE* error to not put penalise too much the big ones. It, in fact, represents a hybrid between the *MAE* and *MSE* loss functions. Its δ represents the threshold set to clip the gradients to δ for residual (abs) values larger than δ . Basically, when the errors are too large the training can be influenced and the weights may be updated too much in the direction of minimising such errors but increasing the residual on the data that has already been learned. To avoid such, the *Huber Loss's* derivative is constant and equal to δ after the residuals are bigger than δ itself. In this work, its value was set to 0.8 as this threshold was considered more than enough in terms of errors for the estimation of the S_a and also because the outliers with bigger errors were really hard to remove and always affected the training of the model. To better tune this loss function, it should be data-driven [45]. For the test phase also the *coefficient of determination (R^2)* was used when evaluating the overall predictions.

$$R^2 = 1 - \frac{\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^m (y^{(i)} - \bar{y})^2} \quad , \text{ where } \bar{y} \text{ represents the mean of all the predictions} \quad (2.3)$$

These metrics were chosen because they can provide different analyses about the training process, the validation and the test. In addition, they supplied the information needed to choose the best model accommodating the upsides and downsides of each of them. Each time a model completed all the pre-set epochs of training a complete analysis of the process was computed. This included the evaluation of the training error and validation errors for every iteration to identify the effectiveness of the chosen learning rates and the starting of overfitting. There is not a unanimous way to identify such behaviours quantitatively, usually ML engineers are experienced enough to look at the shape and slope of the training and validation curves to understand if the model is overfitted or not. In this case, a more quantitative analysis was used based on the evaluation of the training curves and test ones. This consisted in stopping the training process to the epoch in which there was no change in the slope of the validation curves from a negative descending trend from an ascending one (look for example at Figure 3.1 in which

the validation curves suddenly jumps). In addition for each epoch, the correlation plot between the *ground truth* and *predictions* was plotted (refer to Chapter 3 and take as an example Figure 3.3) to observe its evolution and the presence of outliers. This was produced both over the test database and the complete one. The test losses evolution over all the epochs were analysed to identify the best model and the starting of the overfitting phenomena. The equivalent was done using the same errors but computed over the whole dataset (as for example shown in Figure 3.2). In addition, the distribution of the *AE*, computed over all the samples, was observed to quantify the quality of the predictions and of the training process. Finally, the best models were taken and compared. The comparison was first done over the test errors. Then it was extended to the whole dataset comparing the previously used metrics (*MSE*, *MAE*, *MAPE* and R^2). Moreover, the correlations were also plotted one over the other to visually contrast them and identify the outliers. Finally, the *AE* of all the analysed models was computed among the different S_a ranges to identify the most critical areas of each architecture-loss combination. This analysis was combined with the *AE* distribution for each one computed over all the samples in the database. To finally get a visual view of the performance the best model was tested to predict four random images picked from the test dataset.

2.2. Optimisation method

The optimisation of these machining operations is a particular task that must be considered carefully. In this case, no simulator or model was available to generate samples because the machine is a particular one that cannot be compared to the industrial ones. Its size, stability and stiffnesses are far below the traditional ones. As a consequence, every time a new sample needed to be created a large amount of time was used. In addition tests and trials using a never-seen combination of parameters could resolve in a catastrophic failure of either the tool or one part of the machine itself like the motors or the chuck, leading to a huge increase in downtime. The choice of the optimisation method was then restricted to a few ones but, among them, *Bayesian Optimisation* using a *Gaussian Process Regression* was thought to be the best from the beginning. Initially *Reinforced Learning* (RL) was taken into consideration because of its applications in the robotic industry and effectiveness in learning tasks, being similar to the learning process of humans. However, this learning methods require many samples, millions of iterations and it is based on a trial and error approach which was not suited for this work because too risky and dangerous to the CNC itself. Gradient-based methods were also considered, but usually required the construction of a regression model or the development of an additional ANN to predict the chosen outputs of the machining process, reducing the adaptability of the system and increasing the number of samples required.

2.2.1. Introduction to Bayesian Optimisation

Imagine that an oil drilling company has to decide where to drill the next hole to find a new oil field. Beyond any doubt, the company cannot make many trials because each time all the

equipment must be moved to the new site and drilling is not an easy operation but requires plenty of time and money. In addition, they cannot build a model of the all underground of that area because it will require scanning for thousands of square kilometres. This demands the use of a method that is only based on what the company knows so far about the environment and on the previous data that was gathered during the previous drilling operations. The explained example is the perfect candidate for the optimisation family of methods called *Black Box* optimisation in which, the function that maps the inputs to the outputs is not known and the method does not need to know it [47].

This optimisation method is focused on minimising or maximising an *objective function*:

$$\max f(\mathbf{x}) \text{ or } \min f(\mathbf{x}) \quad , \text{ where } \mathbf{x} \in A \quad (2.4)$$

Where the input \mathbf{x} belongs to \mathfrak{R}^d and d is usually smaller than ten. In addition, f must be continuous and expensive to evaluate and calculate its derivatives including the first-order ones. A represents the space in which the optimal \mathbf{x} is located and it is usually an hyper-rectangle [48].

Bayesian optimisation is mainly composed of two main parts: a *statistical model* that mimics the objective function and an *acquisition function* that is used to evaluate the created "model" and understand where to find the optimal points. An optimisation loop using Bayesian optimisation is divided into three main phases. Firstly the initial samples are generated and used to fit the chosen model. As a rule of thumb it is recommended to generate at least $10 \cdot d$ samples [49]. These can be created by picking up random spots among the all parameters space or using sampling strategies like Latin-Hypercube sampling. Once the model is fitted the acquisition function runs all over the space A and finds the best point in order to maximise or minimise the objective function. A new sample is then created based on those chosen inputs and the model is then uploaded. This procedure is then repeated until a stopping criterion, usually based either on the number of optimisation iterations or on the difference between two consecutive outputs, is met.

2.2.2. Choosing the statistical model

The regression function chosen to fit the inputs is very important in a Bayesian optimisation system. This is because it represents the groundings on which the acquisition function will look for new points. In this work, a *Gaussian Process Regression (GPR)* was used to fit the available data. This regressor assumes that the objective function values associated with different inputs have a joint Gaussian distribution [50], [51]. The choice of this regression model was driven by the nature of our problem and the aim of the work. In this application, the intention was not to make a regression model that could associate the machining parameters to the S_a but to find the best combination of those regardless of the input-to-output mathematical relationship. In addition, the outcome of the process is by itself of a statistical nature, due to the fact that with

the same machining conditions there is the possibility that two different final roughnesses could be measured due to variation of the external environmental conditions, the mechanical stability and the reliability of the used machine. This is why a GPR is a well-fitted candidate to relate the inputs to the outputs of the machining process because creates a model with a confident interval, fitted on the available data-points. As it can be understood a statistical regression can be identified with its *mean function* $\mu(\mathbf{x})$ and *covariance function* $k(\mathbf{x}', \mathbf{x})$ whose importance and characteristics will be explained later. In particular, the mapping relationship between input and output can be assumed to be :

$$y = f(x) + \xi \quad (2.5)$$

Where ξ is the noise that, by assumption, follows a Gaussian distribution. In detail, the GPR is able not only to provide the mean value of the estimation but also its variance quantifying the uncertainty of the prediction [52]. This is very useful when dealing with complicated models because it gives the opportunity to evaluate the quality of the estimation and the admissible ranges. Mathematically, given the real observation data \mathbf{y} and its prediction $\hat{\mathbf{y}}$, based on a given set of unknown inputs \mathbf{X}' and known inputs \mathbf{X} , they can be linked using a GPR in the form of a joint distribution:

$$\begin{bmatrix} \mathbf{y} \\ \hat{\mathbf{y}} \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) & \mathbf{K}(\mathbf{X}, \mathbf{X}') \\ \mathbf{K}(\mathbf{X}', \mathbf{X}) & \mathbf{K}(\mathbf{X}', \mathbf{X}') \end{bmatrix} \right) \quad (2.6)$$

In this case, the inputs are matrices in which \mathbf{x}_i represents the *i-th* single sample. The covariance function $k(\mathbf{x}_i, \mathbf{x}'_j)$, also called *kernel*, is one of the cardinal elements of the GPR because it holds the links between the input points. This function must be chosen according to the problem type and the data type. In this study, two kernels were considered: the *Exponential* kernel,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp \left(-\frac{r}{2l} \right) \quad \text{where } r = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)} \quad (2.7)$$

and the *Matern 5/2* kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2} \right) \exp \left(-\frac{\sqrt{5}r}{l} \right) \quad \text{where } r = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)} \quad (2.8)$$

The *Exponential* one is among the simplest ones that could be used a. The *Matern 5/2*, instead is more complicated and has a higher computation time compared to the previous one. It was chosen because was already employed from the previously considered studies [49], [51], [52] that considered its application in mechanically related fields. These kernels are dependent on the signal variance σ^2 , the characteristic length-scale l and the distance between each pair of inputs. The first one works on the total magnitude of the function as a scale factor, the second instead

sets the boundaries for the sensitivity of the function to the distance between points. A high value of l means that to consider two inputs uncorrelated they have to be very far; vice versa a low one needs a lower distance to consider them uncorrelated. These hyperparameters are delicate because they must be tuned, based on the inputs, to improve the fit of the function. The first kernel is one of the most basic ones and has a wider peak compared to, the more complicated, second kernel which possesses a narrower peak but decays faster as the distance between points increases [53]. As explained before the *Matern 5/2* kernel is more sensitive to a sharp change of the data and it is recommended to model smoother series [56]. In particular, the mapping relationship between input and output can be assumed to be :

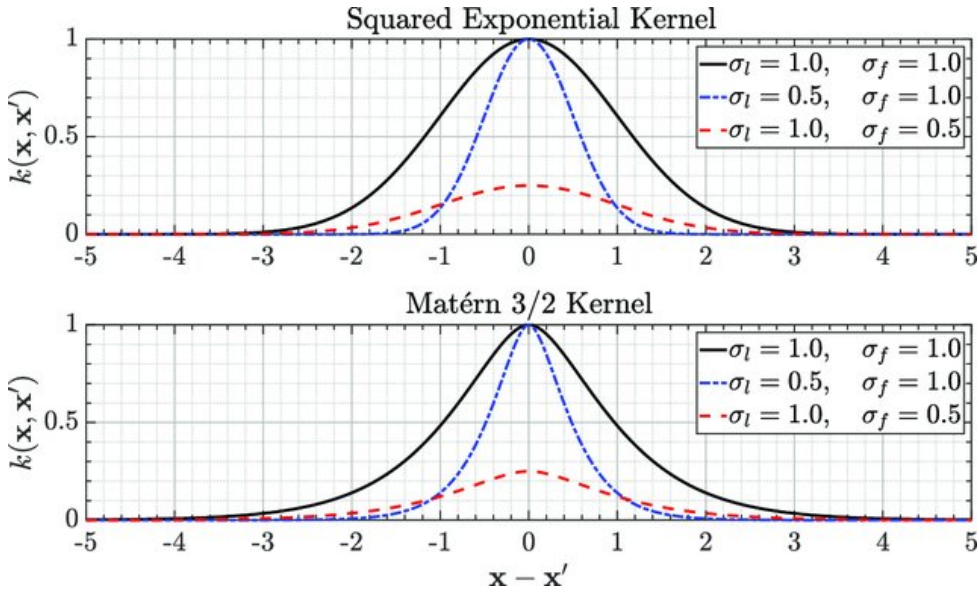


Figure 2.14: Kernels comparisons based on different hyperparameters. Credit to [53].

There are many approaches to find the optimal hyperparameters. Some of them use statistical methods and others use gradient-based methods. In this work, the *maximum likelihood estimate (MLE)* approach was used which aims to maximize the posterior likelihood probability. If we indicate with $\boldsymbol{\eta}$ and $f(X)$ as the observations computed by the GPR at each input point the method aims to maximise the probability of $P(f(X)|\boldsymbol{\eta})$ in mathematical words:

$$\hat{\boldsymbol{\eta}} = \underset{\boldsymbol{\eta}}{\operatorname{argmax}} P(f(X)|\boldsymbol{\eta}) \quad (2.9)$$

To find those values the optimizer initialises some random hyperparameters and then uses an iterative method to find them. To avoid getting stuck into local minimum the process is started more times from different random positions. In this work, the package developed by Sheffield University called GPy [54] and GPyOpt [55] was added in Python to model the GPR. In this case, the optimiser followed the Broyden–Fletcher–Goldfarb–Shanno algorithm [57] with a maximum number of iterations of 2000 and 10 number of restarting.

Once the GPR is fitted then predictions can be made all over the hyper-rectangle. In particular, the conditional probability can be used to generate unknown predictions, $\hat{\mathbf{y}}'$, from unknown inputs \mathbf{X}' , based on the all available inputs, \mathbf{D} . As stated before both the mean and the confidence interval can be predicted.

$$p(\hat{\mathbf{y}}' | \mathbf{X}', \mathbf{D}) = \mathcal{N}(\hat{\mathbf{y}}' | \mu', \sigma^{2'}) \quad (2.10)$$

The mean can be calculated using the covariance matrix and function:

$$\mu' = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{y} \quad (2.11)$$

and the variance :

$$\sigma^{2'} = k(\mathbf{X}', \mathbf{X}') - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} \quad (2.12)$$

2.2.3. Choosing the acquisition function

As stated before the acquisition function has to evaluate the objective function and suggest the next best input to minimise or maximise it. This choice must be done by finding a compromise between exploration and exploitation. The first one implies looking for other areas in which there is the possibility to find better parameters but, because it is an unknown space, the risk of getting worse instead of improving is present; the second instead, aims to search in the neighbour of what the model already knows to be safer and work close to the minimum. Among all the available functions the one used in this work was the *Expected Improvement (EI)*, which provides a good balance between exploration and exploitation and has been already adopted in similar studies [49], [51], and [52].

2.2.4. Assembling the optimiser

Bayesian optimisation was applied to find the combination of the best machining parameters of a slotting operation using the trochoidal tool path. Only the V_f and the rpm were optimised. This was done because due to the nature of the machine it was not possible to find a range of depth of cuts that was compatible with an equally wide one of feeds and speeds. In addition, when real-time optimisation has to be performed, a modification in a_p and a_e required communication between the CAM and the milling machine to generate a totally new G-Code with different tool paths. The optimisation process had a multi-objective nature because two objective functions, in contrast to each other, were created. The first one regards the minimisation of the S_a of the machined surface, while the second the maximization of the MRR of the machining process.

Parameter	Minimum	Maximum
a_p [mm]	0.6	0.6
a_e [mm]	0.6	0.6
n [rpm]	11500	13500
V_f [$\frac{mm}{min}$]	500	950

Table 2.4: Corners of the hyper-space of the inputs of the process.

As it can be seen from Table 2.4, the depth of cuts were fixed to values that were admissible and in the ranges advised by the tool manufacturing company table. The feed rate was varied between a reasonable minimum to keep the machining time (which varied between 50 seconds to 2 minutes per sample according to the different chosen V_f) feasible and a maximum that required a machining power within the limits of the actual CNC. The rotational speed was chosen around the neighbour of the advised one from the machining table. The tool used to machine all the samples was always a new one that had no sign of any wear or defects, as it was used only to manufacture a maximum of 4 samples.

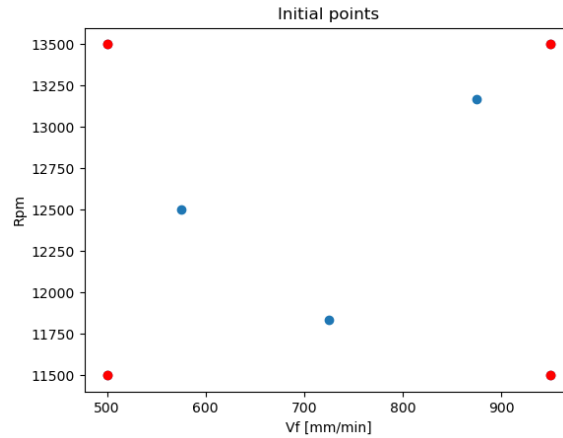


Figure 2.15: Initial points generated using Latin Hypercube Sampling.

The initial inputs were generated from the available space using the Hyper Cube Latin sampling method (blue dots in Figure 2.15). In addition, the four corners of the hyper-rectangle were added [58] to generate a total of seven final inputs (red dots in Figure 2.15).

In a multi-objective optimisation problem that is not a unique best solution but instead, a group of them. This *Pareto Front* is represented by all the non-dominated solutions of the problem in which no objective can be improved without sacrificing at least one other objective [59]. The aim of the optimiser is to find that front and then obtain the best solution based on the previously imposed constraints. The acquisition function used in the model was only able to handle single objective functions and was not suitable to operate with multiple ones. Even if there are extensions of it like the *hypervolume expected improvement (HEVI)* still, the solution adopted was to create a unique objective function. This choice was taken because, when

implementing the online optimisation task the system had to be able to choose between a range of different parameters that were generated all at the same time and not upload after upload. This required the creation of multiple objective functions that had to be optimised at the same time and in the same optimisation loop. In addition, according to previous studies, when using *HEVI* for each objective function a single GPR must be created and fitted; in this case however, the *MRR* is identified by a deterministic function and consequently its related GPR cannot be created. In the end, the final objective functions were generated by summing the two outputs of the process each of them multiplied by a weight that could vary between zero and 0.5.

$$f_j(\mathbf{x}_i) = w_j \cdot y_{1,i} + (1 - w_j) \cdot y_{2,i} \quad , \text{where} \quad \begin{cases} y_{1,i} = -\frac{MRR_i}{60} \\ y_{2,i} = Sa_i \end{cases} \quad (2.13)$$

As it can be noticed from Eq. 2.13 the *MRR* is made negative to aim for the minimisation of the final objective function. Each time the weight w is increased the solutions move toward a higher removal rate to the detriment of a higher S_a . Before fitting the GPR the inputs were scaled down because of their different order of magnitude. In particular, the feed was divided by 100 and the rpm by 1000.

Wiegths	w_1	w_2	w_3	w_4
Value	0	0.15	0.35	0.4

Table 2.5: Weights used to create the different objective functions.

2.3. Assembling the final system

Once the monitoring and the optimiser were developed the combination of both was tested combining them in a system that was able to predict the surface roughness of a machined surface and change the V_f and rpm if the former was bigger than the imposed limit. To accomplish so, the programmed 'brain' had to be able to interact directly with the machine. As reported in the Introduction each company builds its machines using a particular type of controller that interfaces with the hardware using the universal G-Code. Even in this case the machine was controlled by a web Page specifically designed using the *Linux* interface.

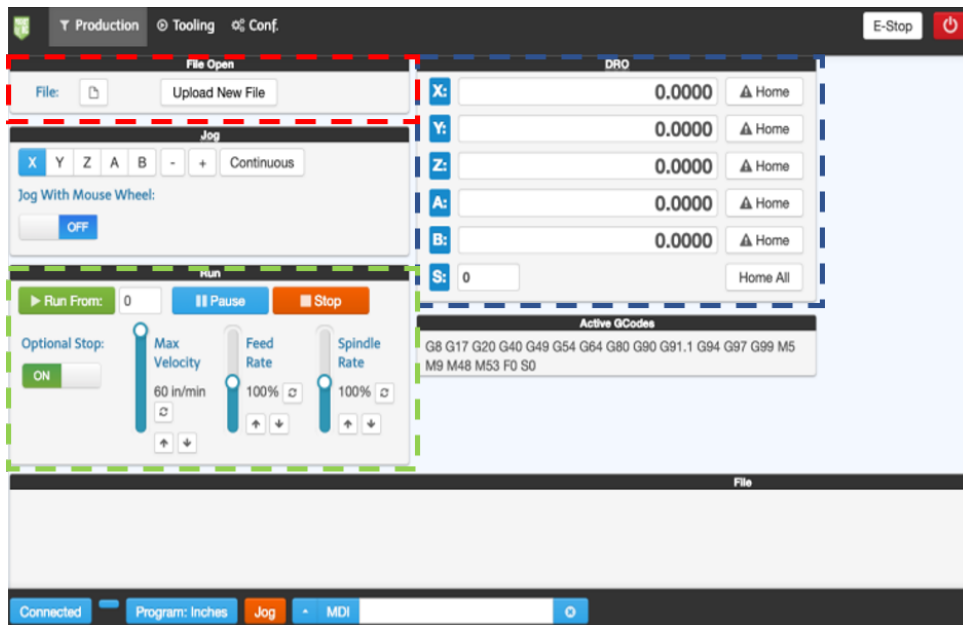


Figure 2.16: Web built interface that controls the CNC machine. The red box represents the part related to the uploading of a new G-Code, the orange one directly acts on the machine movements and can change the V_f and rpm while the tool is cutting, while the blue one indicates the axis coordinates in real time.

The computer had to be able to directly talk to the interface shown in Figure 2.16 by clicking the needed buttons and uploading the correct files. To perform such task a script was created with the *selenium* package [60] that enabled web scraping actions. The steps to get the new machining parameters can be synthesised as follows:

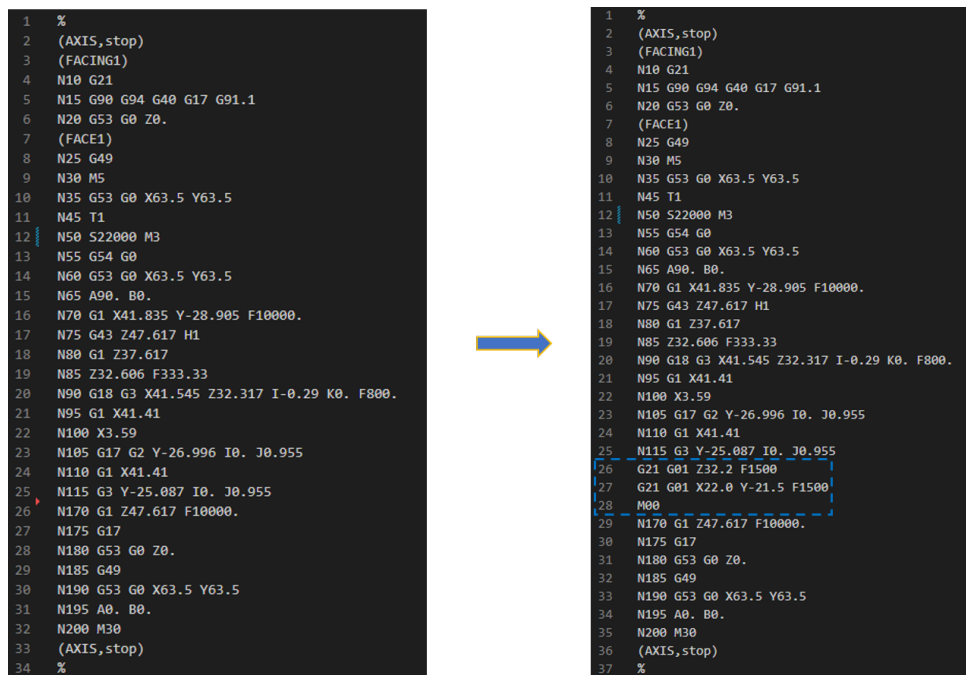
1. Initial G-Code modification
2. G-Code uploading on the interface
3. Machining operation
4. Machined surface's picture acquisition
5. Optimisation
6. G-Code modification
7. G-Code uploading on the interface
8. Second machining operation

Because of the design of the machine's controller, the changing of the machining parameters could not be done while the tool was cutting. This was because even if the *override* functions are available and implemented in the interface, these can only modify the feed rate and the cutting speed by an integer percentage amount that is fixed to be $\pm 5\%$ without the possibility to change them continuously. In addition the monitoring system would had problems to identify

the machined area while the tool was working due to its presence in the middle of the picture and the taken images would have been blurred or spoiled by the vibrations caused by the cutting operation. Moreover, it would have been hard to fully identify the finished surface because the tool movement is not always linear. This is why it was taken the decision to perform the monitoring and optimisation once the machining process was over.

2.3.1. Modifying the G-Code

Once the G-Code was compiled by the CAM program it had to be modified in order to move the camera to the centre of the machines surface and take a picture. To achieve so a *Python* script was made that was capable of reading the whole code to find the start and the end of the machining operation. Once these are identified, the maximum and minimum of each linear coordinates are recognised so that the complete tool path can be characterised. With these values the codes adds four lines of commands in the G-Code that move the tool to the correct *X, Y, Z* position and stop the program waiting for the picture to be taken.



```

1 %
2 (AXIS,stop)
3 (FACING1)
4 N10 G21
5 N15 G90 G94 G40 G17 G91.1
6 N20 G53 G0 Z0.
7 (FACE1)
8 N25 G49
9 N30 M5
10 N35 G53 G0 X63.5 Y63.5
11 N45 T1
12 N50 S22000 M3
13 N55 G54 G0
14 N60 G53 G0 X63.5 Y63.5
15 N65 A90. B0.
16 N70 G1 X41.835 Y-28.905 F10000.
17 N75 G43 Z47.617 H1
18 N80 G1 Z37.617
19 N85 Z32.606 F333.33
20 N90 G18 G3 X41.545 Z32.317 I-0.29 K0. F800.
21 N95 G1 X41.41
22 N100 X3.59
23 N105 G17 G2 Y-26.996 I0. J0.955
24 N110 G1 X41.41
25 N115 G3 Y-25.087 I0. J0.955
26 N170 G1 Z47.617 F10000.
27 N175 G17
28 N180 G53 G0 Z0.
29 N185 G49
30 N190 G53 G0 X63.5 Y63.5
31 N195 A0. B0.
32 N200 M30
33 (AXIS,stop)
34 %

```

```

1 %
2 (AXIS,stop)
3 (FACING1)
4 N10 G21
5 N15 G90 G94 G40 G17 G91.1
6 N20 G53 G0 Z0.
7 (FACE1)
8 N25 G49
9 N30 M5
10 N35 G53 G0 X63.5 Y63.5
11 N45 T1
12 N50 S22000 M3
13 N55 G54 G0
14 N60 G53 G0 X63.5 Y63.5
15 N65 A90. B0.
16 N70 G1 X41.835 Y-28.905 F10000.
17 N75 G43 Z47.617 H1
18 N80 G1 Z37.617
19 N85 Z32.606 F333.33
20 N90 G18 G3 X41.545 Z32.317 I-0.29 K0. F800.
21 N95 G1 X41.41
22 N100 X3.59
23 N105 G17 G2 Y-26.996 I0. J0.955
24 N110 G1 X41.41
25 N115 G3 Y-25.087 I0. J0.955
26 G21 G01 Z32.2 F1500
27 G21 G01 X22.0 Y-21.5 F1500
28 M00
29 N170 G1 Z47.617 F10000.
30 N175 G17
31 N180 G53 G0 Z0.
32 N185 G49
33 N190 G53 G0 X63.5 Y63.5
34 N195 A0. B0.
35 N200 M30
36 (AXIS,stop)
37 %

```

Figure 2.17: On the left is pictured the original G-Code. On the right the modified one with the added parts highlighted by the blue dotted box.

The start and end of the operation can be easily identified because the CAM introduces the name of the specific machining operation as the opening line (*(FACE1)* in Figure 2.17, while *(FACE1)* represents the title of the G-Code) while the end is characterised by an extremely high feed rate that is used to move back the tool to the standard position (line 26 in Figure 2.17). After inserting the coordinates positions in the added lines, the command *M00* is added to make the machine pause until the picture is taken and then the *Python* code automatically presses the start button on the controlling software of the CNC and the tool is positioned at the home

position.

2.3.2. Parameters modification strategy

If the surface roughness S_a was above the imposed limit by the user the system had to take some action and try to change that. To achieve that, two strategies could be used. The first one is the most simple one and is based on the history of the optimization. Instead of looking for new points the system can just look at all the history of the previously found combinations and choose one that will machine a surface with the required surface roughness but still keeping a high removal rate. The second instead, is a more dynamic that requires the system to look for new points never seen before and it is very effective when the surrounding environment is changing and so the conditions are time dependant. In this case the decision is usually based on a previous history that has been limited to only the most recent samples [61] so that the next prediction can be focused on the most recent environmental conditions. In alternative, the system decision can also be based on a pre-calibrated model [62] that is initialised using on site acquisitions right before performing the optimisation so that the statistical model is directly tailored based on the most recent environmental conditions [63].

For this particular case, the first approach was used. Even if, it is a less precise and simpler one it was more suitable for the type of environment in which the tool was cutting. This was mainly due to the actual machine and material. As stated before, mostly aluminium was machined using not aggressive parameters because of the low power and stability of the actual CNC. This deeply influenced the state of the tool that did not worn enough to be considered an environmental condition change even when machining with the most aggressive combination of parameters. In addition, the on-site calibration of the model was not feasible because machining a sample and measuring it required much time and resources and it was not representative of a real industrial context in which the machine must always be ready to machine. In the end, after the CNN had estimated the surface roughness of the newly machined surface, if this was higher than the pre-imposed reference value, the optimisation module looked at the history generated using the Bayesian optimisation approach and picked all the combinations of parameters that produced an S_a that was below the threshold. Among those, the ones fed to the machine were taken evaluating the V_f of each. The combination with the highest feed rate was the one that guaranteed the required S_a with the highest material removal rate and was chosen to machine the next piece. Once the choosing phase was over the system automatically modified the G-Code of the next machining operation and uploaded it on the web interface and the new sample could be machined.

3 | Results & Discussion

3.1. CNN Training

In this section, the training of each model is discussed. Performances and training analysis are formulated and explanations about the errors' trends are given. It may look like a repetitive section for each model but it is not because each trained network behaved differently and had its own peculiarities.

3.1.1. Xception using MSE loss function

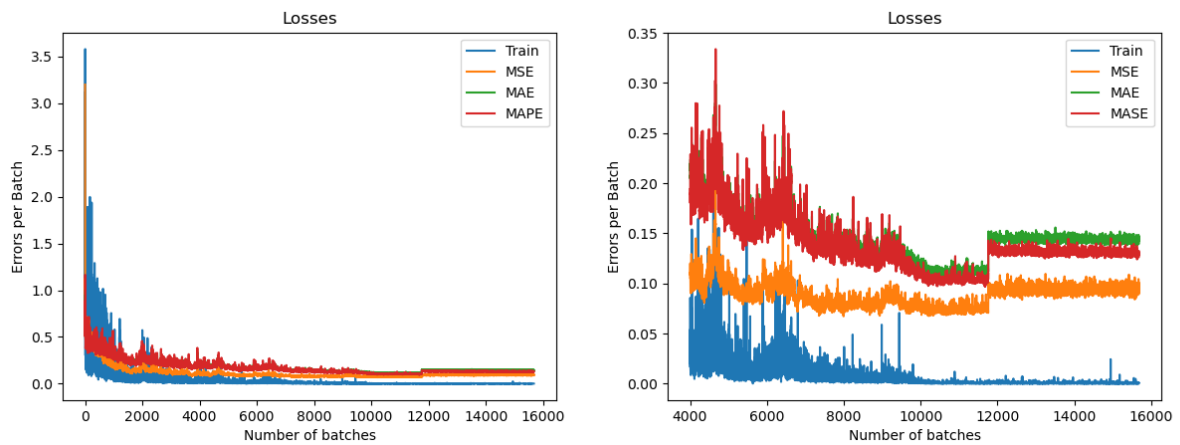


Figure 3.1: Training and validation errors of the Xception model trained using the *MSE* loss function. The right image represents a zoom of the left one.

As it can be noticed from Figure 3.1, the training process progressed smoothly but overfitting was verified. This can be noticed because the validation errors suddenly increase while the training error becomes saturated. This is also justified by test error evolution shown in Figure 3.2.

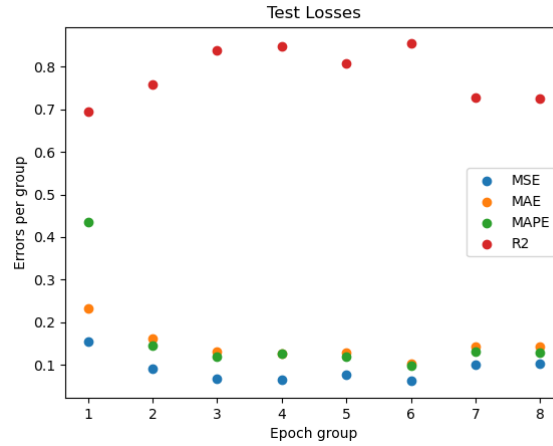


Figure 3.2: Evolution of test errors (Xception model trained using the MSE loss function). Each epoch group considers the model after 8 epochs of training.

Indeed, after the sixth group of epochs, the error increased while the R^2 started to decrease leading to a poorer estimation performance of the network.

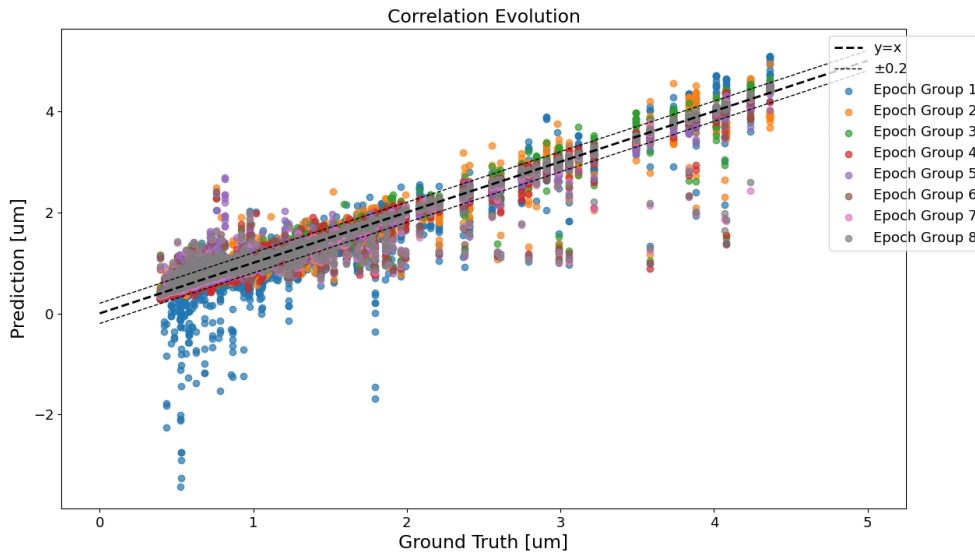


Figure 3.3: Correlation evolution between the ground truth and the predictions per epoch group (Xception model trained using the MSE loss function).

Looking at Figure 3.3 it can be noticed how the training improved the estimation performances of the network epoch by epoch. Initially, the model was characterised by big errors and a big number of outliers. As the training advanced the model got better and better at predicting the correct values from the given pictures. In the end, only a little number of outliers are present while most of the predictions' AE were below $0.1 \mu\text{m}$ (Figure 3.4).

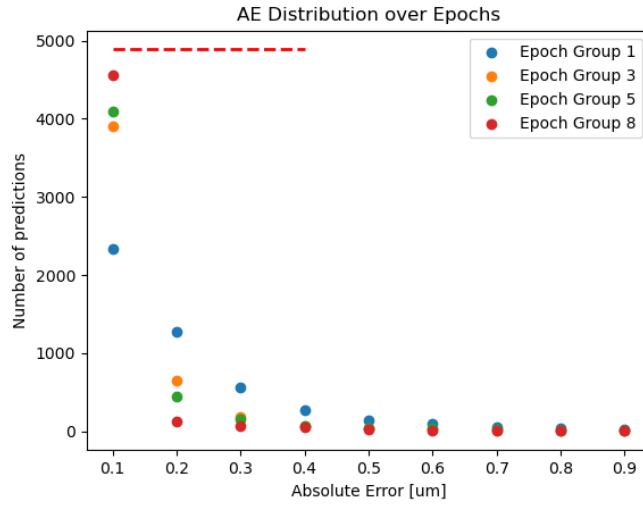


Figure 3.4: *AE* distribution calculated over the whole dataset (Xception model trained using the *MSE* loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.

What is interesting is that initially, the model improved hugely but as the number of epochs increased, only minor improvements were achieved. This was related to the fact that the network saturated rapidly and also the learning rate reduced as further epochs were executed, performing a fine-tuning training instead of a massive one.

3.1.2. Xception using MAPE loss function

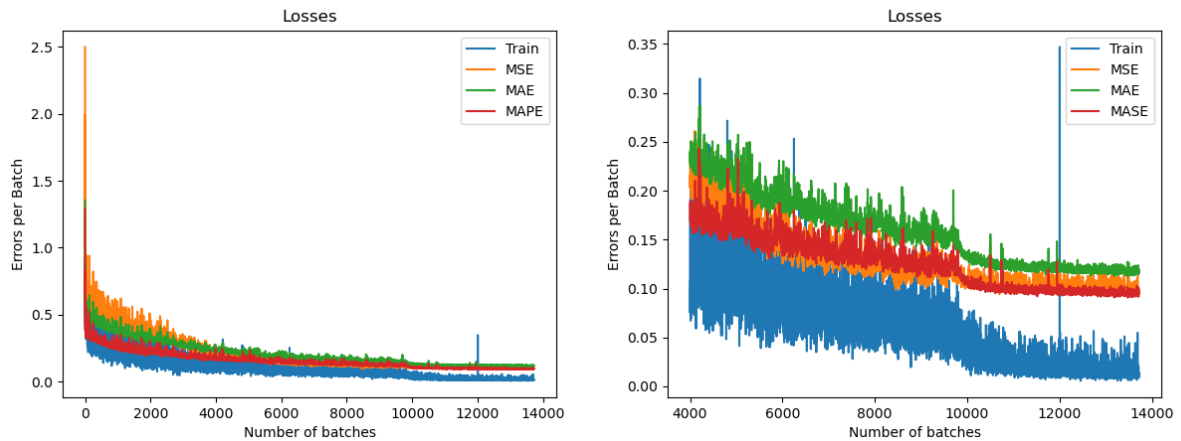


Figure 3.5: Training and validation errors of the Xception model trained using the *MAPE* loss function. The right image represents a zoom of the left one.

In this case, the training did not produce any overfitting. As it can be noticed from Figure 3.5 both the validation and training errors keep following the decreasing trend even during the last epochs. The loss function used in this training session is less sensitive to outliers but is less penalising for big mistakes compared to a squared difference.

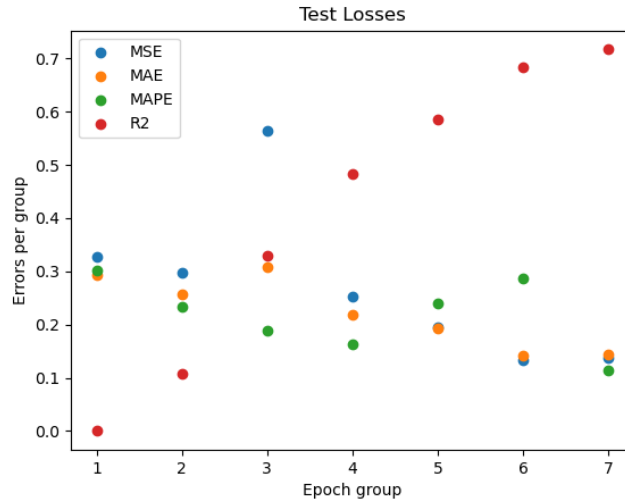


Figure 3.6: Evolution of test errors (Xception model trained using the *MAPE* loss function). Each epoch group considers the model after 8 epochs of training. The R^2 is zero in the first epoch because it resulted in a negative value.

As it can be noticed the improvement rate is slower compared to the previously used loss function with a really poor performance at the initial stage of training.

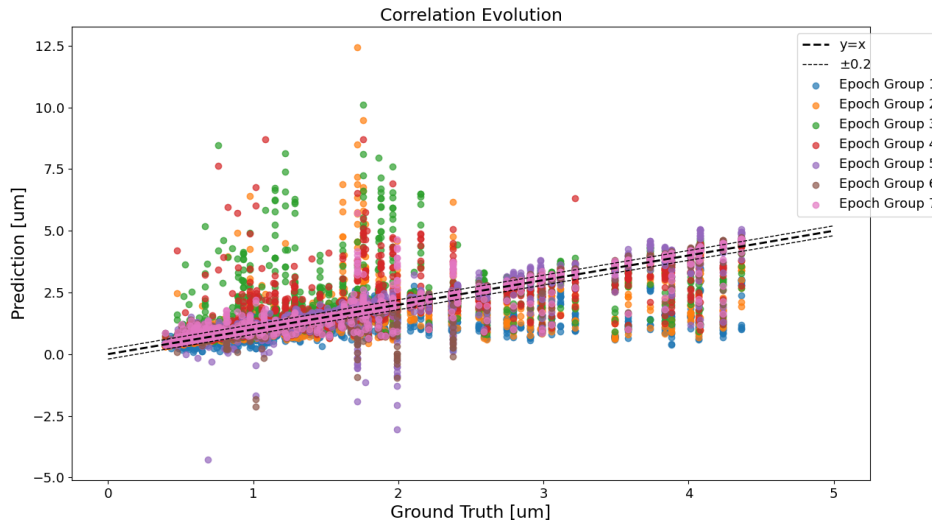


Figure 3.7: Correlation evolution between the ground truth and the predictions per epoch group (Xception model trained using the *MAPE* loss function).

Looking at Figure 3.7 it can be noticed how the training improved the estimation performances of the network epoch by epoch. Initially, the model was characterised by big errors and a big number of outliers. As the training advanced the model got better and better at predicting the correct values from the given pictures. In this case, however, still, a big number of outliers are present even after the many executed epochs and the model has a higher overestimation error

over the outputs than the underestimation one.

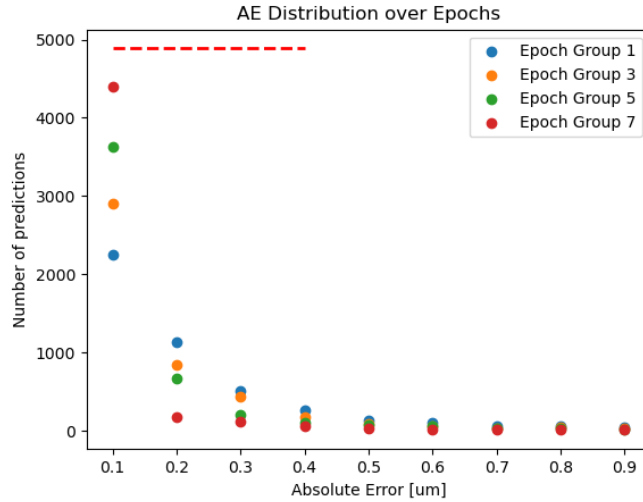


Figure 3.8: *AE* distribution calculated over the whole dataset (Xception model trained using the *MAPE* loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.

3.1.3. Xception using Huber loss function

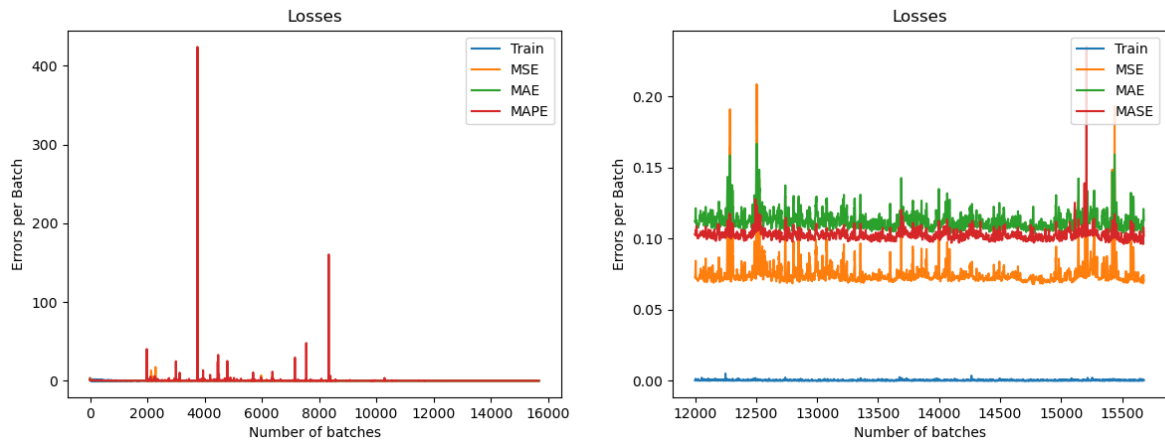


Figure 3.9: Training and validation errors of the Xception model trained using the *Huber* loss function. The right image represents a zoom of the left one.

During this training, the validation errors had some huge peaks related to some batches of images. What is interesting is that these disappear at the end of the training as it can be seen from Figure 3.9. In the final training session, the network was saturated because no improvement in both the training error and the validation was measured. The training error instead, followed a pretty standard flow and was able to decrease rapidly as the number of epochs increased as it can be seen from Figure 3.10. In Figure 3.9 this cannot be noticed because the peaks of the validation losses are too high even when compared to the initial training errors.

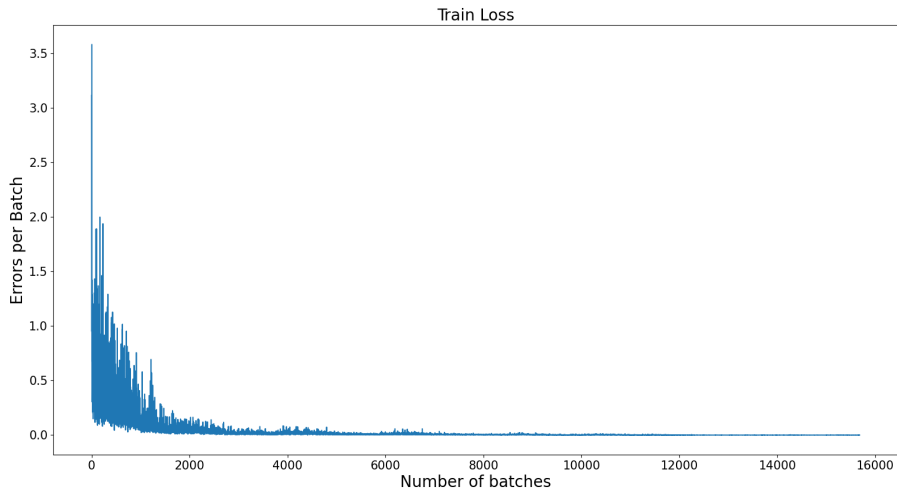


Figure 3.10: Training error evolution per iteration (Xception model trained using *Huber* loss function).

Even when analysing the test losses evolution a slight saturation at the end can be noticed, in which the errors do not improve considerably compared to the previous epochs. Also is interesting to notice that the *MAPE* and the *MAE* are very similar in value as the model improves (Figure 3.11).

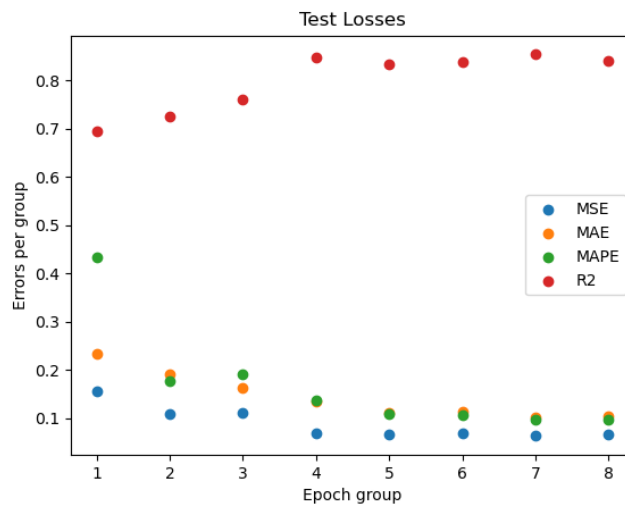


Figure 3.11: Evolution of test errors (Xception model trained using the *Huber* loss function). Each epoch group considers the model after 8 epochs of training.

The correlation plot shows the asymmetrical nature of the loss function as the model tends to commit higher underestimation errors than overestimation ones. Even after all the training epochs still, the biggest outliers are present as underestimations.

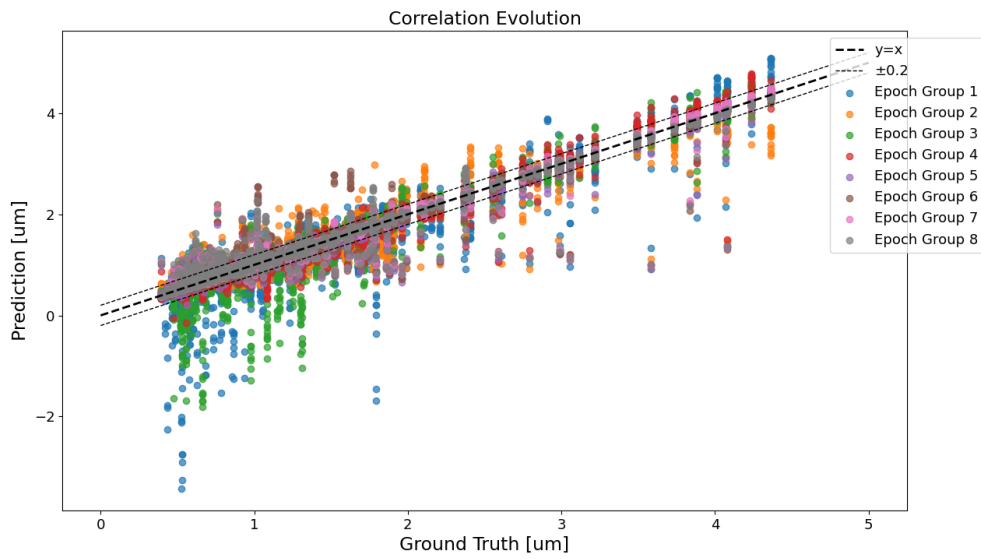


Figure 3.12: Correlation evolution between the ground truth and the predictions per epoch group (Xception model trained using the *Huber* loss function).

Even the *AE* evolution plot in Figure 3.13 shows the same trend as the others with an improving behaviour as the number of epochs increases. The error saturation can be noticed because the advancement after the fifth epoch is not as sharp as before.

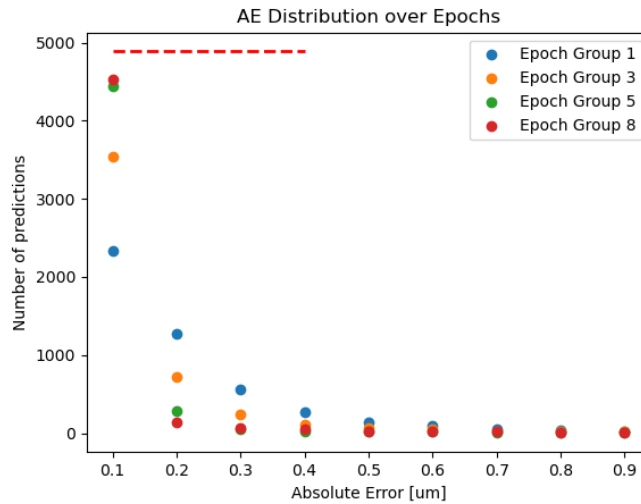


Figure 3.13: *AE* distribution calculated over the whole dataset (Xception model trained using the *Huber* loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.

3.1.4. ResNet50 using MSE loss function

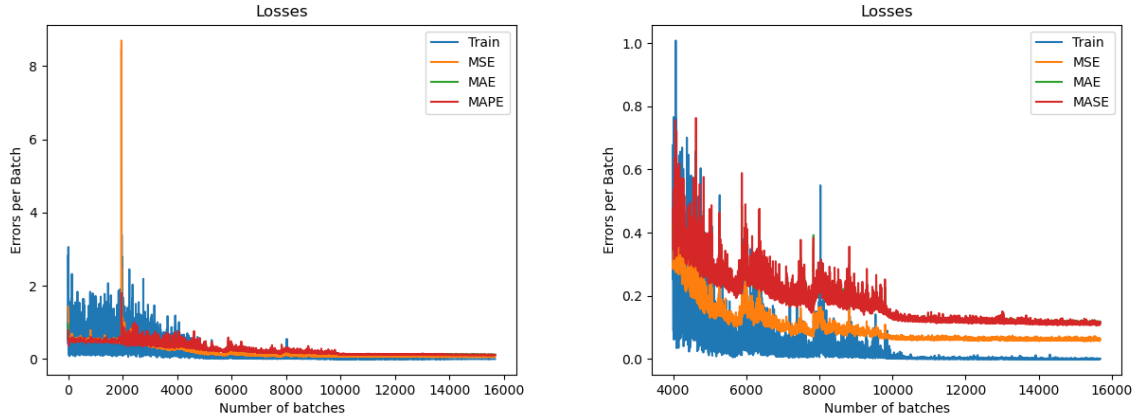


Figure 3.14: Training and validation errors of the ResNet50 model trained using the *MSE* loss function. The right image represents a focus on a particular range of iterations of the left one.

In this case, the training did not produce any overfitting. As it can be noticed from Figure 3.14 both the validation and training errors keep following the decreasing trend but saturation occurs during the last thousands of iterations. It is interesting to see how the losses' trend bounces two times before stabilising and saturating in the end. These bounces can be attributed to particular batches, in which the inputs were still 'strange to the trained' model, generated by the imbalances of the training dataset.

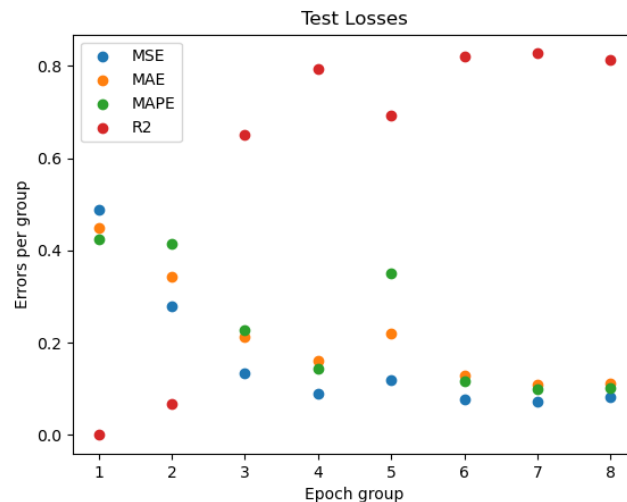


Figure 3.15: Evolution of test errors (ResNet50 model trained using the *MSE* loss function). Each epoch group considers the model after 8 epochs of training. The R^2 is zero in first epoch because it resulted in a negative value.

During the initial stage of training, the model performed really poorly and a little overfitting can be noticed, in Figure 3.15 at the end of the last epoch.

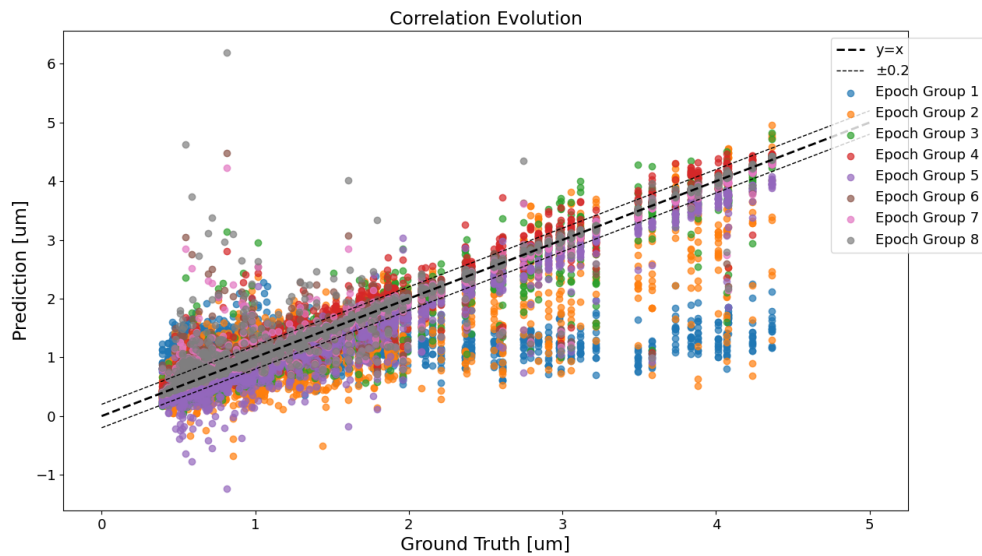


Figure 3.16: Correlation evolution between the ground truth and the predictions per epoch group (ResNet50 model trained using the *MSE* loss function).

Looking at Figure 3.16 it can be noticed how the model struggles to get rid of the big outliers. These, indeed, are still present even after all the computed epochs of training and get bigger when the last one is completed, reinforcing the hypothesis of a possible intimation of overfitting.

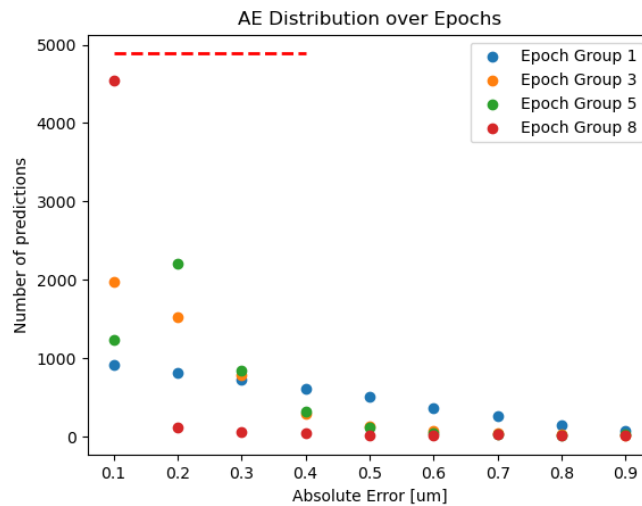


Figure 3.17: *AE* distribution calculated over the whole dataset (ResNet50 model trained using the *MSE* loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.

It is interesting to see that, unlikely to the Xception, this network did not have a gradual improvement during the training. As it can be noticed from Figure 3.17, there is a big jump in the error distribution after the fifth epoch which also corresponds to the decrease in the learning rate.

3.1.5. ResNet50 using MAPE loss function

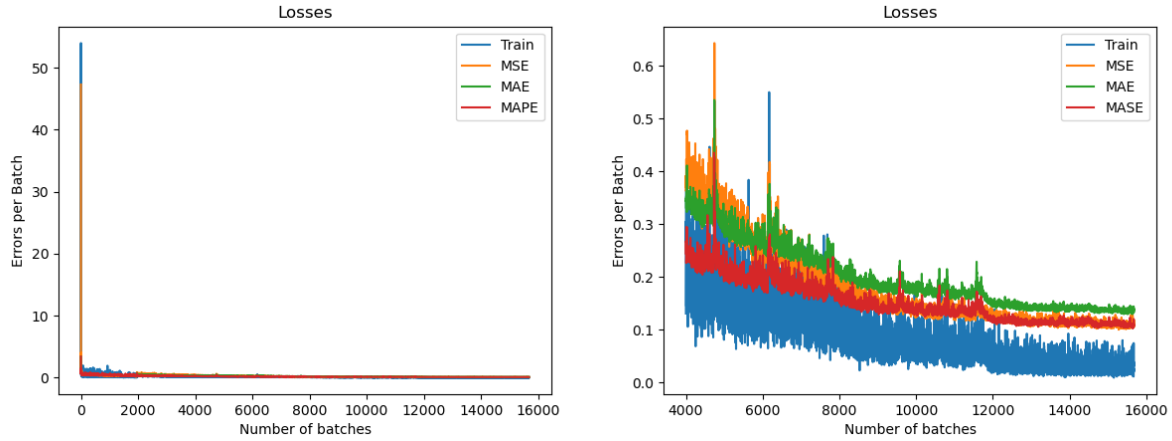


Figure 3.18: Training and validation errors of the ResNet50 model trained using the *MAPE* loss function. The right image represents a zoom of the left one.

Because the weights are randomly initialised the starting training and validation errors are big compared to the rest. However, in the last stages, the trend is a decreasing one and the reduction in the learning rate helps to lower them even more. Even here the test error analysis shows little sign of overfitting after the last epoch.

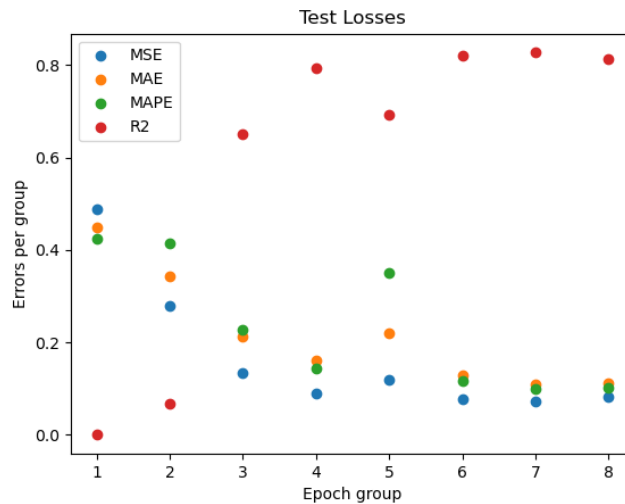


Figure 3.19: Evolution of test errors (ResNet50 model trained using the *MAPE* loss function). Each epoch group considers the model after 8 epochs of training. The R^2 is zero in first epoch because it resulted in a negative value.

As it can be noticed, the model performs badly during the initial training stages and a little sign of overfitting can be noticed at the end of the last epoch as the test losses increase (Figure 3.19).

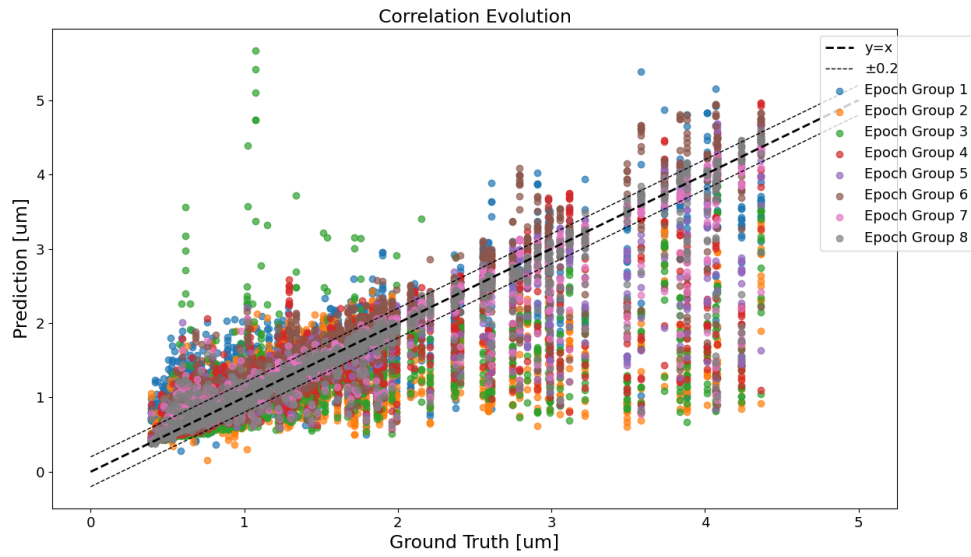


Figure 3.20: Correlation evolution between the ground truth and the predictions per epoch group (ResNet50 model trained using the *MAPE* loss function).

The correlation evolution plot shows that the training process was able to significantly improve overestimation outliers but not the underestimation ones that are heavily present even in the last stages of the training process. The many white spaces that can be noticed in Figure 3.20 and all the previous ones of the same type (3.3, 3.7, 3.12, 3.16) represent a lack of data in those areas. This is created due to the imbalance of the training dataset that does not have the same amount of samples per each S_a range, which reasons were justified in Chapter 4.

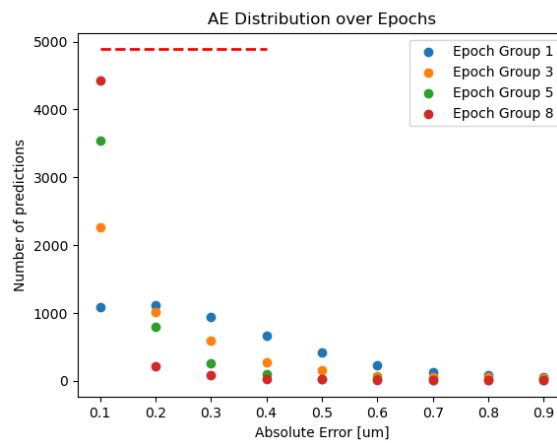


Figure 3.21: *AE* distribution calculated over the whole dataset (ResNet50 model trained using the *MAPE* loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.

3.2. CNN Models comparison

Once all the models were trained the best had to be chosen. To determine the most suitable one among all of them the different errors were analysed. In particular, both the performance of the models over the whole dataset and the test dataset were considered. As shown in the previous section, some networks did not improve after a certain number of epochs but, instead, they reduced their performances. To consider this, the best model for each training error and architecture was chosen based on the evolution of the error among the test and total dataset. If the same model performed better on the total set after a certain number of training iterations but had a better accuracy on test one after a different number of training steps, priority was given to the latter. This is because the chosen one will be implemented to monitor the machine in a new environment and maybe will have to make predictions based on inputs never seen before. Moreover, if the priority was given to the best performance over the whole dataset and, considering that eighty percent of this was used for training, choosing that network could result in using a model with overfitting problems. So the best ones were firstly selected among each training session with different loss functions, resulting in a total of five networks, and after, the best one was picked among them evaluating the same metrics used before.

<i>Model</i>		<i>CompleteDataset</i>				<i>TestDataset</i>			
Architecture	Loss	MSE	MAE	MAPE	R ²	MSE	MAE	MAPE	R ²
Xception	MSE	0.015	0.038	0.036	0.97	0.062	0.102	0.097	0.854
Xception	MAPE	0.049	0.057	0.038	0.912	0.138	0.143	0.114	0.717
Xception	Huber	0.016	0.042	0.037	0.969	0.063	0.102	0.096	0.854
ResNet50	MSE	0.022	0.044	0.038	0.956	0.072	0.109	0.1	0.827
ResNet50	MAPE	0.069	0.12	0.089	0.872	0.107	0.164	0.145	0.77

Table 3.1: Best models errors and metric calculated using the complete dataset and the test one.

3.2.1. Xception models comparison

The Xception models performed better than the ResNet50 ones both on the test and the global dataset.

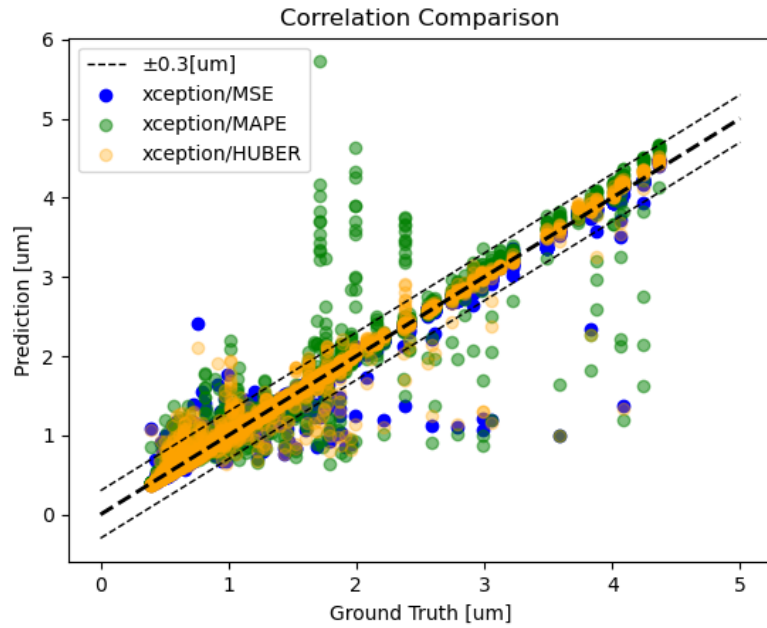


Figure 3.22: Correlations comparisons of the three Xception models over the whole dataset.

As it can be seen from Figure 3.22 the model trained with the *MAPE* as loss function performed worse.

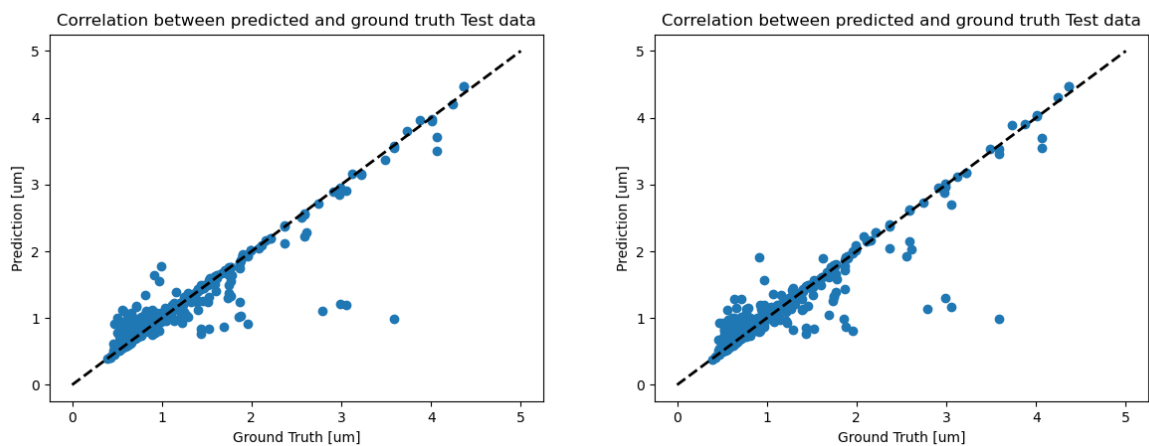


Figure 3.23: On the left are represented the results of the correlation over the test data of the model trained with the *MSE* loss function. On the right the results of the Xception model trained with the *Huber* loss function.

In particular, it cannot be used in a real machining application because it has many outliers both in the overestimation region and in the underestimation one. The other two instead, are

quite similar also in terms of errors. Even the test analysis confirms that the differences between the two models are very small.

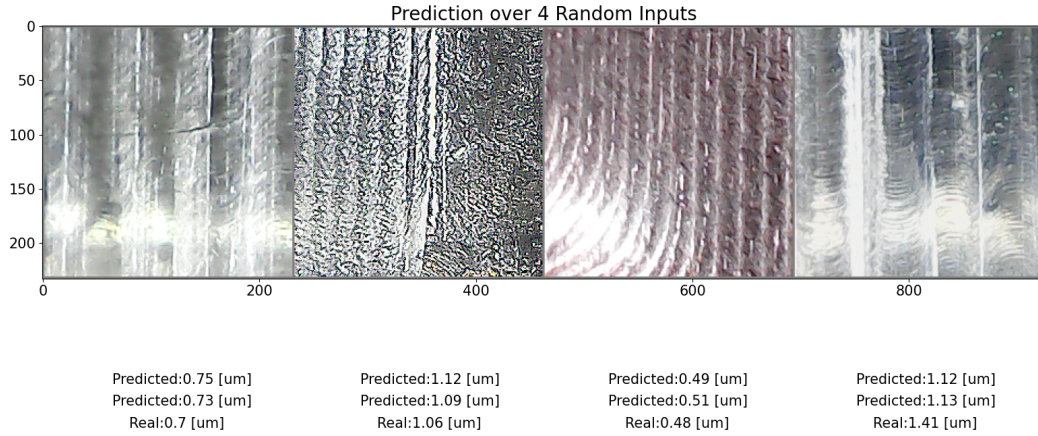


Figure 3.24: The first prediction is made using the model trained with the *MSE* loss function. The second, using the model trained with the *Huber* loss function.

Figure 3.24 was created to show the reader that the model is able to make predictions using real pictures of machined surfaces. As it can be seen, some of them are blurred or with different brightnesses because they were randomly chosen from the test dataset which was formed also from pictures that have been generated during the data enhancement process. In addition, it can be noticed that different parameters have been used to create the samples; especially the radial depth of cut was different among the different pictures as the distance between each pattern varies among them. This example is not sufficient to quantify the accuracy of the trained models but shows how predictions can be made using pictures of real machined surfaces.

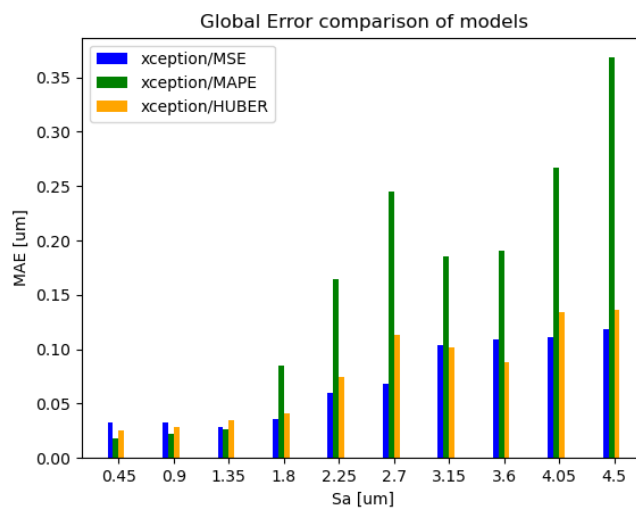


Figure 3.25: *MAE* calculated over the total available samples divided by S_a ranges.

Finally, a study over the distribution of the MAE was done to analyse the performances of the models in different areas of the available input space. Surprisingly, the worst model performs

better than the others until $1.35 \mu\text{m}$ but then loses its advantage and becomes the worst, by far. All the previously shown plots are crucial to evaluate and choosing the best model because only one metric or just the numerical values are not enough to conduct a correct evaluation.

3.2.2. ResNet50 models comparison

As stated before the ResNet50 models underperformed compared to the Xception ones. As it can be seen from Figure 3.26 both models contain more outliers and a bigger dispersion of the data around the centre diagonal line compared to the previous ones. In addition, they both have higher overestimation and underestimation. This is reflected in Table 3.1 with higher values of the errors and a lower R^2 .

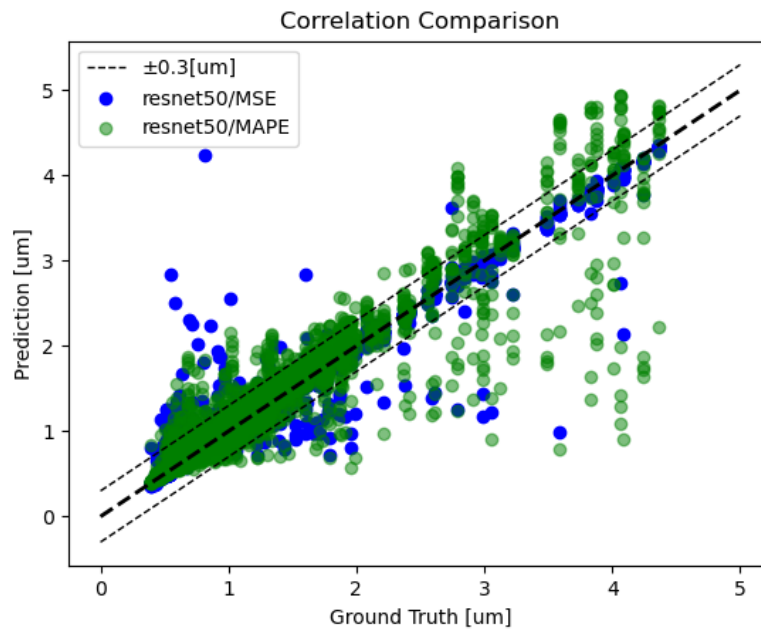


Figure 3.26: Correlations comparisons of the two ResNet50 models over the whole dataset.

Surprisingly, the test results looked very similar to the Xception models ones. This shows how this network possesses high adaptability and flexibility due to its simple and less deep architecture.

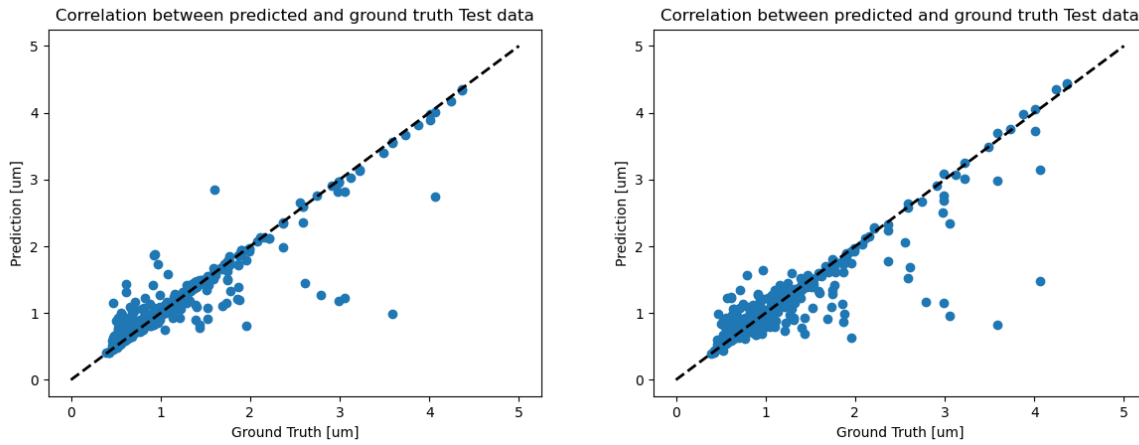


Figure 3.27: On the left are shown the results of the correlation over the test data of the model trained with the *MSE* loss function. On the right the results of the model trained with the *MAPE* loss function.

Even these networks can be used to make predictions based on pictures extracted from real machined surfaces. Care must be put into using the correct pre-processing transformations because they are different from those used for the Xception models, as explained in Chapter 2.

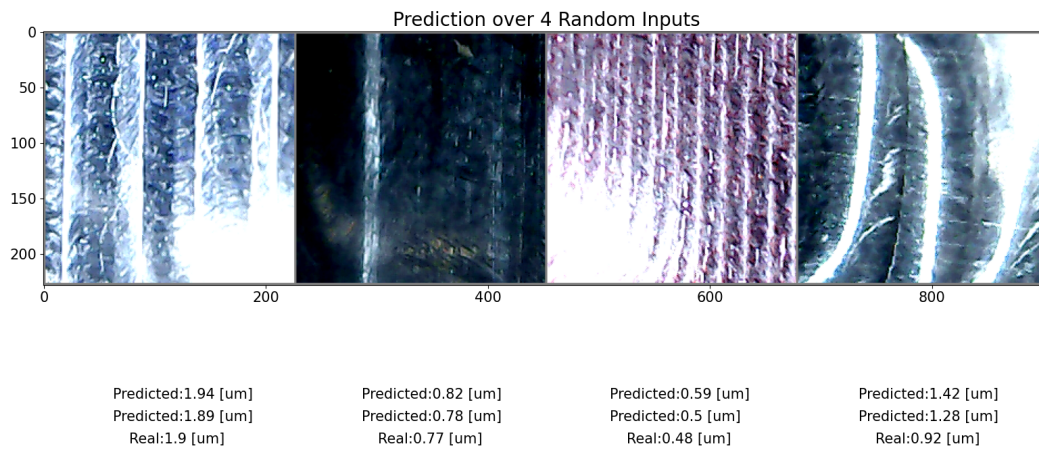


Figure 3.28: The first prediction is made using the model trained with the *MSE* loss function. The second, using the model trained with the *MAPE* loss function.

Finally, the study over the distribution of the *MAE* by S_a ranges could be made among all the five models together. As it can be seen from Figure 3.29 for low values of Roughnesses the difference between the models is not as big as for the higher values of S_a . This is mainly due to the distribution of the samples inside the training dataset which, as explained before, was not uniform but presents many more samples in the low S_a range. However, the Xception models were able to tackle this problem by relying on a more advanced and upgraded architecture that was able to lower the prediction errors even in areas with a scarce amount of training data.

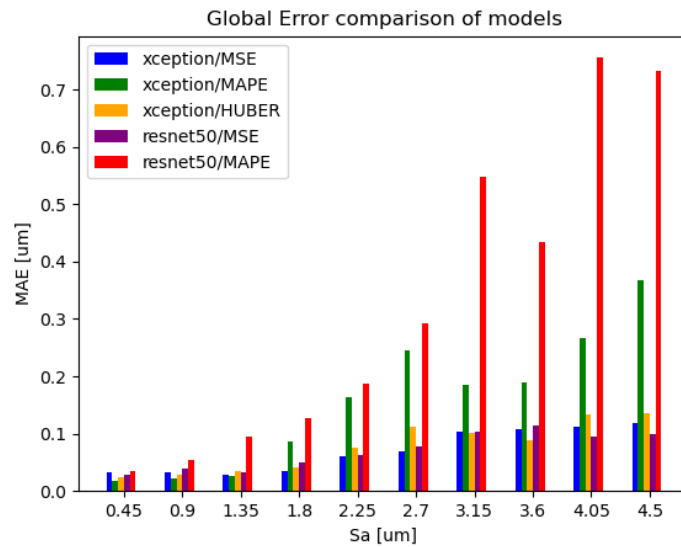


Figure 3.29: *MAE* calculated over the total available samples divided by S_a ranges.

3.2.3. Final model test performances

After having analysed all the single models' performances over the complete dataset and the test one, only one was chosen to be tested over 52 new samples that were machined using new parameters. This was done to see its adaptability and performance over a totally new dataset that is composed of new pictures that are not enhanced but are directly taken on the machine. Even in this application both the full brightness and half brightness conditions were tried while taking the new images. To increase the prediction accuracy the final estimation was made after averaging all the outputs of the network fed with the real image and additional ones generated by enhancing it with the different six different techniques used to create the dataset. This also reduced the probability of creating outliers.

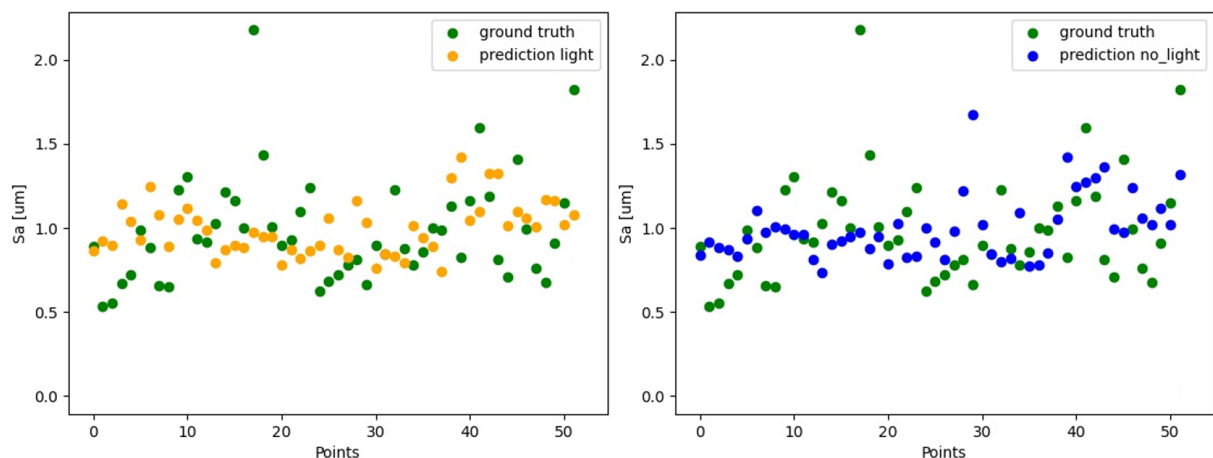


Figure 3.30: On the left are represented the results of the estimations when the input are picture taken using the full brightness of the camera's LED. On the right, when half brightness is used.

It is hard to specify if there was a clear difference between the case of full brightness or half. Even in Figure 3.31 it can be seen that the difference between the two cases is minimal.

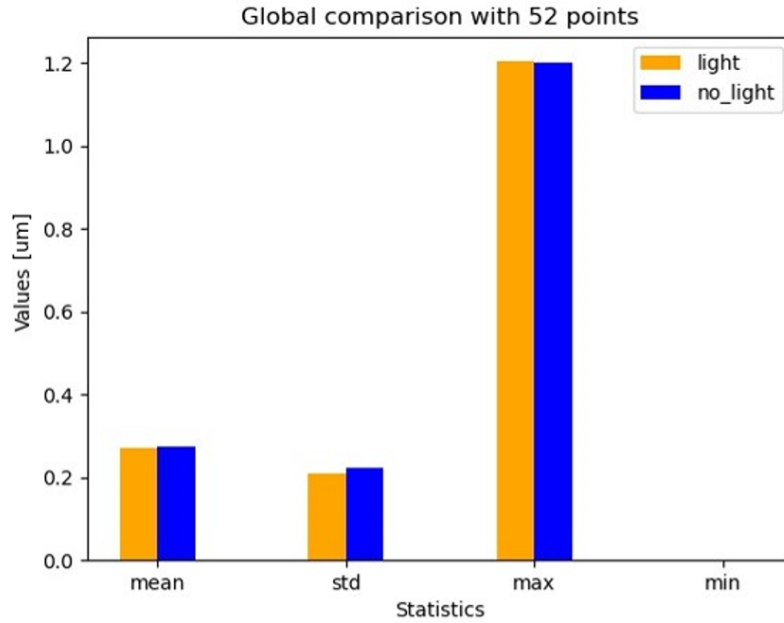


Figure 3.31: Analysis of the AE between the estimations performed over the new 52 machine samples.

In Figure 3.31 it can be seen that the MAE is below $0.3 \mu m$ but the maximum measured AE is above $1 \mu m$. In addition, the minimums are not $0 \mu m$ but, because the results were rounded to the second significant digit, they resulted in $0 \mu m$. Clearly, it can be noticed that the model was not outliers-free but instead at least two huge ones can be observed (Figure 3.30). One interesting notice is that for lower values or S_a the model performed slightly better when the light was fully turned on while for higher values when it was in half brightness mode. This can be reasonably explained because when the roughness is lower, more details are present and the tool marks are usually closer to each other so a higher illumination helps in better identification of the features. When instead the surface is rougher too much can cause unwanted reflections that can spoil the real picture confusing the network during the estimation process.

3.3. Static Optimisation

The trochoidal slotting operation on aluminium was optimised by applying what was introduced and explained in Chapter 2. The *Matern 5/2* kernel was used because of its better sensibility to the distance between two near points compared to the *Exponential* one.

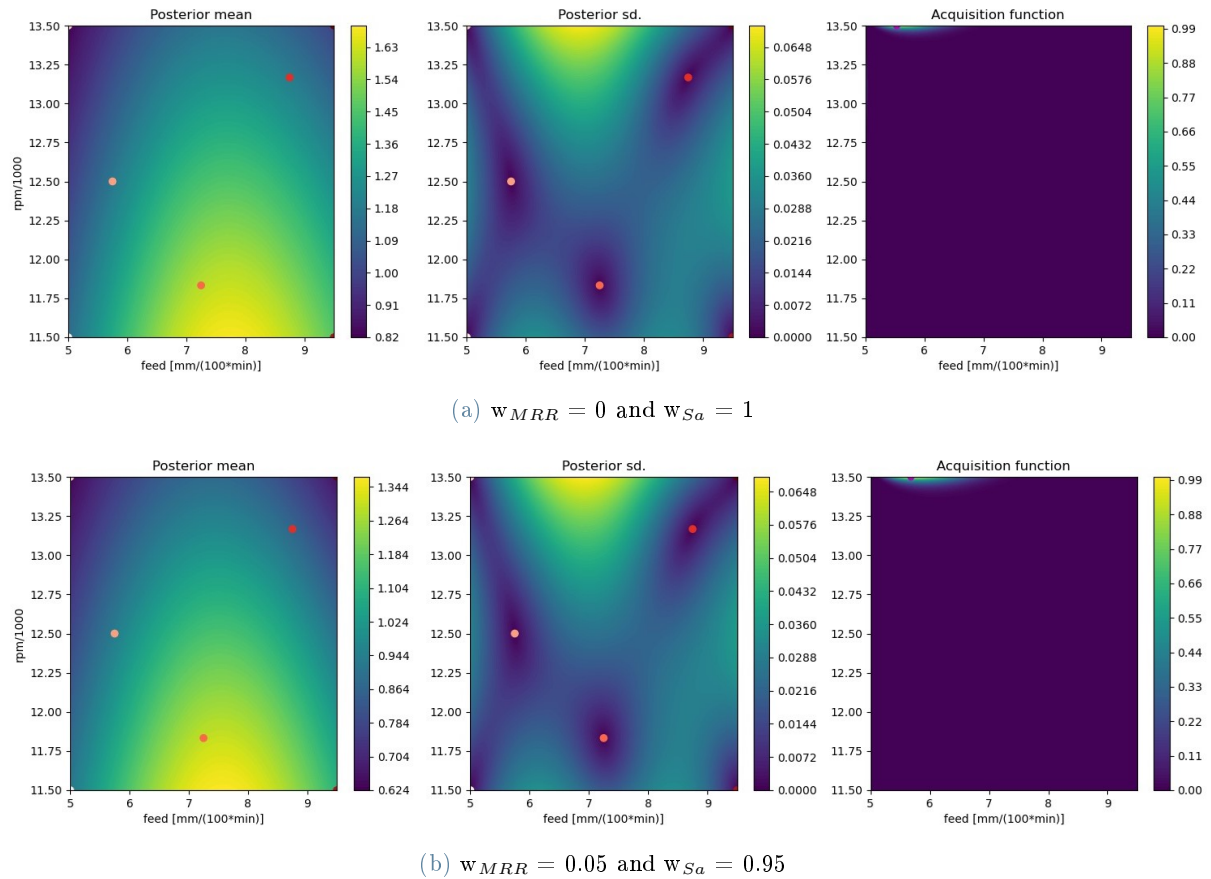


Figure 3.32: Posterior mean, Posterior standard deviation and Acquisition function of the different models created using different weights. They are named "Posterior" because computed after the GPR model was fitted on the available data-points and the BO process was concluded. The red points present on the first two heat maps represent the data-points used to fit the model. The one on the last picture to the right instead, represents the newly suggested one from the BO process.

A little increase in the weights does not sharply influence the decision of the optimiser which decides to stay in the neighbourhood of the lowest *feed rate* but pushes for higher *rpm* which proved to be beneficial to the roughness and do not affect the *MRR*. Among all the reasons to explain why a higher speed of rotation produces a better surface quality, considering the machine and tools, a reasonable one could be that when the machining parameter is higher the tool is in contact with the piece more frequently generating a smoother transition between each tooth as they bite into the material and, considering that the tool used only have two flutes, this generates a more stable process.

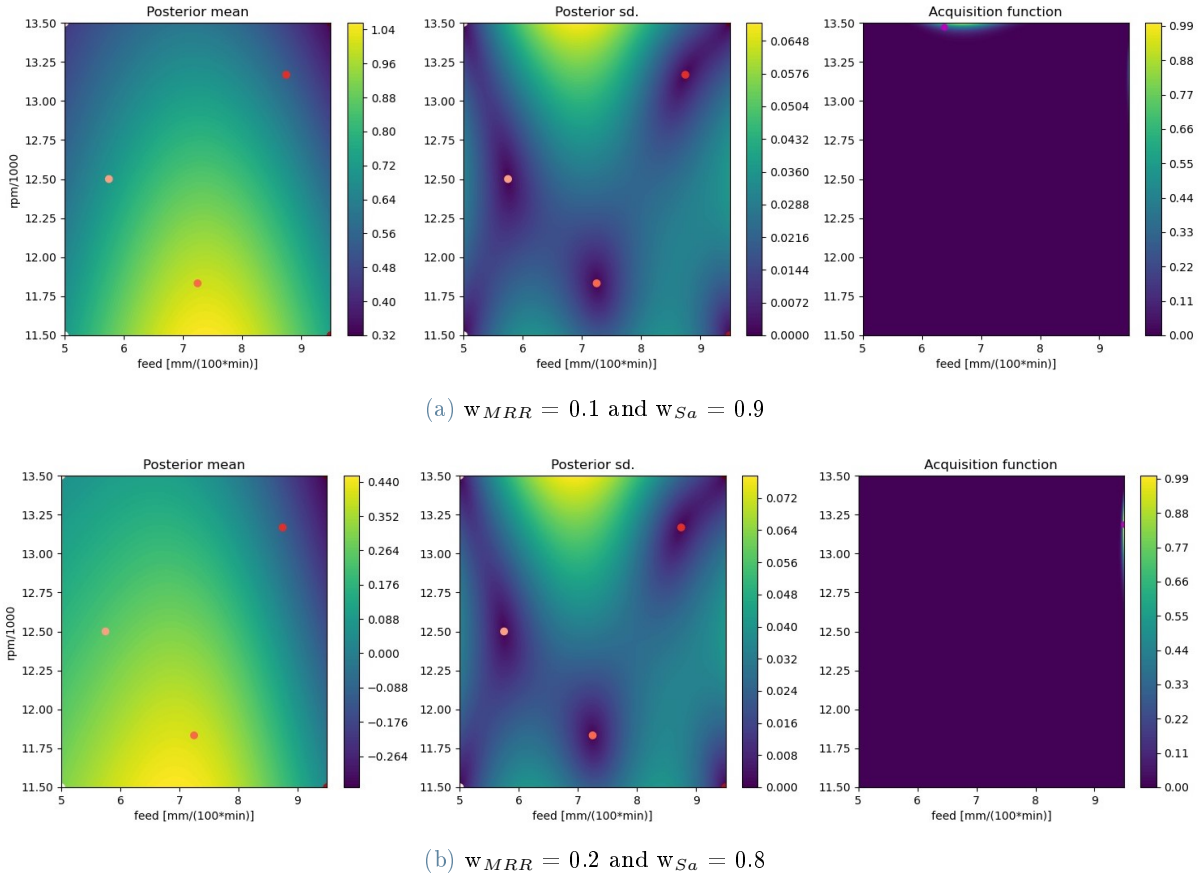


Figure 3.33: Posterior mean, Posterior standard deviation and Acquisition function of the different models created using different weights. They are named "Posterior" because computed after the GPR model was fitted on the available data-points and the BO process was concluded. The red points present on the first two heat maps represent the data-points used to fit the model. The one on the last picture to the right instead, represents the newly suggested one from the BO process.

In Figure 3.33 it can be seen how a bigger set of weights is able to 'convince' the optimiser to look for spaces in which there is a higher MRR to the detriment of the S_a . The optimizer in fact, suggests a point that is pushed to the far right of the last heat map portrayed in Figure 3.33, where the maximum MRR can be obtained. It is interesting to see that a fifty-fifty balance between the MRR and S_a weights are not needed to push the optimiser to suggest the highest $feed\ rate$ possible, but instead, this result is already achieved with a twenty to eighty per cent one. The optimiser is also very sensitive to small changes in the ratio of the weights as it can be noticed by comparing the (a) and (b) suggested points figured in the far right heat maps present in Figure 3.33.

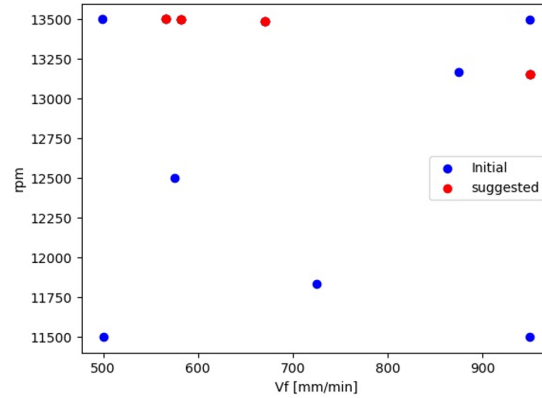


Figure 3.34: Suggested points by the optimiser after the first iteration is completed.

As the weights increase the optimiser reduced the *rpm* and increases the *feed rate* to push toward higher values of *MRR*.

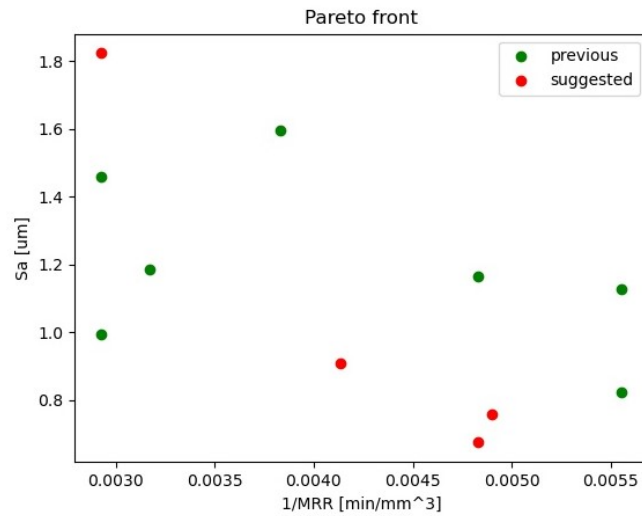


Figure 3.35: Correlation between S_a and MRR that generates a Pareto with some of the achieved results, after the first iteration of the optimiser.

The optimiser was able to find new points that lay on a hypothetical Pareto front. Indeed as can be noticed from Fig 3.35 the new points generated with lower *feed rate* produced samples with the lowest S_a measured. This can be attributed to the fact that a too low V_f generates a *feed per tooth* that is not enough to correctly remove the chips from the surface because they are compressed and stacked instead of cut. The optimization procedure was stopped after just one iteration and four new suggested points because the optimiser was not able to propose any other new parameter combinations.

3.4. Combined system



Figure 3.36: The final assembled system. There can be noticed the python script, the CNC controller web page, and the machine with the installed monitoring system.

The assembled system was able to successfully identify the machined surface roughness and, because this was above the set threshold, change the feed rate and speed to obtain a better surface quality.

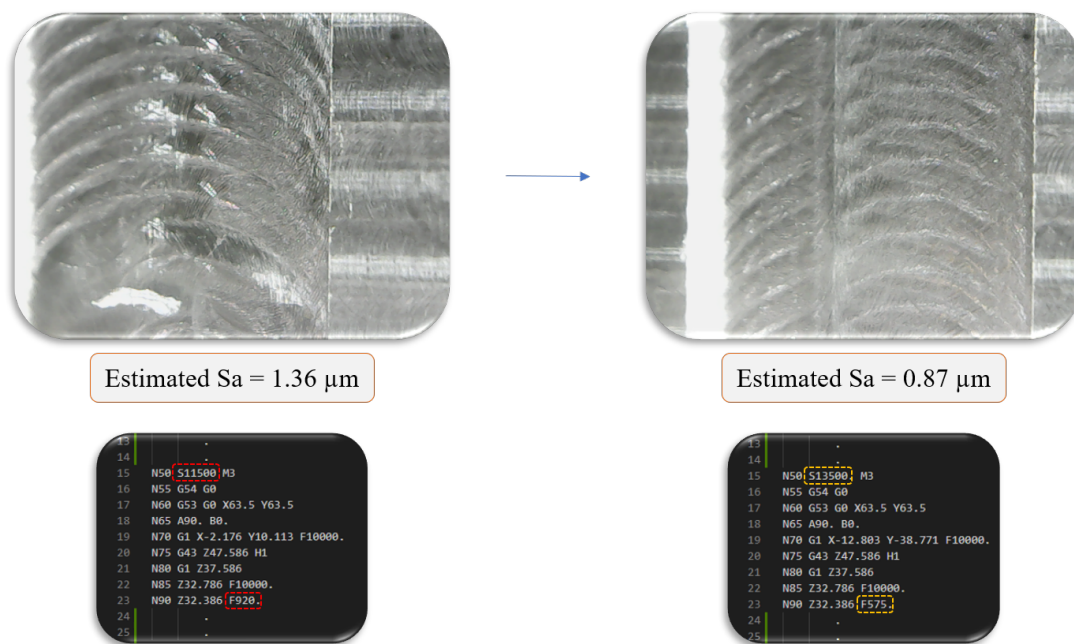


Figure 3.37: On the left is portrayed the machined surface generated following the initial G-Code. On the right the one using the modified one. The correspondent G-Codes are represented below and the machining parameters are highlighted by the dotted boxes.

4 | Closing the Circle

"Only the people who are crazy enough to think they will change the world are the ones who'll do."

- Steve Jobs -

4.1. Summary

In this thesis, a monitoring system capable of identifying the surface roughness of a milled surface using a simple USB Camera was implemented. This was realised by training different architectures of Convolutional Neural Networks and comparing their performances over a fixed dataset and a newly created one. The training of these involved the creation of a big dataset of images realised by machining aluminium and steel billets (mainly the first material) with different parameter combinations to obtain a wide range of roughness values. Image enhancement was also used to obtain even more useful images. The creation of a sample required many resources and time due to the limitations of the used machine which was a small 5-axis CNC mill. Because of that, parametric 3D modelling and CAD to CAM linking were used to speed up the modelling and G-Code generation times. In addition, an optical non-contact measurement system was used to directly extract useful information regarding multiple machined surfaces in a single shot just by taking pictures from above. The created dataset was split between training, validation and testing and, because of the unbalanced distribution of the roughness values, care was put to guarantee that each dataset contained all the values of measured S_a reducing the probability that, during training, each model only saw a part of the obtained roughnesses. Subsequently, Multi-Objective Bayesian Optimization was used to find the best combination of feed and speed for a specifically trochoidal slotting milling operation. Because the aim of the optimization was to obtain the lowest surface roughness with the highest material removal rate possible, the objective function was created using a weighted sum between the two target outputs. Different weights allowed exploring different areas of the allowed inputs space obtaining different parameters combination. Finally, after all the trained models had been compared using different error metrics evaluated over the training and test datasets, the best model was chosen and the monitoring system was combined with the optimisation one. The scope of this operation was to prove the possibility of evolving a normal CNC machine into an autonomous device. The assembling of this system included the linking between the optimisation module, the monitoring system and the CNC. Because the machine was controlled using a web-built interface developed

by the company, the monitoring and optimisation had to be connected to a particular web page. This was achieved using a web scraping script that was able to substitute the human-to-screen interaction. Once completed the system was able to take a G-Code, modify it to let the monitoring system perform its estimation, analyse the estimated data and, if needed, change the targeted machining parameters of the next G-Code using a simple strategy based on the data found using the optimization procedure. The new machining operation could then be started with the new updated parameters and G-Code.

4.2. Conclusions

The realised monitoring system was able to correctly characterise most of the machined surfaces. The errors in the training and test set showed that the architecture was well-trained and effective. In the end, the best model proved to be the *Xception* architecture trained with the *MSE* loss function. Also, the *Huber* loss function proved to be effective but suffered from asymmetry and a higher number of outliers. The final *MAE* measured over the new 52 machined samples proved that the system was accurate enough to predict the various roughnesses. Clearly, each machining sector has its specifications in terms of tolerances and precisions. For example, in a micro-machining application, the accuracy of the network would not be sufficient to consider it feasible to be implemented in such an environment. Because different machining strategies were used the adaptability was improved compared to previous studies in which only a single machining operation with a fixed tool path is chosen. The use of the camera directly mounted on the machine that takes pictures as soon as the machining operations are finished was rarely adopted in the previous studies, which mainly focused on conducting the estimation once the manufacture was taken out of the device. In addition, the machining environment was not specifically designed, with an external concentrated source of illumination or advanced mirrors, to accommodate a machine vision system. This, combined with the use of a cheap USB camera mounted on a 3D printed stand, gave this work a relative novelty compared to most of the previous studies in which the system is realised using very expensive and advanced optics and cameras. Once the model is fully trained it can be run on any modern device and the estimation time is below one second, which makes it suitable to be installed in a typical manufacturing environment. However, the presence of few but huge outliers needs to be improved because it can lead to wrong predictions and as a consequence wrong parameters modification. In this work, their presence was hard to get rid of due to the nature of the manufacturing process which, as stated before generated most samples with a particular range of roughness. In addition, the dimension of the dataset should be raised without applying data enhancement extensively so that the set is composed of truly different samples. The optimisation strategy used obtained a good distribution of feasible parameters with only a single new iteration. In total, only 11 samples were machined to obtain such results and no prediction model was needed. This was the perfect application for this type of operation in which creating a new specimen required time and resources due to the many operations needed between machining and measuring. Even if a

full Pareto curve was not obtained the method proved to be effective and identified three new combinations of parameters that had a higher *MRR* compared to the minimum one but a lower surface roughness. Compared to previous studies this method possesses higher adaptability because only needs a few iterations to achieve satisfactory results and so can be adapted to different machining operations and strategies. The fully assembled system proved to be successful in changing the machining parameters once an anomaly in the S_a of the machined surface was detected by the monitoring system. The optimisation strategy adopted in this case was simple but productive. However, it is not suitable to be applied in a dynamic environment in which the conditions can change and are time-dependent. To be implemented in a real industrial system also the tool wear must be considered when choosing the best combination of parameters, as a consequence, this strategy would not be suitable anymore and the model would have to be uploaded every time. In addition, even if the benefits of the optimisation system can already be seen, in an industrial context the optimisation strategy should target more machining parameters including the depth of cuts, which indirectly influence the final quality of the machined surfaces as higher ones generate higher instabilities, machining forces, vibrations and less heat dissipation, and should also aim to minimise the tool wear. This will require additional samples to be produced, many more iterations and an additional monitoring system that can visually check the state of the tool after each machining operation.

4.3. Future Work

To make this system fully applicable in a real industrial context tool wear will also be considered and the optimisation strategy will include also the minimisation of the final tool wear and will optimise also the depths of cut. This will require the linking between the CAM software and the optimization module because each time the radial depth of cut is changed the tool path changes. In addition, the monitoring system will be improved to remove the outliers and reduce accuracy errors by using state-of-the-art architectures or tailor-made networks. It would be really interesting and useful to create a repository in which companies could upload samples, data and could download tailor-made datasets to train their models for specific tasks. This will increase the accuracy of the models and the training speed because it will save the samples creation process that has to be done each time datasets are not available. In addition, the adaptability could be further raised using meta-learning [64], [65] which, when a new material must be analysed, can derive a new set of weights of the CNN instead of randomly initialising them during the training. Considering the tool wear while uploading the machining parameters directly on the machine will require the implementation of Bayesian optimisation also for this task. In this case, because the environment is time-dependent, each time the monitoring system detects the anomaly in the roughness the model would be uploaded and the optimiser can suggest a new set of machining conditions that guarantee a surface roughness in the constrained limits, a maximum material removal rate and minimal final tool wear.

Bibliography

- [1] M.Eugen Merchant. Mechanics of the Metal Cutting Process. I. Orthogonal Cutting and a Type 2 Chip, *Journal of Applied Physics*, Vol. 16, Number 5, 1945.
- [2] Luca Pagani, Paolo Parenti, Salvatore Cataldo, and Massimiliano Annoni. Indirect cutting tool wear classification using deep learning and chip colour analysis, *The International Journal of Advanced Manufacturing Technology*, Vol. 111, pp. 1099–1114, 2020.
- [3] Matthieu Rauch, Emmanuel Duc, and Jean-Yves Hascoet. Improving trochoidal tool paths generation and implementation using process constraints modelling, *International Journal of Machine & Tools Manufacture*, Vol. 49 , pp. 375–383, 2009.
- [4] Abram Pleta, Farbod Akhavan Niaki, Laine Mears. A comparative study on the cutting force coefficient identification between trochoidal and slot milling, *46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA; Procedia Manufacturing 26* 2018.
- [5] <https://grabcad.com/library/>.
- [6] Viktor Astakhov, J. Paulo Davim. Tools (Geometry and Material) and Tool Wear, 2008.
- [7] Tool life testing in milling — Part 2: End milling, ISO 8688-2:1989.
- [8] Baofeng He, Siyuan Ding, Zhaoyao Shi. A COMPARISON BETWEEN PROFILE AND AREAL SURFACE ROUGHNESS PARAMETERS *METROLOGY AND MEASUREMENT SYSTEMS*, Vol. 28 No. 3, pp. 413–438, 2021.
- [9] Geometrical Product Specifications (GPS) — Surface texture: Profile method — Terms, definitions and surface texture parameters, ISO 4287:1997.
- [10] <https://www.rapidirect.com/blog/cnc-history>.
- [11] <https://www.youtube.com/watch?v=MIjWkEVNZZU>.
- [12] Santonab Chakraborty, and Shankar Chakraborty, A Scoping Review on the Applications of MCDM Techniques for Parametric Optimization of Machining Processes, *Archives of Computational Methods in Engineering*, Vol. 29, pp. 4165-4186, (2022).
- [13] Amit R. Patel, Kashyap K. Ramaiya, Chandrakant V. Bhatia, Hetalkumar N. Shah, and Sanket N. Bhavsar, Artificial Intelligence: Prospect in Mechanical Engineering Field—A

- Review, *Data Science and Intelligent Applications. Lecture Notes on Data Engineering and Communications Technologies, Vol. 52, Springer, (2021).*
- [14] Y.D. Chethana, H.V. Ravindrac, Y.T. Krishnegowda. Optimization of machining parameters in turning Nimonic-75 using machine vision and acoustic emission signals by Taguchi technique, *Measurement*, Vol. 144, pp. 144–154, 2019.
- [15] T. Y. Wu, K. W. Lei. Prediction of surface roughness in milling process using vibration signal analysis and artificial neural network, *The International Journal of Advanced Manufacturing Technology*, pp. 102:305–314, 2019.
- [16] S. Palani, U. Natarajan. Prediction of surface roughness in CNC end milling by machine vision system using artificial neural network based on 2D Fourier transform, *The International Journal of Advanced Manufacturing Technology*, pp. 54:1033–1042, 2011.
- [17] Achmad P. Rifai, Hideki Aoyama, Nguyen Huu Tho, Siti Zawiah Md Dawal, Nur Aini Masruroh. Evaluation of turned and milled surfaces roughness using convolutional neural network, *Measurement*, Vol. 161, 2020.
- [18] Yonglun Chen, Huaian Yi, Chen Liao, Peng Huang, Qiuchang Chen. Visual measurement of milling surface roughness based on Xception model with convolutional neural network, *Measurement*, Vol. 186, 2021.
- [19] Huaian Yi, Yonglun Chen, Lingli Lu. A Visual Classification Method for Milling Surface Roughness Based on Convolutional Neural Network, *Measurement*, Vol. 161, 2020.
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition, *arXiv:1512.03385v1*, 2015.
- [21] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, *arXiv:1610.02357*, 2017.
- [22] Gao Huang, Zhuang Liu, Laurens van der Maaten. Densely connected convolutional networks, *arXiv:1608.06993v5*, 2018.
- [23] Cem Boga, Tahsin Koroglu. Proper estimation of surface roughness using hybrid intelligence based on artificial neural network and genetic algorithm, *Journal of Manufacturing Processes*, Vol. 70, pp. 560–569, 2021.
- [24] Zhenhui Wang, Juan Lu, Chaoyi Chen, Junyan Ma, Xiaoping Liao. Investigating the multi-objective optimization of quality and efficiency using deep reinforcement learning, *Applied Intelligence*, 2022.
- [25] Ghassan Al-Kindi, Hussien Zughraer. An Approach to Improved CNC Machining Using Vision-Based System, *Materials and Manufacturing Processes*, Vol. 27, pp. 765–774, 2012.
- [26] Ravi Sekhar, T. P. Singh, Pritesh Shah. Machine learning based predictive modelling and control of surface roughness generation while machining micro boron carbide and carbon

- nanotube particle reinforced Al-Mg matrix composites, *Particulate Science and Technology*, Vol. 40, pp. 355-372, 2021.
- [27] Tien-Dung Hoang, Quang-Vinh Nguyen, Van-Cuong Nguyen, Ngoc-Hien Tran. Self-adjusting on-line cutting condition for high-speed milling process, *Journal of Mechanical Science and Technology*, Vol. 34, pp. 3335-3343, 2020.
- [28] Anbesh Jamwal, Rajeev Agrawal, Monica Sharma, G.S Dangayach, and Sumit Gupta, Application of optimization techniques in metal cutting operations: A bibliometric analysis, *Materials Today: Proceedings*, 2020.
- [29] Industrial automation systems and integration — Physical device control — Data model for computerized numerical controllers — , ISO 14649-1.
- [30] E. H. Glaessgen, D.S. Stargel. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles, *American Institute of Aeronautics and Astronautics*, Paper for the 53rd Structures, Structural Dynamics, and Materials Conference, 2012.
- [31] Tommaso Tassi. A stabilization methods for transport dominated problems enhanced by Artificial Neural Networks: towards turbulence LES modeling, *Politecnico di Milano*, 2021.
- [32] C.C. Aggarwal. Neural Networks and Deep Learning, *Springer International Publishing*, 2018.
- [33] A. Ng. Convolutional Neural Networks, <https://www.coursera.org/learn/convolutional-neural-networks>.
- [34] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics*, 1943.
- [35] Bernard Marr. A Short History of Machine Learning – Every Manager Should Read, *Forbes*, 2016.
- [36] A. Ng. Machine learning by Stanford university, <https://www.coursera.org/learn/machine-learning>.
- [37] <https://www.pentamachine.com/>.
- [38] <https://mpwr.iscar.com/Milling/MachiningPower>.
- [39] <https://www.keyence.com/ss/products/measure-sys/vr/>.
- [40] <https://www.bysameyee.com/usb-digital-microscope-40x-to-1000x-bysameyee-8-led-magnification-endoscope-camera-with-carrying-case-amp-metal-stand-compatible-for-android-windows-7-8-10-linux-mac-p00082p1.html>.
- [41] LeCun, Yann; Cortez, Corinna; Burges, Christopher C.J. "The MNIST Handwritten Digit Database", *Yann LeCun's Website yann.lecun.com*. Retrieved 30 April 2020.
- [42] <https://www.cs.toronto.edu/kriz/cifar.html>.

- [43] <https://pytorch.org/>.
- [44] <https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a>.
- [45] <https://stats.stackexchange.com/questions/465937/how-to-choose-delta-parameter-in-huber-loss-function>.
- [46] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization, *arXiv:1412.6980*, 2014.
- [47] Lorenzo Maggi. A tutorial on Bayesian optimization with Gaussian processes, *Laboratory of Information, Networking and Communication Sciences*, <https://www.youtube.com/watch?v=VWLI1jthE24>, 2021.
- [48] Peter I. Frazier. A tutorial on Bayesian Optimisation, *arXiv:1807.02811v1*, 2018.
- [49] Alejandro Morales-Hernández, Sebastian Rojas Gonzalez, Inneke Van Nieuwenhuysse, Jereon Jordens, Maarten Witters, Bart Van Doninck. Constrained multi-objective optimisation of process design parameters in settings with scarce data: an application to adhesive bonding, *arXiv:2112.08760v1*, 2021.
- [50] Carl Edward Rasmussen, Christopher K. I. Williams. Gaussian Processes for Machine Learning, *The MIT Press*, (20005).
- [51] Ali Baheri, Sharim Bin-Karim, Alireza Bafandhe, Christopher Vermillion. Real-time control using Bayesian optimization: A case study in airborne wind energy systems, *Control Engineering Practice*, Vol. 69, pp. 131-140, 2017.
- [52] Jiyoung Jung, Kundo Park, Byungjin Cho, Jinkyoo Park, Seunghwa Ryu. Optimization of injection molding process using multi-objective bayesian optimisation and constrained generative inverse design networks, *Journal of intelligence manufacturing*, 2022.
- [53] Florian M. Heckmeier, Christian Breitsamteru. Aerodynamic probe calibration using Gaussian process regression, *Meas. Sci. Technol.*, pp. 31-125301, 2020.
- [54] Sheffield University. A Gaussian Process (GP) framework in Python, <https://gpy.readthedocs.io/en/deploy/>.
- [55] Sheffield University. a Python open-source library for Bayesian Optimization, <http://sheffieldml.github.io/GPyOpt/>.
- [56] https://secondmind-labs.github.io/markovflow/notebooks/choosing_and_combining_kernels.
- [57] Dong C. Liu. and Jorge Nocedal, On the limited memory BFGS method for large scale optimization, *Mathematical Programming*, Vol. 45, pp. 503-528, 1989.
- [58] Kundo Park, Youngsoo Kim, Minki Kim, Chihyeon Song, Jinkyoo Park, Seunghwa Ryu.

- Designing staggered platelet composite structure with Gaussian process regression based Bayesian optimization, *Composites Science and Technology*, Vol. 220 , 2022.
- [59] Shi Cheng, Yuhui Shi, and Quande Qin. On the Performance Metrics of Multiobjective Optimization, *Lecture Notes in Computer Science*, Conference Paper, 2012.
- [60] Selenium package, <https://anaconda.org/conda-forge/selenium>.
- [61] Ali Baheri, Shamir Bin-Karim, Alireza Bafandeh, Christopher Vermillion. Real-time control using Bayesian optimization: A case study in airborne wind energy systems, *Control Engineering Practise*, Vol. 69, pp. 131-140, 2017.
- [62] Myunghee Kim, Ye Ding, Philippe Malcolm, Jozefien Speeckaert, Christoper J.Siviy, Conor J. Walsh, Scott Kuindersma. Human-in-the-loop Bayesian optimization of wearable device parameters, *PLOS ONE*, 2017.
- [63] Cristian L. Cortes, Pascal Lefebvre, Nikolai Lauk, Micheal J. Davis, NEil Sinclair, Stephen K. Gray, Daniel Oblak. Sample-efficient adaptive calibration of quantum networks using Bayesian optimization, *arXiv:2106.06113v1*, 2021.
- [64] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, Nando de Freitas. Learning to learn by gradient descent by gradient descent, *30th Conference on Neural Information Processing Systems*, 2016.
- [65] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, Sergey Levine. Meta-Learning with Implicit Gradients, *arXiv:1909.04630*, 2019.

A | Appendix A: Main Scripts

A.1. Machining Parameters Combination

MATLAB script used to create all the possible machining parameter combinations to create the samples:

```

1  clc
2  clear all
3  close all
4
5  % mac_comb = xlsread('machined_combo.xlsx',1,'T2:V121') ;
6
7  diameter = 4;
8  ap = [0.2, 0.4, 0.6, 0.8, 1, 1.2];
9  n = [21500, 20000, 20000,20000,20000,20000];
10
11 per_ae = [0.15, 0.30, 0.45, 0.60, 0.75, 0.90];
12 %ae = round(diameter.*per_ae,1);
13 ae = [0.2, 0.4, 0.6];
14 min_f = 500;
15 max_f = 800;
16 delta_f = 100;
17
18 feed = min_f:delta_f:max_f;
19
20 AP = zeros(length(ap),length(ae)+1);
21 AP(:,1) = ap;
22 AP(:,2:end,1) = ae.*ones(length(ap),length(ae));
23
24 AP(:,2:end,2) = min_f.*ones(length(ap),length(ae));
25 AP(1,2:end,3) = max_f;
26 AP(2,2:end,3) = [800 800 700];
27 AP(3,2:end,3) = [800 700 600];
28 AP(4,2:end,3) = [800 600 500];
29 AP(5,2:end,3) = [600 500 400];
30 AP(6,2:end,3) = [500 500 400];
31 % AP(7,2:end,3) = [300 250 200 150 75 50 50];
32 %%
33 row = 1;
34 %combos

```



```

35 for i = 1:length(ap)
36     for j = 2:length(ae)+1
37         for k = 1:find(feed==AP(i,j,3))
38             combo(row,1) = AP(i,1,1);
39             combo(row,2) = AP(i,j,1);
40             combo(row,3) = feed(k);
41             combo(row,4) = round(pi*n(i)*diameter/1000,1);
42             combo(row,5) = combo(row,3)/(n(i)*2);
43             row = row+1;
44         end
45         row = row+1;
46     end
47     row = row+1;
48 end
49
50 combo(combo(:,1)==0,:)=[];
51 updated_combo = combo;
52 index=[];
53
54 % for i = 1:size(mac_comb,1)
55 %     in = find(sum(combo(:,1:3) == mac_comb(i,:),2)==3);
56 %     index = [index;in];
57 % end
58 updated_combo(index,:) = [];
59 writematrix(updated_combo, 'updated_combo');
60
61 %% Organising them
62 n=20;
63 ap1 = updated_combo(updated_combo(:,1)==ap(1),:);
64 ap2 = updated_combo(updated_combo(:,1)==ap(2),:);
65 ap3 = updated_combo(updated_combo(:,1)==ap(3),:);
66 ap4 = updated_combo(updated_combo(:,1)==ap(4),:);
67 ap5 = updated_combo(updated_combo(:,1)==ap(5),:);
68 ap6 = updated_combo(updated_combo(:,1)==ap(6),:);
69
70 % l = [size(ap1,1), size(ap2,1), size(ap3,1), size(ap4,1), size(ap5,1), size(ap6
71     ,1)];
72 l = [size(ap1,1), size(ap2,1), size(ap3,1), size(ap4,1), size(ap5,1)];
73 l_sorted = sort(l)

```

A.2. Data Enhancement

Python script used to enhance the data:

```
1
2 import numpy as np
3 import os
4 import glob
5 import cv2
6 import torchvision
7 from PIL import Image
8 from matplotlib import pyplot as plt
9 import csv
10 import pandas as pd
11 import openpyxl
12 trs1 = torchvision.transforms.ColorJitter(brightness=0.1, contrast=0.1, hue=0.1)
13 trs2 = torchvision.transforms.RandomAdjustSharpness(sharpness_factor=20)
14 trs3 = torchvision.transforms.RandomRotation(degrees=(0, 90))
15 trs4 = torchvision.transforms.ColorJitter(brightness=0.25, contrast=0.25, hue
    =0.25)
16 trs5 = torchvision.transforms.ColorJitter(brightness=0.5, contrast=0.5, hue=0.5)
17 trs6 = torchvision.transforms.RandomHorizontalFlip(p=0.99)
18 def aug_save(img, imgname, path):
19     tr_img1 = trs1(img)
20     tr_img2 = trs2(img)
21     tr_img3 = trs3(img)
22     tr_img4 = trs4(img)
23     tr_img5 = trs5(img)
24     tr_img6 = trs6(img)
25
26     dummy_imgname = imgname.split(".")[0]
27
28     img1name = dummy_imgname + '_aug1.png'
29     img2name = dummy_imgname + '_aug2.png'
30     img3name = dummy_imgname + '_aug3.png'
31     img4name = dummy_imgname + '_aug4.png'
32     img5name = dummy_imgname + '_aug5.png'
33     img6name = dummy_imgname + '_aug6.png'
34
35     img.save(path+'/'+imgname)
36     tr_img1.save(path+'/'+img1name)
37     tr_img2.save(path+'/'+img2name)
38     tr_img3.save(path+'/'+img3name)
39     tr_img4.save(path+'/'+img4name)
40     tr_img5.save(path+'/'+img5name)
41     tr_img6.save(path+'/'+img6name)
42
43     names = [[imgname], [img1name], [img2name], [img3name], [img4name], [
    img5name], [img6name]]
44
```

```
45     return(names)
```

A.3. Model Training

Python script used to train the models:

```
1  import timm
2  import torch
3  import torch.nn
4  import torchvision
5  from torch import nn
6  from torch.utils.data import Dataset
7  from torchvision import datasets
8  from torchvision.transforms import ToTensor
9  from torch.utils.data import DataLoader
10 import torchvision.transforms as transforms
11 import matplotlib.pyplot as plt
12 import random
13 import numpy as np
14 from customDataser import Roughness
15 from torchvision.transforms import ToTensor
16 from torchvision.transforms import CenterCrop
17 from torchvision.transforms import Normalize
18 from torch.utils.data import random_split
19 import pickle
20 import matplotlib.pyplot as plt
21 import statistics
22 from torchmetrics import R2Score
23 from torchmetrics import MeanAbsolutePercentageError
24 from torchmetrics import MeanAbsoluteError
25 from GPUUtil import showUtilization as gpu_usage
26 from numba import cuda
27 from Functions import csv_range
28 from Functions import imshow
29 from Functions import model_test
30 from Functions import correlation
31
32 # CNN Model
33 model_type = 'resnet50'
34 loss_type = 'MSE'
35 device = torch.device('cuda')
36 itererations = range(2,9)
37 l_rates = [0.0005, 0.0005, 0.0005, 0.0005, 0.0003, 0.0003, 0.0001, 0.0001,
38           0.0001, 0.00005]
39
40 for iteraz in itererations:
41     it_num = iteraz
42     model_number = 3
```

```

43     composed_res = transforms.Compose([ToTensor(), CenterCrop(224), Normalize(
44         mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
45     composed_xce = transforms.Compose([ToTensor(), CenterCrop(229), Normalize(
46         mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])])
47
48     if model_type == 'xception':
49         composed = composed_xce
50         model = timm.create_model(model_type, pretrained=False, num_classes = 1)
51         model = model.to(device)
52
53     elif model_type == 'resnet50':
54         composed = composed_res
55         model = torch.hub.load('pytorch/vision:v0.10.0', model_type, pretrained
56 = False)
57         model.fc = nn.Linear(2048, 1)
58         model = model.to(device)
59     # DATASET LOADING
60     if it_num <= 1:
61         # print('Fully connected layer:',model.fc)
62
63         # dataset = Roughness(csv_file = 'C:/Users/AMSE/OneDrive/Filippo/
64 Educazione/3-University/KALST/Kaist_university/Thesis/Machine_Learning/Git/
65 code_mine/CNN/Datasets/dataset.csv', root_dir = photo_path, transform =
66 composed)
67         # train_data = Roughness(csv_file = mother_csv+'/dataset_train.csv',
68 root_dir = photo_path, transform = composed)
69         # val_data = Roughness(csv_file = mother_csv+'/dataset_val.csv',
70 root_dir = photo_path, transform = composed)
71         # test_data = Roughness(csv_file = mother_csv+'/dataset_test.csv',
72 root_dir = photo_path, transform = composed)
73
74         # print('Training dataset length:', len(train_data))
75         # print('Validation dataset length:', len(val_data))
76         # print('Test dataset length:', len(test_data))
77         # print('Datset length:', len(dataset),'+=' ,len(train_data)+len(
78 val_data)+len(test_data))
79
80         # pickle.dump(dataset, open(mother_csv+'/' +model_type+'/data_all.dat', '
81 wb'))
82         # pickle.dump(train_data, open(mother_csv+'/' +model_type+'/data_train.
83 dat', 'wb'))
84         # pickle.dump(val_data, open(mother_csv+'/' +model_type+'/data_val.dat',
85 'wb'))
86         # pickle.dump(test_data, open(mother_csv+'/' +model_type+'/data_test.dat
87 ', 'wb'))
88         # print('Model and Data created for the:', model_type)
89         train_data = pickle.load(open(mother_csv+'/' +model_type+'/data_train.dat
90 ', 'rb'))

```

```

78     val_data = pickle.load(open(mother_csv+'/'+model_type+'/data_val.dat', '
rb'))
79     test_data = pickle.load(open(mother_csv+'/'+model_type+'/data_test.dat',
'rb'))
80     dataset = pickle.load(open(mother_csv+'/'+model_type+'/data_all.dat', '
rb'))
81     print('Model Created')
82
83     if it_num > 1:
84         model.load_state_dict(torch.load(model_path+'/trained_model'))
85         # model = torch.hub.load('pytorch/vision:v0.10.0', model_type,
pretrained = False)
86         # model.fc = nn.Linear(2048, 1)
87         model = model.to(device)
88         train_data = pickle.load(open(mother_csv+'/'+model_type+'/data_train.dat
', 'rb'))
89         val_data = pickle.load(open(mother_csv+'/'+model_type+'/data_val.dat', '
rb'))
90         test_data = pickle.load(open(mother_csv+'/'+model_type+'/data_test.dat',
'rb'))
91         dataset = pickle.load(open(mother_csv+'/'+model_type+'/data_all.dat', '
rb'))
92
93         print('Training dataset length:', len(train_data))
94         print('Validation dataset length:', len(val_data))
95         print('Test dataset length:', len(test_data))
96         print('Datset length:', len(train_data)+len(val_data)+len(test_data))
97         print('Model and Data Loaded for the:', model_type)
98
99     # TRAINING
100     # HYPER PARAM
101     learning_rate = l_rates[iteraz-1]
102     # learning_rate = 0.0005
103     batch_size = 16
104     n_epoch = 8
105     delta_h = 0.8
106
107     # Datasets
108     train_dataloader = DataLoader(train_data, batch_size, shuffle=True)
109     val_dataloader = DataLoader(val_data, 1, shuffle=True)
110     test_dataloader = DataLoader(test_data, round(len(test_data)*0.5)+1, shuffle
=True)
111     random_dataloader = DataLoader(test_data, 4, shuffle=True)
112
113     # Training
114     # Losses per batch
115     optimizer = torch.optim.Adam(params = model.parameters(), lr = learning_rate
)
116     MSE = nn.MSELoss()
117     MSE = MSE.to(device)

```

```

118 MAE = MeanAbsoluteError()
119 MAE = MAE.to(device)
120 MAPE = MeanAbsolutePercentageError()
121 MAPE = MAPE.to(device)
122 HUBER = nn.HuberLoss(delta = delta_h)
123 HUBER.to(device)
124 r2score = R2Score()
125 r2score = r2score.to(device)
126 batch_loss = []
127 MSE_val = []
128 MAE_val = []
129 MAPE_val = []
130
131 print("Initial GPU Usage")
132 gpu_usage()
133
134 for epoch in range(n_epoch):
135
136     print('Learning rate:', learning_rate)
137
138     for i,data in enumerate(train_dataloader):
139
140         inputs, labels = data
141         labels = labels.type(torch.FloatTensor)
142         inputs = inputs.to(device)
143         #print(inputs.size(), labels.size())
144         y_pred_train = model(inputs)
145         y_pred_train = y_pred_train.to(device)
146         labels = labels.view([len(labels), 1])
147         labels = labels.to(device)
148         train_loss = MSE(y_pred_train, labels)
149         train_loss.backward()
150         optimizer.step()
151         optimizer.zero_grad()
152         batch_loss.append(train_loss.item())
153
154         # Validation of training
155
156         loss_MSE, loss_MAE, loss_MAPE, R2 = model_test(val_data, device,
157 model, save_var_path+'/Training/Loss', '', False)
158         MSE_val = np.append(MSE_val, loss_MSE)
159         MAE_val = np.append(MAE_val, loss_MAE)
160         MAPE_val = np.append(MAPE_val, loss_MAPE)
161
162         if i%50 == 0:
163             print('Epoch {}, batch {} loss: {}'.format(epoch + 1, i + 1,
164 train_loss))
165
166     model.train()

```

```

166     # if epoch % 3 == 0:
167     #     [prediction, g_truth] = correlation(dataset, device, model,
save_var_path+'/Figures/Correlation'+str(epoch), False)
168     #     pickle.dump(prediction, open(save_var_path+'/Training/
correlation_p'+str(epoch)+'.dat', 'wb'))
169     #     pickle.dump(g_truth, open(save_var_path+'/Training/correlation_g'+
str(epoch)+'.dat', 'wb'))
170
171     print(f"epochs {epoch+1}, loss:{train_loss}")
172
173     # Variable save
174     torch.save(model.state_dict(), save_var_path+'/trained_model')
175     pickle.dump(batch_loss, open(save_var_path+'/Training/Loss/batch_loss'+
.dat', 'wb'))
176     pickle.dump(MSE_val, open(save_var_path+'/Training/Loss/MSE'+'.dat', 'wb
'))
177     pickle.dump(MAE_val, open(save_var_path+'/Training/Loss/MAE'+'.dat', 'wb
'))
178     pickle.dump(MAPE_val, open(save_var_path+'/Training/Loss/MAPE'+'.dat', '
wb'))
179
180     # TEST
181     model.load_state_dict(torch.load(actual_model_path+'/trained_model'))
182     model_test(test_data, device, model, save_var_path+'/Test', '', True)
183
184     # Plotting the errors
185     batch_loss = pickle.load(open(save_var_path+'/Training/Loss/batch_loss'+'.
dat', 'rb'))
186     MSE_val = pickle.load(open(save_var_path+'/Training/Loss/MSE'+'.dat', 'rb'))
187     MAE_val = pickle.load(open(save_var_path+'/Training/Loss/MAE'+'.dat', 'rb'))
188     MAPE_val = pickle.load(open(save_var_path+'/Training/Loss/MAE'+'.dat', 'rb'
))
189
190     x_batch = list(range(len(batch_loss)))
191     fig1 = plt.figure()
192     plt.plot(x_batch, batch_loss)
193     plt.plot(x_batch, MSE_val)
194     plt.plot(x_batch, MAE_val)
195     plt.plot(x_batch, MAPE_val)
196     plt.xlabel('Number of batches')
197     plt.ylabel('Errors per Batch')
198     plt.title('Losses')
199     plt.legend(["Train", 'MSE', 'MAE', 'MAPE'], loc="upper right")
200     #plt.show()
201     plt.savefig(save_var_path+'/Figures/Training_errors')
202
203     # Plotting the correlation
204     prediction, g_truth = correlation(dataset, device, model, save_var_path+'/
Figures', True)

```

```
205     pickle.dump(prediction, open(save_var_path+'/Training/prediction'+'.dat', '
206     pickle.dump(g_truth, open(save_var_path+'/Training/g_truth'+'.dat', 'wb'))
207
208     print('Number of excecuted Epochs:', iteraz*8)
```


B | Appendix B: CNC Specifications

Pocket NC V2-50CHB technical specifications:

Variable	Value
Spindle speed	from 1000 to 50000 rpm
Spindle run-out	2.5 μm
Power	600W
Max tool Diameter	4 mm
Collet	CHB 1.5, 3, 4 mm μm
Max X travelling	115.5 mm
Max Y travelling	128.3 mm
Max Z travelling	90.1 mm

List of Figures

1	Sketch of an Orthogonal cutting model. The tool is moving and it is perpendicular to the piece along the cutting line.	2
2	Sketch of a more realistic Orthogonal cutting model. The sliding between planes can be noticed and the chip deformation is added.	3
3	Representation of a basic milling operation. The main milling parameters have been included.	3
4	Facing and slotting operations. During slotting the minimum width of the channel that can be created is the diameter of the tool.	4
5	Conventional linear facing path and trochoidal one. The blue lines represent when the tool is cutting while the yellow ones when the tool is not engaged with the material.	5
6	Solid tool and tool body with insert descriptions and comparison. Credits to [5].	6
7	Scheme representing the bottom and side view of a single cutting insert and the tool body in which the radial and axial rake angles can be seen (pay attention to the fact that the rotation directions are different for each view).	7
8	Tool wear representation according to the the ISO standards. [7].	8
9	Typical tool wear evolution according to different cutting speeds. [6].	8
10	R_a definition according to ISO 4287:1997. l_r represents the sampling or inspection length considered for the evaluation and the centre line is such that the aggregate of the zones over the line is equivalent to the aggregate of the regions beneath the line. Credits to [9].	9
11	Example of a machinist table that can be found in any machinist or mechanical manuals. Credits to Wikipedia.	10
1.1	The CNC machine used in this work (Pocket NC V2-50 CHB, see the Appendix B: CNC Specifications, to look for the technical specifications).	18
1.2	Development phases of a machined piece before actually entering the CNC machine.	19
1.3	Example of a G-Code produced by the post processor of the CNC machine used in this work.	20
1.4	General structure of the perceptron. Inspired by the connections between biological neurons [31], [35].	22

1.5	Example of ANN with n inputs nodes, two hidden layers (one with three neurons the other with two) and a single output node.	23
1.6	A 2D convolution operation explained. In this case the filter extracts the trace of each sub-matrix to make it easier to understand.	25
1.7	Scheme of a complete convolutional neural network.	26
2.1	Aluminium billets after being measured and catalogued.	28
2.2	3D model of a machined billet. On the top surface each rectangle at a different height represents a different sample. The deepest ones will have rounded edges in the real billets, generated from the milling operation.	28
2.3	Step by step procedure to machine a single billet. Inside the CAM software the CNC company post processor was installed.	28
2.4	Cutting table of a family of tools based on the operation and material. Credit to MISUMI corp.	29
2.5	The different machining strategies developed by the CAM software.	30
2.6	Two machined aluminium billets. The four different adopted machining strategies can be noted on the surface.	31
2.7	The measuring of the S_a and R_a of a machined billet. The optical machine can be noticed with the sample laying on the measuring table. The markings to identify the billet (blue letters) can also be noticed.	31
2.8	On the left is shown the global view of the camera mounted on the machine. On the right the camera holding device.	32
2.9	On the left is portrayed the full bright taken picture. On the right the half bright one.	32
2.10	Applied enhancements. [A] Real image, [B] Jitter change, [C] Sharpness change, [D] Rotation, [E] Jitter change, [F] Jitter change and [G] Mirror.	33
2.11	On the left (a) it is represented the general structure (not the complete one) of the ResNet50 with the fully connected layer highlighted by the red dotted circle. On the right (b), the structure of a residual block with the skip connection is highlighted by the orange dotted circle. [20].	34
2.12	On the left (a) is shown the general structure (not the full one) of the Xception with the residual layers highlighted by the orange dotted rectangular boxes and the fully connected layer by the red dotted circle. On the right (b), the structure of a Depthwise Separable Convolution block is schematised. [21].	35
2.13	Total dataset distribution among different roughness values.	37
2.14	Kernels comparisons based on different hyperparameters. Credit to [53].	42
2.15	Initial points generated using Latin Hypercube Sampling.	44
2.16	Web built interface that controls the CNC machine. The red box represents the part related to the uploading of a new G-Code, the orange one directly acts on the machine movements and can change the V_f and rpm while the tool is cutting, while the blue one indicates the axis coordinates in real time.	46

2.17	On the left is pictured the original G-Code. On the right the modified one with the added parts highlighted by the blue dotted box.	47
3.1	Training and validation errors of the Xception model trained using the <i>MSE</i> loss function. The right image represents a zoom of the left one.	49
3.2	Evolution of test errors (Xception model trained using the <i>MSE</i> loss function). Each epoch group considers the model after 8 epochs of training.	50
3.3	Correlation evolution between the ground truth and the predictions per epoch group (Xception model trained using the <i>MSE</i> loss function).	50
3.4	<i>AE</i> distribution calculated over the whole dataset (Xception model trained using the <i>MSE</i> loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.	51
3.5	Training and validation errors of the Xception model trained using the <i>MAPE</i> loss function. The right image represents a zoom of the left one.	51
3.6	Evolution of test errors (Xception model trained using the <i>MAPE</i> loss function). Each epoch group considers the model after 8 epochs of training. The R^2 is zero in the first epoch because it resulted in a negative value.	52
3.7	Correlation evolution between the ground truth and the predictions per epoch group (Xception model trained using the <i>MAPE</i> loss function).	52
3.8	<i>AE</i> distribution calculated over the whole dataset (Xception model trained using the <i>MAPE</i> loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.	53
3.9	Training and validation errors of the Xception model trained using the <i>Huber</i> loss function. The right image represents a zoom of the left one.	53
3.10	Training error evolution per iteration (Xception model trained using <i>Huber</i> loss function).	54
3.11	Evolution of test errors (Xception model trained using the <i>Huber</i> loss function). Each epoch group considers the model after 8 epochs of training.	54
3.12	Correlation evolution between the ground truth and the predictions per epoch group (Xception model trained using the <i>Huber</i> loss function).	55
3.13	<i>AE</i> distribution calculated over the whole dataset (Xception model trained using the <i>Huber</i> loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.	55
3.14	Training and validation errors of the ResNet50 model trained using the <i>MSE</i> loss function. The right image represents a focus on a particular range of iterations of the left one.	56
3.15	Evolution of test errors (ResNet50 model trained using the <i>MSE</i> loss function). Each epoch group considers the model after 8 epochs of training. The R^2 is zero in first epoch because it resulted in a negative value.	56
3.16	Correlation evolution between the ground truth and the predictions per epoch group (ResNet50 model trained using the <i>MSE</i> loss function).	57

3.17	<i>AE</i> distribution calculated over the whole dataset (ResNet50 model trained using the <i>MSE</i> loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.	57
3.18	Training and validation errors of the ResNet50 model trained using the <i>MAPE</i> loss function. The right image represents a zoom of the left one.	58
3.19	Evolution of test errors (ResNet50 model trained using the <i>MAPE</i> loss function). Each epoch group considers the model after 8 epochs of training. The R^2 is zero in first epoch because it resulted in a negative value.	58
3.20	Correlation evolution between the ground truth and the predictions per epoch group (ResNet50 model trained using the <i>MAPE</i> loss function).	59
3.21	<i>AE</i> distribution calculated over the whole dataset (ResNet50 model trained using the <i>MAPE</i> loss function). The dotted red line represents the number of samples analysed. In this case is the size of the database itself.	59
3.22	Correlations comparisons of the three Xception models over the whole dataset.	61
3.23	On the left are represented the results of the correlation over the test data of the model trained with the <i>MSE</i> loss function. On the right the results of the Xception model trained with the <i>Huber</i> loss function.	61
3.24	The first prediction is made using the model trained with the <i>MSE</i> loss function. The second, using the model trained with the <i>Huber</i> loss function.	62
3.25	<i>MAE</i> calculated over the total available samples divided by S_a ranges.	62
3.26	Correlations comparisons of the two ResNet50 models over the whole dataset.	63
3.27	On the left are shown the results of the correlation over the test data of the model trained with the <i>MSE</i> loss function. On the right the results of the model trained with the <i>MAPE</i> loss function.	64
3.28	The first prediction is made using the model trained with the <i>MSE</i> loss function. The second, using the model trained with the <i>MAPE</i> loss function.	64
3.29	<i>MAE</i> calculated over the total available samples divided by S_a ranges.	65
3.30	On the left are represented the results of the estimations when the input are picture taken using the full brightness of the camera's LED. On the right, when half brightness is used.	65
3.31	Analysis of the <i>AE</i> between the estimations performed over the new 52 machine samples.	66
3.32	Posterior mean, Posterior standard deviation and Acquisition function of the different models created using different weights. They are named "Posterior" because computed after the GPR model was fitted on the available data-points and the BO process was concluded. The red points present on the first two heat maps represent the data-points used to fit the model. The one on the last picture to the right instead, represents the newly suggested one from the BO process.	67

3.33	Posterior mean, Posterior standard deviation and Acquisition function of the different models created using different weights. They are named "Posterior" because computed after the GPR model was fitted on the available data-points and the BO process was concluded. The red points present on the first two heat maps represent the data-points used to fit the model. The one on the last picture to the right instead, represents the newly suggested one from the BO process. . . .	68
3.34	Suggested points by the optimiser after the first iteration is completed.	69
3.35	Correlation between S_a and MRR that generates a Pareto with some of the achieved results, after the first iteration of the optimiser.	69
3.36	The final assembled system. There can be noticed the python script, the CNC controller web page, and the machine with the installed monitoring system. . . .	70
3.37	On the left is portrayed the machined surface generated following the initial G-Code. On the right the one using the modified one. The correspondent G-Codes are represented below and the machining parameters are highlighted by the dotted boxes.	70

List of Tables

2.1	Machining parameter ranges divided by tool diameter and relative to the aluminium billets. Please notice that the choice of one parameter influences the range of the others due to the limited power of the machine.	30
2.2	Pipeline of pre-processing transformations.	37
2.3	Learning rate evolution over 10 epochs.	37
2.4	Corners of the hyper-space of the inputs of the process.	44
2.5	Weights used to create the different objective functions.	45
3.1	Best models errors and metric calculated using the complete dataset and the test one.	60

List of Most Important Symbols

Variable	Description	unit of measurement	SI unit
W_o	width of cut	mm	m
V_c	cutting speed	m/min	m/s
V_f	feed rate	m/min	m/s
f	feed	mm per rev	m/rad
t_o	uncut chip thickness	mm	m
t	real chip thickness	mm	m
γ	rake angle	degrees	rad
α	clearance angle	degrees	rad
n	angular speed	rpm	rad/s
MRR	Material Removal Rate	mm^3/min	m^3/s
a_p	axial depth of cut	mm	m
a_e	radial depth of cut	mm	m
z	number of teeth	-	-
C	machining constant	mm/min^{1-n}	m/s^{1-n}
R_a	Arithmetical mean height of the assessed profile	μm	m
S_a	Arithmetical mean height of the assessed surface	μm	m
R^2	Coefficient of determination	-	-

List of Abbreviations

Abbreviation	Description
<i>2D</i>	Two Dimensional
<i>3D</i>	Three Dimensional
<i>CNC</i>	Computer Numerical Control
<i>ISO</i>	International Organization for Standardization
<i>CAD</i>	Computer Aided Design
<i>CAM</i>	Computer Aided Manufacturing
<i>ANN</i>	Artificial Intelligence
<i>ML</i>	Machine Learning
<i>CNN</i>	Convolutional Neural Networks
<i>MNIST</i>	Modified National Institute of Standards and Technology
<i>CIFAR – 10</i>	Canadian Institute For Advanced Research-10
<i>GPU</i>	Graphics Processing Unit
<i>MSE</i>	Mean Squared Error
<i>MAPE</i>	Mean Absolute Percentage Error
<i>MAE</i>	Mean Absolute Error
<i>AE</i>	Absolute Error
<i>RL</i>	Reinforced Learning
<i>GPR</i>	Gaussian Process Regression
<i>BO</i>	Bayesian Optimization
<i>LED</i>	Light Emitting Diode

Acknowledgements

Since my love for mechanics started I have always been cursed and followed by dreams and ideas, even this work is one of them. The combination of old and new has always fascinated me and the possibility of looking forward standing on the giants' shoulders is truly a gift. This thesis was developed gathering all the experience gained through times spend in machine shops, nights fixing my Scarabeo, hours talking to and working with machinists, an unlimited amount of minds sharing between my university and high school friends and of course, a huge dose of aspiration generated by the one and only Iron Man movie. This was consolidated and deepened throughout my all university career and especially thanks to my Advisor, professor *Sanha Kim*; who followed my development and the thesis from the beginning to the end instructing and showing me the beautiful and impactful part of research aimed to change the world into a better place. Thank you. In addition, I would love to thank KAIST for accepting and supporting me with all the necessary services to arrive at this point. Also, my lab-mates are part of this. Their help and availability were crucial and precious elements that helped me in some of the hard times during my university and personal career here in Korea. Thank all of you. A special thanks goes to *Gyuhyeon Han* who helped me with the AI part sharing with me his knowledge, time and coding resources introducing me to this amazing paradigm. Also *Taemin Kim* thank you for your time and help provided replying to my infinite number of questions and requests, together with *Dong Geun Kim* who was the first that followed me during my first research project. Thank you to all of you. A big thanks also goes to my Italian advisor, professor *Massimiliano Annoni* who helped me to consolidate my work in my home university in Italy. I would love also to thank Politecnico di Milano for giving me the chance and opportunity to become and engineer able to reason and get out of trouble with my own mind and to discover how beautifully natural phenomenon and events can be explained using maths. Even if the journey at POLIMI was long and sometime painful was sincerely worthed and contributed in moulding the Filippo that I am today. Of course, I cannot forget to thank my big family. All of them have been always close to me and supported me on every roller coaster I drove even if dead-ended. My father was always one of my biggest inspirations and role models; he guided me since I was born together with my mother and my brothers who always believed in me and pushed me to be the best fuelling me with the most genuine competitiveness. My grandfather was a big mentor that always supported me and helped me in all the ways through my life together with my grandmothers who always prayed and made sure that I would not have an empty stomach. Thank you to all of you. A special thank also goes to my uncle Guido who told me to start university because otherwise : 'Non sai cosa ti perdi' or you don't know what you are missing. You were damn right. Thank you

to all the other members of my family, my cousins, my uncles and aunts. Thank you to all my university friends that I met in Milano, especially Giulio, Jacob, Gabriele, Matteo, Tommaso, Davide, Claudio, Alessandro, Michele, and Nicola. You were always next to me through the whole university experience and we shared some of the best memories of my life so far. Your contribution to this work and to my personal development was and is still massive. Thank you to all of you. Thank you, Tommy, I cannot be more grateful to had a mentor like you both during my university life and also while enjoying Milano, you were the person that opened the first door to a new world that changed my life, forever. Thank you Andre Gori; even if we spent a short amount of time together we had a lot of fun and enjoyed our time even during the stressful parts of the university. Thank you to Gully one of my oldest friends, we shared so much together that one book would not be enough to cover all. Even if now we are far, I will not forget the battles and the fun we had together that forged me to become the Filippo I am today. Thank you to Frucci, my oldest friend. We always shared beautiful conversations and experiences that are impossible to forget and helped me to realise how lucky I am to grow up with you. Thank you to Leone, those high school times we had together are some of the best ever. Even after that, we shared so many good moments, conversations and every time I speak with you it is like knowing a new part of me and meeting one. Thank you to Leo Bocca with who I shared some remarkable moments of my childhood, especially during the Summer, which developed me so much as a person. Your confidence, maturity and sarcasm are something that I truly admire. Thank you to my high school classmates. I cannot forget those moments and how beautiful they were and valuable. Thank you to my old appa mates in Milano Bovisa that helped me facing the advertises and obstacles of the university as a rookie and taught me the importance of sharing. In particular, thank you to Nello, you were among the first ones to show me how important it is to care more about learning than taking good marks and how important it is to be driven by passion. Thank you to my appa mates of Milano Dergano. That combination of engineers and humanity was an amazing experience that showed the importance of embracing differences as points of strength. Thank you, Alfred, Petu, Cola, Gigi and Zanna. Thank you to all the wonderful people I have met in Korea so far who enriched this experience even more with their different cultures and with the great moments shared together. Thank you to Sara and Olli the Italians with who I have shared this since the beginning and who stayed close to my shoulder against shoulder. Thank you to Andrea Finazzi with who I shared some amazing times, conversations and soccer even if we got to know lately. Thank you to Hans, my last roommate and great tennis partner, we shared a lot inside the university and outside, even while having different morning routines. Thank you to Jakob(I know this is the wrong spelling) who helped me with my thesis by discussing with me the AI part while sharing some good memories together. Thank Bruno, you are a great mind and a very kind person always available to help others. Your AI knowledge is massive and helped me considerably. In addition, we shared some beautiful moments inside and outside the pitch. Thank you to Bellaya, you showed me some amazing and breathtaking parts of life that I did not know yet and what it means to be a genuine and authentic leader not only inside the pitch. Even as an 'agiosci' you

still can play longer than me. Thank you to Wabi we shared so many talks on that roof in N7 that I think we can write a novel out of it. We shared some good and beautiful times together and you pushed through my thesis every day. Thank you Ibra, your kindness and discipline are certainly an example that I will not forget and we had a lot of fun together. Moreover, you helped me with my university and professional life every time I asked you. Thank you Dom, we fought and bled together but we enjoyed some truly authentic moments that I will remember forever; your authenticity and inner strength are amazing. Thank you Xhaseem you are such a kind and professional guy. But your social skills are incredibly good. You were one of my strongest examples in Korea and what you are doing for your country is amazing. Thank you Dani, you are a such calm and polite guy, I really appreciated the time we spent together. Thank you to my Chugku FC team. We built a really strong connection during this year here in Korea that helped me to learn and enjoy the time there even more. This is not just a football team is a family. Last but not least I would like to thank *Chaeram*, she stayed close to me during the last crucial months of this work supporting me mentally and helping me relax and get comfortable even during the most stressful times.

Finally, I wanted to thank all my school professors, especially the Buzzi ones that showed me how beautiful mechanics is and pushed me to know more about it. In the end, I would like to thank all those people that did not believe in me, all those girls that rejected me; thank you all, you throw another log into my motivational fire and you pushed me to work harder and harder to follow my real passions.

Che dire, dopo tutto questo papiro in inglese non potevo non aggiungere una parte in italiano. Anche se ultimamente sogno e mastico inglese dalla mattina alla sera sono pure sempre un italiano con globuli rossi e bianchi toscani. E da buon italiano non posso non ringraziare la mia famiglia con tutto il cuore, gli americani dicono 'blood is thicker than water' ma noi invece diciamo 'che te lo dico a fa'. Grazie a mio padre che è stato il primo a farmi innamorare della meccanica con la sua BMW GS 80 Basic e con quel tavolo della Beta arancione scintillante che, guarda caso, ha fatto proprio partire la scintilla; e quell' Andy Warhol di cui ci raccontavi da piccoli che ci ha fatto capire di quanto, anche se ingegneri, bisogna lasciare spazio tra l'ingranaggi per un pennello o al connubio con un campo totalmente diverso, perché 'Fil, l'ingegneria della Vespa era un aerospaziale e Dio non tira di certo i dadi'; grazie a mia madre donna di gran cuore che mi ha sempre supportato e ascoltato in qualsiasi momento e mi ha sempre mantenuto umile, parsimonioso e attento. Come diceva 2Pac: 'For a woman it ain't easy tryin' to raise a man' ma te ci sei riuscita alla grande. Grazie a miei due fratelli Edo e Danilo, abbiamo condiviso tanto insieme e ancora c'è da fare vai; non basterebbero milioni di parole per descrivere la nostra storia, quindi che dire questo traguardo è anche vostro e spero che possa spingervi nel raggiungere i vostri sogni più impossibili perché io sarò sempre il primo a supportarvi e combattere con voi. Non posso dimenticarmi di ringraziare mio nonno Raffaello, una delle persone più fondamentali della mia vita e uno dei miei mentori più importanti insieme a mio padre. Mi hai sempre supportato psicologicamente, moralmente ed economicamente e mi hai trasmesso talmente tanto

che è dura trovare poche parole per esprimere la mia gratitudine e fortuna per averti avuto sempre vicino. Filù è passato da piantare bullette e rubare chiodi ai muratori a ingegneria meccanica. Come dicevo sempre da bambino: 'Mio nonno è un mago'. Grazie alle mie due nonne Maria e Maria Assunta che non mi hanno mai fatto patire la fame e mi hanno sempre supportato perfino spiritualmente con i loro immensi Rosari e preghiere, insegnandomi quanto sia bella la vita quando si vive semplicemente e ci si rende conto di quanto siamo fortunati. Grazie ai miei cugini con cui ho passato dei momenti memorabili e trascorso innumerevoli estati, grazie a tutti i miei zii e zie che hanno sempre creduto in me, tutti i parenti di Castagneto che mi hanno fatto passare sempre delle estati meravigliose e mi hanno insegnato il valore della terra e del rispetto della natura. Infine grazie a mio nonno Danilo, che pur non avendolo mai conosciuto, è riuscito a trasmettere e imprimere valori come l'onore, il rispetto e il non arrendersi mai che sono stati fondamentali nel mio percorso sia personale che professionale, non sarai mai dimenticato.