UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Márcio Sumariva Nandi

Tool Wear Prediction System Using Deep-Learning Techniques on High Precision Milling Process

Florianópolis
2020

Márcio Sumariva Nandi

Tool Wear Prediction System Using Deep-Learning Techniques on High Precision Milling Process

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação na disciplina **DAS 5511: Projeto de Fim de Curso** do curso de Graduação em Engenharia de Controle e Automação.
Orientador(a): Prof. Marcelo Ricardo Stemmer

Florianópolis
2020

Márcio Sumariva Nandi

Tool Wear Prediction System Using Deep-Learning Techniques on High Precision Milling Process

AGRADECIMENTOS

Gostaria de expressar minha gratidão aqui por todas as pessoas que estiveram presentes comigo durante o percurso deste curso de graduação, em especial aos meus colegas e amigos de curso que sempre expressaram união em busca dos melhores caminhos para resolver os problemas que todos enfrentamos em nosso percurso profissional até aqui. E por oferecer e compartilhar visões e oportunidades que não conseguimos a princípio ver sozinhos.

Gostaria de agradecer também às instituições que me propiciaram oportunidades de crescer, primeiramente à UFSC que apesar dos recursos limitados e impasses políticos me propiciou formação acadêmica, profissional e ética com respeito e qualidade. Gostaria muito que todas as pessoas que à atacam com argumentos vagos tivessem a oportunidade de formação semelhante à que tive, para então poder debater com clareza e seriedade.

Em segundo lugar ao Instituto Fraunhofer por propiciar esses valiosos anos de aprendizado profissional e aplicado. E que me auxiliaram na realização desse documento e de todas as atividades relacionadas a ele. Agradecimentos especiais à minha orientadora Zhen Zhen, e ao meu colega e líder de departamento Tae Hun Lee.

Meus agradecimentos especiais à minha segunda família: a todos os integrantes e ex-integrantes da República Matagal. Apesar da distância, vocês sempre farão parte viva e vibrante da minha vida. O que vivemos e compartilhamos durante os anos que estive lá ficarão marcados para sempre. Muito obrigado pelo apoio, e pela amizade que com certeza continuará.

Por fim, gostaria de expressar minha profunda e imensa gratidão por minha família. Não poderia deixar de dedicar parte especial desse texto a vocês, pai e mãe. Que durante todo esse período de luta sempre estiveram do meu lado me apoiando no que fosse possível, mesmo que as condições parecessem dizer o contrário. Espero do fundo do coração que esse texto seja apenas um símbolo de tudo que gostaria de retribuir pelo que me fizeram durante todo esse período. Vocês são minha motivação, meu exemplo e meu objetivo. De coração, meu muito obrigado.

RESUMO

O projeto visa implementar um sistema para predição de desgaste de ferramenta em processos de fresamento de alta precisão. A abordagem escolhida utiliza dados de sensores obtidos de testes reais realizados em uma fresadora CNC de alta precisão combinado com medições de desgaste no flanco da ferramenta realizadas via microscópio. Com objetivo de selecionar as melhores metodologias para o desenvolvimento do projeto, uma cuidadosa pesquisa do estado da arte foi realizada. Nesta fase, as arquiteturas de *deep-learning* e estudos envolvendo do fenômeno de desgaste foram revisadas no intuito de selecionar a abordagem mais apropriada para a resolução do problema. O sistema de aquisição implementado grava dados de: potência consumida pelo eixo-árvore, emissão acústica e hipersônica, forças aplicadas na ponta da ferramenta e vibração no corpo do eixo-árvore. Portanto, o documento aborda o *setup* de *hardware* e *software* para o sistema de aquisição utilizado na máquina, planejamento dos parâmetros do processo e análise dos dados coletados. Além disso, o projeto e implementação do sistema de aquisição é descrito. Depois disso, o projeto e implementação do módulo de pré-processamento é relatado. Depois que os dados dos experimentos são adquiridos, os arquivos são processados por este módulo, que extrai informação de emissão acústica, ultrassônica, e vibração para gerar os *datasets.* Nessa etapa, todos os *datasets* são gerados utilizando transformada de Fourier, considerando que um dos objetivos do projeto é comparar o desempenho de diferentes sensores no problema de predição de desgaste de flanco. Ao final, o módulo de predição é descrito. O documento discute o uso de diferentes arquiteturas de redes neurais, técnicas para extração de *features* e optimização do treinamento. Para o projeto, três diferentes arquiteturas de *deep-learning* foram escolhidas para a tarefa de predição. O projeto compara o desempenho de cada arquitetura e cada sinal usado. Os resultados mostraram uma performance superior para os dados de vibração em combinação com as redes LSTMs, alcançando 81% de precisão no modo de classificação, e 176 µm de erro quadrático médio no modo de regressão.

Palavras-chave: Predição de Desgaste de Ferramenta. Redes Neurais Artificiais. Deep-Learning. Fresamento de Alta Precisão. Sistema de Aquisição.

ABSTRACT

The project is aimed to implement a system for tool wear prediction for high precision milling process. The approach chosen uses sensor data gathered from real test performed in a CNC machine center for precision milling combined with microscope measurement of tool flank wear. In order to select the best methodologies for the project development, a careful state of the art research was carried out. On this step, the deep-learning architectures and researches involving tool wear phenomena were revised in order to select the most appropriate approach to solve the problem. The acquisition software system implemented records data coming from: the spindle power consumption, acoustic and ultra-sonic emission, forces applied in the tool tip and vibration of the spindle body. Therefore, the document englobes the hardware and software setup for the acquisition system on the machine, process parameters planning and analysis of the data collected. Besides that, the project and implementation of the acquisition software is described. After that, the pre-processing module project and implementation is reported. After the experiment data is gathered, the data files are processed by the pre-processing module, which firstly extracts information from the acoustic, ultra-sonic emission, and vibration files in order to generate the datasets. On this steps, all datasets are generated using fast Fourier transformation, once one of the goals of this work is to compare the performance of different sensors in the flank wear prediction task. At last, the prediction module is described. The document discuss the use of different neural network architectures, feature extraction and training optimization techniques. For the project, three different deep-learning architectures were chosen for the prediction task. The project compares the performance between each architecture and each used signal. The results showed a superior performance for the vibration data in combination with LSTMs achieving 81% of accuracy on the classification approach and a mean squared error of 176 µm on the regression approach.

Key-words: Tool Wear Prediction. Artificial Neural Networks. Deep-Learning. High Precision Milling. Acquisition System.

LIST OF FIGURES

LIST OF TABLES

ABREVIATIONS AND ACRONYMS

AE	Acoustic Emission
AI	Artificial Intelligence
ANN	Artificial Neural Network
BPTT	Back Propagation Through Time
CNC	Computerized Numeric Control
DAQ	Data Acquisition module
DNN	Deep Neural Network
DL	Deep Learning
DoE	Design of Experiments
FIFO	First In First Out
FFT	Fast Fourier Transform
GUI	Graphical User Interface
HDF	Hierarchical Data Format
HMM	Hidden Markov Model
IPT	Institute for Production Technology
LSTM	Long Short-Term Memory
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
PC	Personal Computer
PLC	Programmable Logic Controller
ReLU	Rectified Linear Unit
RMS	Root Mean Square
RNN	Recurrent Neural Network
RPM	Rotations Per Minute
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TBPTT	Truncated Back Propagation Through Time
TCM	Tool Condition Monitoring
TCP	Tool Center Point
WME	Window Mean Error

ZMQ	Zero M Queue communication framework

SYMBOLS AND NOTATIONS

ω_k	k-esian weight of neuron
$f(\cdot)$	activation function
x	input vector
y	output label vector
$\bar{y}$	predicted output
X	inputs of a dataset
Y	output labels of a dataset
$\bar{Y}$	prediction values over a dataset
$L(y, \bar{y})$	Loss function
net_j	activation input signal of neuron j
δ_j	gradient value for a single neuron
∇	gradient vector
μ	learning rate
$\Delta\omega$	weight correction
$E(x)$	error function
o_i	output of neuron i
$\sigma(\cdot)$	sigmoid function
$\tanh(\cdot)$	hyperbolic tangent
h	hidden state of the neural network
c	cell vector of Long Short-Term Memory networks
$freq$	frequency
VB	flank wear
gr	groove number
P_k	kth degree Polynomial function
NN_k	neural network at epoch k
W	training weights for a dataset
ϵ	epsilon: very small value, around 10^{-8}

SUMMARY

1. INTRODUCTION

1.1. Motivation

The metal cutting industry is one of the oldest and most important manufacturing processes of our society. It is present in almost any sector of the market over almost all production systems nowadays. Inside metal cutting, milling is one of the most relevant machining processes, not only because of its power to produce complex workpiece profiles, but also because of its flexibility in manufacturing different types of goods with the same equipment [3].

The increasing demand for customized and flexible production has led the machining industry to new challenges at the same time as increasing precision is demanded [5]. Furthermore, even though this field of production exists for decades, it still pursues methods to decrease downtime and production cost, preventing accidents and equipment damage. Besides that, themes like chattering control, tool condition monitoring and tool breakage detection are still addressed as unsolved questions.

In order to attend the mentioned new and old demands, the metal cutting industry is moving gradually towards the complete integration of the shop floor. Big equipment providers are migrating to open communication frameworks; the presence of smart components is increasing every year; open frameworks have been developed for acquisition, monitoring and control [5].

Allied with those resources, the Big-Data age also came as a handy tool in order to provide new solutions for the machining process. In the context of integrated manufacture, acquired data now can be stored and used as elements for new powerful tools to monitor and optimize such processes. The usage of Big-Data with Machine Learning (ML) is causing recently a revolution in the industry [37]. Once the integration of the shop-floor is implemented and data is available, a new range of solutions becomes possible.

All these trends can be perceived as small steps towards a new way of producing. However, there is still work required in order to combine technologies like smart-sensors, big data-sets and smart systems. Furthermore, system for position error compensation, tool condition monitoring, tool breakage detection and thermal compensation present limitations or need adjustments in order to be transported to industrial environment [5].

1.2. Fraunhofer IPT

Allying the upcoming equipment and systems produced in the market with new key technologies like ML and Big Data is the main goal of Fraunhofer Institut für Produktionstechnologie (Institute for Production Technology – IPT) – see Figure 1. The institute is known for providing technology systems for production all around the world.

Figure 1 – Fraunhofer Institute for Production Technology.

Source: [1]

Fraunhofer is widely spread across the world with more than 80 working units, most of them located in Germany. It has partnership with big companies in the fields of machining, automobilists and aero-space industry.

The IPT Institute has the task of transporting technology from the academic environment directly into industrial practice as itself defines [1]. The focus is to provide reliable tools for the specific tasks required from customers. In other words, the projects use updated state of the art together with high technology equipment in order to develop innovative systems for the industrial environment.

The goal of the Precision Technology and Automation Department – the place where this project was developed – is to provide solution for high precision machining. As the name suggests, this branch demands smaller tolerance levels and better positioning control to maintain the desired precision. So, error sources like thermal deflection, chattering, tool wear and positioning offset are more harmful to those processes in comparison to conventional operations.

In order to study those entities, it is necessary to acquire information from sensors and the machine PLC system. Therefore, works regarding high frequency data acquisition are also carried out in the department.

1.3. Problem Description

The present work is aimed to develop a system using ML or more precisely Deep Learning (DL) – which is going to be defined afterwards – in order to generate models for tool flank wear prediction.

Flank wear can be defined as the loss of material on the edge of the cutting tool, caused by the constant contact between the tool and the workpiece. As described by [2] the reasons behind tool wear can be the shock between tool and workpiece, abfraction on the cutting edge and imperfections on the tool composition. Usually it increases gradually during the working time of the tool.

In high precision milling process, the effects of tool wear are intensified due to the small tolerance level required – in the order of micrometers. Since the tool wear affects the shape and sharpen of the tool, the side effect is transferred to the workpiece. That is, deterioration of the tool may provoke undesired vibration on the system and poor surface quality. Resulting in rework or even rejection of the manufactured product.

Besides that, the tool cutting power decreases with the increase of tool wear, consequently increasing the load on the machine. Which can also damage the equipment.

By monitoring the tool wear phenomena, it is possible to determine the exact moment for the tool change. However, measuring directly this entity in the industrial environment is a very costly procedure, which requires interrupting the process, removing the tool and using special equipment to check its status. Such procedures are undesired in those environments.

In practice, an expert operator replaces the machine tool after a given lifetime threshold. This method is low accurate and can cause problems. On one hand, overusing a tool can damage workpiece and machine as previously described. On the other hand, underusing it will generate extra direct and undirect (setup time) costs [3].

Studies reported that 6-20% of the downtime in milling is caused by tool wear and tool breakage [3, 4]. Some authors ranked such factors as the biggest barrier in the actual manufacture industry. In [5], the author presented the concept of smart spindles, which is a new generation of rotary spindles developed to fulfill the most important tasks for the Industry 4.0 incoming requirements of flexibility and quality. In the third chapter, when approaching the key technologies required on those new devices, the author first mentioned the importance of Tool Condition Monitoring (TCM). The text highlighted it as one of the critical factors in order to achieve high accuracy and efficiency in milling, once it is directly related to the performance of the machine.

Also related to the topic, according to [4], a good TCM system integrated to the milling machine can increase the lifetime of the cutting tool on 10-50%, shorting the downtime and cutting costs in the order of 10-40%.

Zhou and Xue [3] pointed out that studies involving TCM have been developed for more than 30 years. Although there is still not a consensus about which is the best way to measure and control the tool wear progress on milling processes. As direct measurement for the issue is out of hand, the indirect measurement is then recommended.

1.4. Objective

In order to build the referred prediction module, deep learning techniques will be studied and employed in the project. However, in order to use such data driven technique, It is mandatory to possess data related to the process. Therefore, a set of experiments was designed in order to generate this data for the training step. The experiments employed on the machine gather information from multiple sensors simultaneously.

The goal of this work is to implement prediction models using different DL architectures. On this way, each combination of sensor data and DL architecture will be tested. Finally, by analyzing the performance of each model trained for the task, the most deterministic sensor data and also the most suitable technique can be identified. Furthermore, the presented results provide further understanding about the signal behavior for the studied phenomena.

As a result, this project will present the following elements:

- A sensor system installation for studying the tool wear phenomena during milling operation;
- The project and implementation of the acquisition system responsible for the end-point sensors data acquisition;
- A dataset containing all information gathered during the experiments performed on the milling machine;
- And finally, a software programmed to train the prediction models. This system pre-processes the data, extracts the relevant features, and trains the final models using different approaches.

1.5. Monography Structure

Chapter 2 presents a literature review about the relevant knowledge fields applied in this work, including machine learning, deep-learning, TCM and tool wear prediction.

On Chapter 3 the experiment step is reported, giving details about the hardware and software setup. The chapter also presents the component description for all employed sensors.

Chapter 4 talks about the Acquisition software project and implementation, as well as the results achieved by the system.

Chapter 5 presents a signal analysis of the sensor data gathered on the experiments. Following by the pre-processing strategy for extracting information from them.

Chapter 6 presents the AI modules. Listing the algorithms, methods used and training results providing a detailed comparison between each trained prediction model.

The text is concluded in chapter 7, which gives the results and limitations of the work. The author finishes the document by listing suggestions for future works.

2. LITERATURE REVIEW

2.1. Artificial Intelligence

Throughout history Artificial Intelligence (AI) has received different definitions. Russel and Norvig [6] classified AI into 4 different categories: systems that thinks like humans, system that act like humans, system that thinks rationally and systems that act rationally. Between the given definitions, one of the most implied is “The branch of computer science that is concerned with the automation of intelligent behavior.” Or in other words, it is the field of study that aims to program computers to perform actions which demand human intelligence.

Machine Learning (ML), as described by Arthur Samuel [7], is the “Field of study that gives computer the ability to learn without being explicitly programmed.” On [8] the author also defined it as “A Computer program is said to learn from an experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.” In general terms, the second definition is a formalization of the first one.

[9] explained better the process of learning when saying the ML algorithms use sample data called “training data” in order to generate knowledge models. In order to generalize the observed examples, the algorithm uses statistical tools for regression and decision making instead of arithmetic rules.

ML is divided into mainly two categories [9][10]: supervised and unsupervised learning:

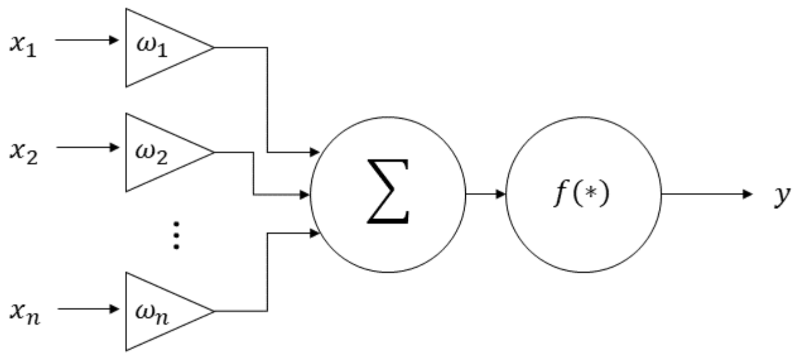
- Supervised learning: On this mode the reference data is available to the algorithm. So, the computer has a dataset containing inputs and target outputs. The process of learning happens by presenting the dataset to the model multiple times. The number of samples mapped correctly from input to output determine the accuracy of the model. The main examples of this method are classification and regression.
- Unsupervised learning: on this mode, the reference data is not available to the algorithm. Therefore, the task consists in observing and finding patterns between different inputs. By using this inference mechanism, it is possible to discover structures common to each group on the dataset. Clustering is the most typical example of this approach.

2.1.1. Artificial Neural Networks

The most popular approach for supervised learning is the Artificial Neural Network (ANN) [11]. Those networks have been used in a wide variety of problems with success due to its capacity of generalization in presence of noisy input data [3]. Because of its massive parallelization, the mechanism is powerful in representing strong non-linear functions [10]. By doing so, ANNs can interpret problems and perform decisions without any assumption of the data's structure [12].

An ANN, as explained in [13], uses a representation of human brain in order to process information. For this task, it has a model of an artificial neuron containing input weights, an activation function and an output, see Figure 2. Primarily, the inputs of the neuron are multiplied by their respective weights. Then, the result is summed up and passed through the activation function. This simple representation can also be translated into equation 1.

Figure 2 – Simple model of an Artificial Neuron.

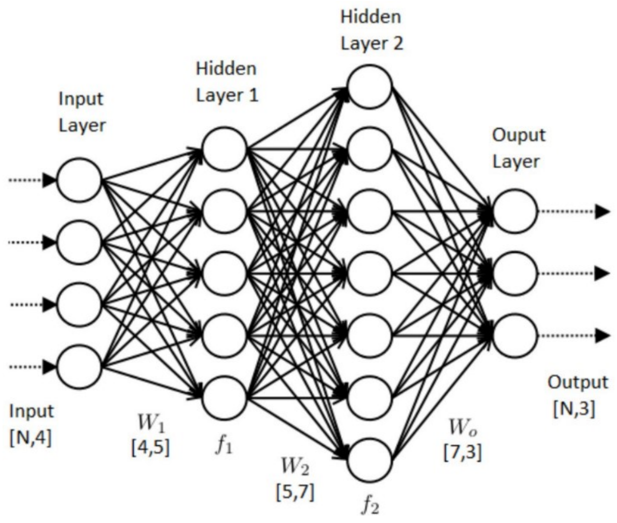


Source: [13]

$$y = f\left(\sum_{i=0}^{n} x_i \,.\, \omega_i\right) \qquad (1)$$

So, a neural network is nothing but a pool of neurons, grouped in a certain number of layers determined by the expert. Each node receives the output of all nodes of the preceding layer and its output will feed all the neurons in the following layer. Figure 3 presents an example of such. Therefore, the information flows through three different types of layers: the input layers, which receive the input vector; the hidden layers located in the middle of the network; and the output layer, which determines the prediction value.

Figure 3 – Example of an ANN.



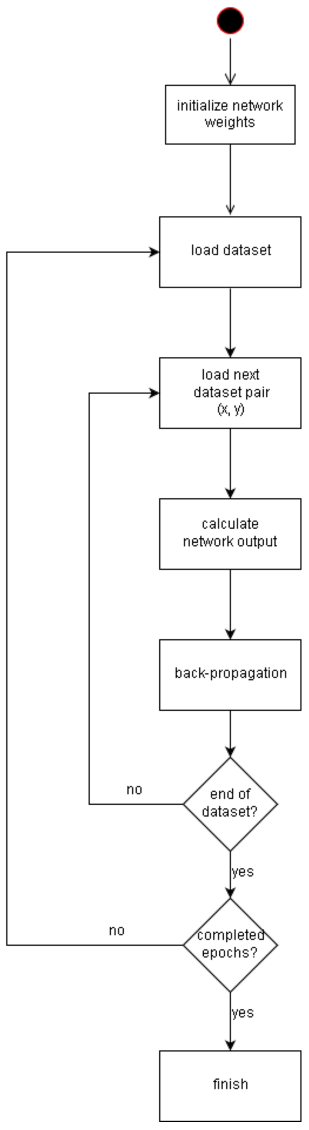
Source: [13]

In general terms, the input and output layer are fixed in the algorithm´s architecture, since they process the input and target output vectors respectively. Therefore, by raising the size and depth of the hidden layers it is possible to increase the representation power of a network as a total [14]. On the other hand, a bigger network enlarges the time and dataset size required to train it. As a result, projecting an ANN or any of its variants becomes an intuitive and context specific task [3].

The procedure for training ANNs consists of presenting the training examples to the network n times – called epochs – and using the back-propagation algorithm to adjust the weights $\omega_{i,j}$ of the artificial neurons. [15] revised the back-propagation algorithm and its popular variants since they are widely used in ML models. The author explained the algorithm for training neural networks in the following steps as shown in Figure 4. The 2 most important steps performed in such algorithm consists of:

- Forward step: a data example is presented and the output is calculated by the network;
- Backward step: an error signal is calculated from the difference between desired and predicted outputs. This signal, calculated by the loss function is back-propagated using a gradient-descent algorithm through the network until the input layer.

Figure 4 – Representation of back-propagation algorithm.



Source: [6]

So, considering $L(.)$ a loss function to calculate an error value between the network output $\bar{y}$ and the target values y as defined by Equation 2.

$$E = L(t, y) \qquad (2)$$

Given ω the weights of a neuron j with n inputs, net_j is defined as the input of the activation function of the referred neuron j as described in Equation 3.

$$net_j = \sum_{k=0}^{n} \omega_{kj} \qquad (3)$$

The $\delta(.)$ function calculates the gradient function of the error signal. The value calculated by Equation 4 will be further used to apply the correction on the weights of the network.

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j} = \begin{cases} \dfrac{\partial L\left(o_j, y\right)}{\partial o_j}\dfrac{df(net_j)}{dnet_j}, if\ j\ is\ an\ output\ neuron \\ \left(\sum_{l\,\in\,L} \omega_{jl}\delta_l\right)\dfrac{df(net_j)}{dnet_j}, if\ j\ is\ an\ inner\ neuron \end{cases} \quad (4)$$

In Equation 4 the derivative of the error is calculated from the derivative of the loss function for output neurons. When determining δ for the inner neurons, the error is calculated by multiplying the weights and δ values of the following layer L. On this way, the network is corrected from the output to the input layer. The term $\frac{df(net_j)}{dnet_j}$ is the derivative of the chosen activation function related to its input net_j.

The correction value $\Delta\omega_{ij}$ is then calculated multiplying a value μ corresponding to the learning rate. By changing the value of this factor, the network will present a more aggressive or soft behavior along the time. See Equation 5.

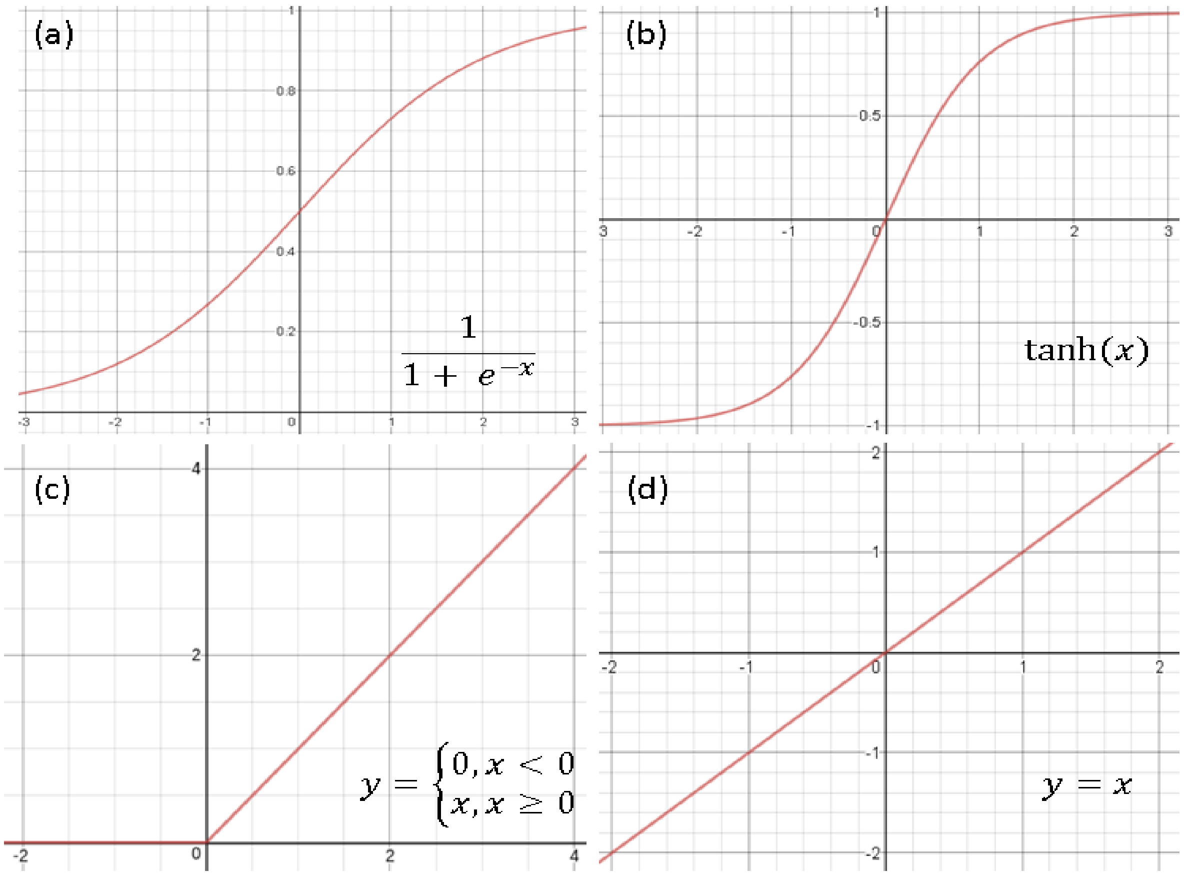
$$\Delta\omega_{ij} = -\mu\frac{\partial E}{\partial \omega_{ij}} = -\mu\ o_i\delta_j \quad (5)$$

For the back-propagation algorithm to work, the activation function $f(*)$ must be differentiable in all its extent. This property will guarantee a smooth correction vector applied to the weights of the network throughout the training epochs.

In the literature, we can find different functions used for activating the neurons. Their performance depends on the required application, output format and numeric cost in computing the forward and backward steps. Figure 5 shows some common examples of such functions.

It is common to find in the scientific and industrial world other procedures to modify the conventional back propagation algorithm. These changes are aimed to optimize its performance. Examples are batching the input data processed in each forward and backward steps; using momentum or any other not-fixed learning rate on the back-propagation step; and removing dead neurons to speed up convergence.

Figure 5 – Examples of activation functions.
(a) sigmoid; (b) tanh; (c) Rectified Linear Unit; (d) linear.



Source: [6]

2.2. Deep Learning

On the literature review done in [16], the author listed five different definitions for this field of knowledge. From which, some of them are:

I. "A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification."

II. "Deep Learning is a set of algorithms in machine learning that attempt to learn in multiple levels, corresponding to different levels of abstraction. It typically uses artificial neural networks. The levels in these learned statistical models correspond to distinct levels of concept, where higher-level concepts are defined from lower-level ones, and the same lower-level concepts can help to define many higher-level concepts."

In [17] the author defined Deep Learning (DL) as "a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces."

In other words, DL can be perceived as a set of tools which usually employ neural networks in their core mechanism. Therefore, their advanced structures are capable to interpret superior classes of complex data. By using multiple levels of abstraction, deep neural networks (DNN) are able to succeed in handling problems where simple neural networks have failed [8].

In face of highly noisy and complex data, simple ANNs must have a higher number of layers and neurons in order to interpret the data features. However, the training algorithm cannot fit the model in such cases, sometimes because of the unbearable dataset size, or most likely because the gradient-descent gets trapped into a local minimum point.

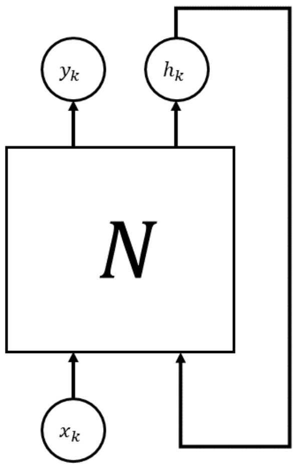
DNNs proved to outperform ANNs in problems like: image classification, pattern recognition, speech recognition and natural language processing [8, 17]. Recently those algorithms were successfully applied to signal processing. So, in the following sub-sessions it will be explained some DNN algorithms which will be used in the scope of this project.

2.2.1. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a special type of Recurrent Neural Network (RNN). RNNs are neural networks projected to interpret sequential data. Normal ANNs calculate output values based only on the data fed into the network, so these networks have problems when the past inputs are also important to determine the current output.

In RNNs, the current input and the hidden state are used in order to generate the output. So, the architecture of the RNN is very similar to the so-called vanilla ANNs. The difference is that now, the network must train also to keep a hidden state in memory. By using the past hidden state combined with the actual input value, the module can determine the next output and hidden state [18]. Figure 6 shows a schematic of an RNN, the Network N processes the input x producing the output y and the hidden state h, which will be used together with the next input to calculate the new output.

Figure 6 – Schematic of a Recurrent Neural Network.



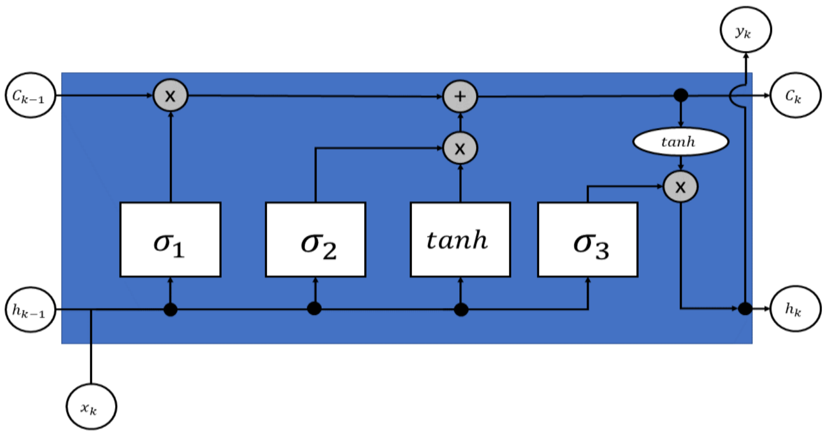
The RNNs are particularly interesting when processing sequential data. An example which is suitable for RNNs is machine translation. Once a sentence is processed sequentially by feeding into the network the chars of the expression.

Although, [19] showed the difficulties in learning long-term relations through Gradient-descent algorithms in RNNs. In short, the neurons responsible for maintaining the hidden state of the RNN are trained using one-step back-propagation. Therefore, it is easy to lose information from the hidden state when a task requires long-term memory once the network weights will be updated and the hidden state will be calculated a couple times.

The LSTM architecture is aimed to solve the lack of mechanisms to store long-term information. See Figure 7. The LSTM is a variation of the RNN, therefore it also process data sequentially. The upper line of the network corresponds to what's called cells (C). It is responsible for storing long-term information. As defined by [20], the LSTM possess three gates incorporated in its architecture, which actually are NN layers with Sigmoid activation.

The first gate σ_1 is responsible for selecting the information on the cell to be excluded, or possibly replaced. The second gate σ_2 selects the data which will be included on the cell. All gates use for their tasks the concatenated vectors $x_k|h_{k-1}$.

Figure 7 - Schematic of a LSTM network.

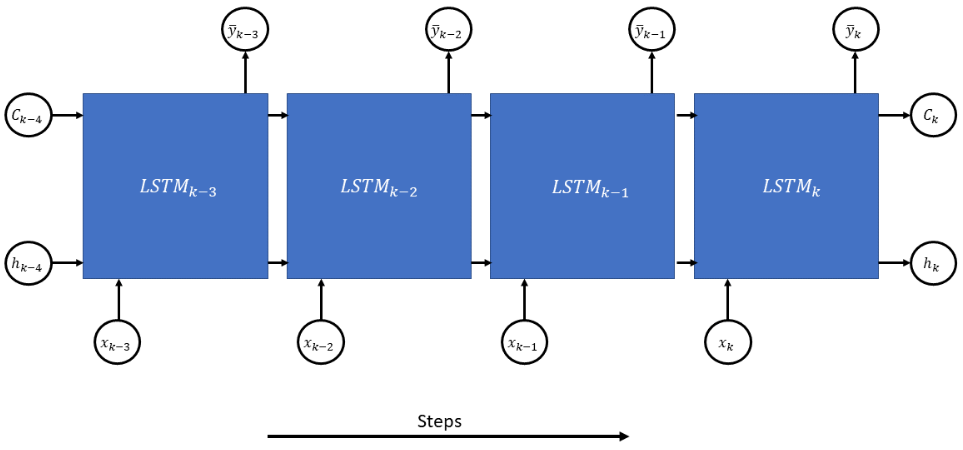


Source: [20]

The third network processes the information from the state and input in order to determine the new values which will be added to the cell. The output of this network is then multiplied by the second gate output vector and then added to the cell vector. The prediction value of the LSTM will then be calculated from the cell value, now updated with the new input. But, before selecting the output from the cell, the vector will be filtered out using a single $tanh$ layer in order to bound the values $-1 < c_{ki} < 1$. Finally, the third gate selects the new output and hidden state vectors.

Therefore, the information is processed in an LSTM using a chain mechanism. That is, short-term information navigates in the hidden state line, while long term relations on the dataset are stored and recovered from the cell line. See Figure 8.

Figure 8 – Representation of information flow in LSTM network.



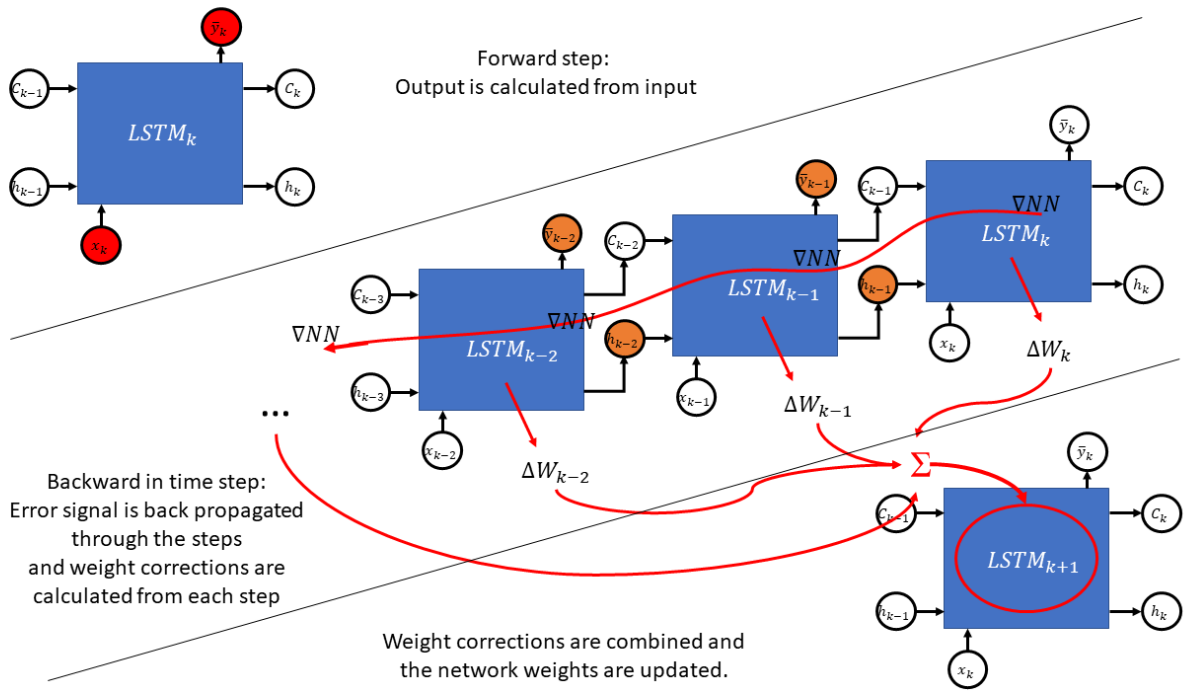
Source: [20]

Training the LSTM consists in training the four previously described networks together. In order to predict correctly, the LSTM gates must learn when to edit values on the cell. One important thing to pay attention to is the fact that the network now have to consider past computations, instead of one single back-propagation step as usually applied in simple ANNs. In this sense, [20] developed the Back-Propagation-Through-Time (BPTT) algorithm. Which apply the error gradients backward in time.

The BPTT algorithm can be perceived as an application of the Calculus chain rule when back propagating the error values in time. Theoretically, the BPTT works by unrolling all processed steps. Each step back in time holds one fed input, one copy of the network and one output. Errors are then calculated and accumulated for each step. The network weights are then updated with the accumulated error.

In other words, each step back in time taken by the BPTT may be seen as an additional layer, and the hidden state signal of the last step is taken as an input on the subsequent step. Figure 9 summarizes the working principle of the BPTT algorithm.

Figure 9 – Steps taken from BPTT algorithm for weight updating.


Forward step:
Output is calculated from input
Backward in time step:
Error signal is back propagated through the steps and weight corrections are calculated from each step
Weight corrections are combined and the network weights are updated.
$LSTM_k$
$LSTM_{k-1}$
$LSTM_{k-2}$
$LSTM_{k+1}$
∇NN
ΔW_k
ΔW_{k-1}
ΔW_{k-2}
Σ

Source: [20]

Computing the BPTT algorithm for n steps requires n weight correction updates and the algorithm must calculate the gradient n times for the network. On one hand, the computation of this algorithm is numerically expensive. On the other hand, computing multiple gradients can cause weights to vanish or explode. And make slow learning and model skill noisy.

Therefore, a variation of this algorithm called Truncated Back Propagation Through Time (TBPTT) is used. The new variation requires 2 parameters: k_1, the number of computing steps per update; and k_2, the number of timesteps to compute the BPTT. The new algorithm then compute the forward step for k_1 steps. After this, the BPTT is calculated for the last k_2 computations of the NN.

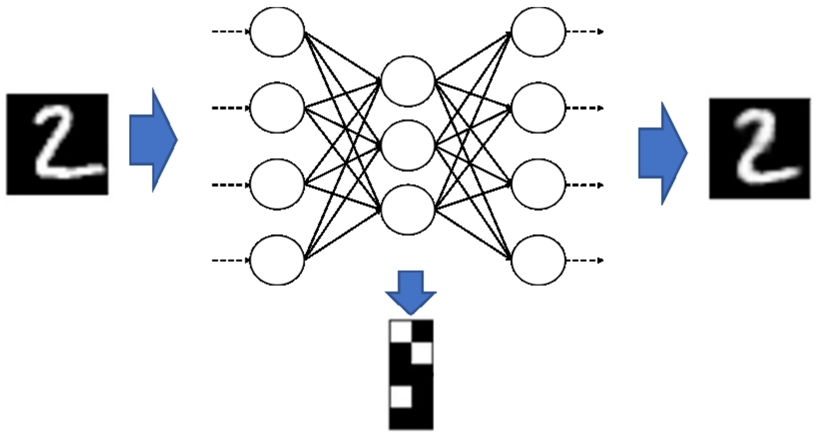
The LSTMs have proved to be efficient in memorizing sequential features and producing good results in fields like image generation, speech recognition and natural language processing. Their memory capacity have been a handy tool for a wide variety of problems.

2.2.2. Auto-Encoders

Auto-encoder is a Neural-Network architecture which tries to reproduce the same values from the input on the output [21]. Although it looks useless to have a tool to replicate values on the output, auto-encoders are a powerful unsupervised-learning tool for dimensionality reduction and feature extraction. This is because when it comes to this NN architecture, we are not interested in the network output layer. Instead, the interest resides entirely on the middle layer of the network. See Figure 10.

Auto-encoders are typically divided into 2 regions: the encoder, responsible for mapping the input to the so-called code – output of the middle layer in Figure 10; and decoder, responsible for interpreting the code and reconstructing the input vector.

Figure 10 – Representation of a simple auto-encoder.



Source: [21]

As the reader can notice, the encoder’s output has (always) a lower dimensionality when compared to the input size. This fact is particularly important to auto-encoders because the main goal is not to reproduce perfectly the input. Instead, the size of the code is kept smaller so the auto-encoder can only replicate approximately the input vectors.

Through this process, the encoder must learn a consistent way to compress the input vector, losing a minimum amount of information. Furthermore, the encoder ends up learning how to give priority to certain features on the input, providing valuable data interpretation.

The resulting logic is simple. If the decoder can create similar input vectors looking only at the code, this means the generated code has the most relevant information about the data.

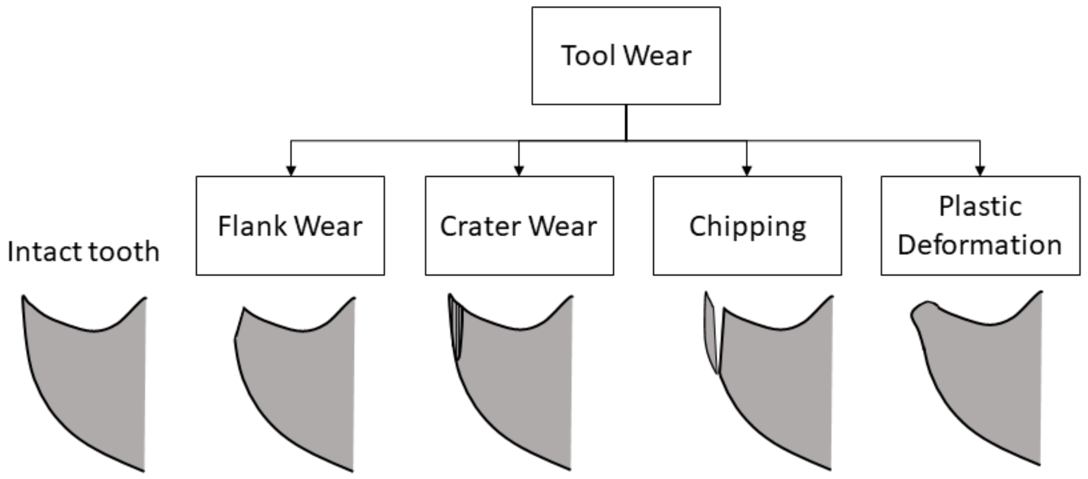
One common application of Auto-encoder in Deep learning consists of its use for feature selection. So, an input is compressed by an encoder before it is presented to the prediction network. On this way, the prediction model must learn how to interpret the code and perform its task, while the auto-encoder is responsible for dimensionality reduction, feature extraction and noise suppression of the dataset.

2.3. Tool Wear Prediction and Tool Condition Monitoring

When approaching the tool wear problem, [2] described some effects of the tool wear in the milling. According to the author, the flank wear is the prevalent wear type suffered by cutting tools on this type of process. Flank wear is defined as the loss of material in the relief face of the tool, caused mainly by the rubbing effect of this with the workpiece material.

[31] described some tool wear types which can be observed during the usable lifetime of a cutting tool. See Figure 11. When approaching the tool wear problem in High Speed Milling, the author pointed out that the flank wear and central wear (in spherical tools) are the most influencing types of wear in the progress of tool deterioration.

Figure 11 – Types of tool wear in milling tools.



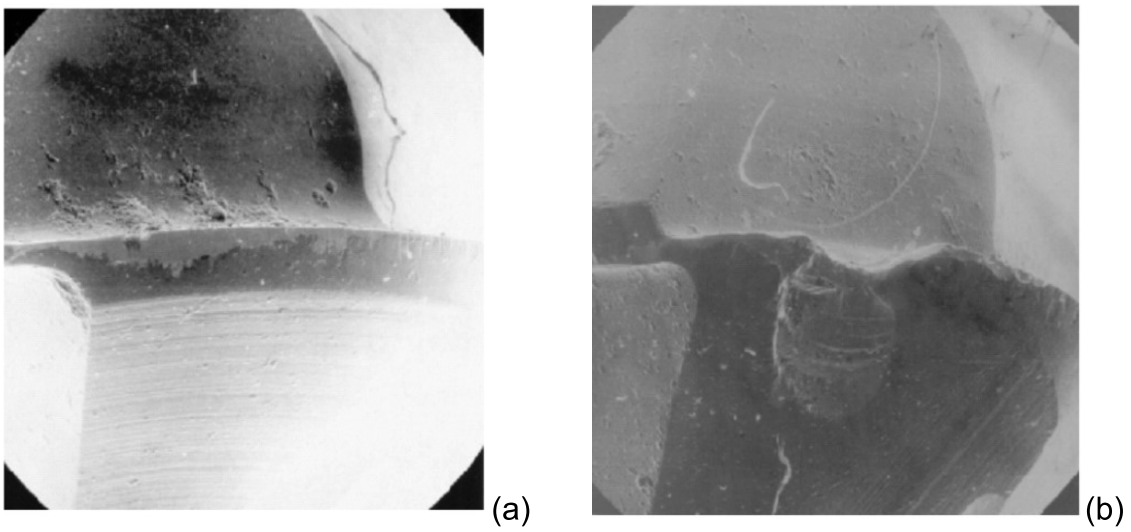
Also, when describing the tool wear progress characteristics [31], the document explained that on the beginning of the lifetime, the tool presents a first wear on the top of the cutting edge by a deformation on the cutter edge material. This deformation develops intensively for the first 30 minutes of operation. After this point, it can be observed a time interval when the wear stabilizes, and the surface quality even improves.

The flank wear continues to develop gradually until the protective coating of the tool surface is damaged. As a consequence, heavy material losses on the tool edge allied with an increasing wear surface provoke a rapid increase on the cutter load. This effect also results in increasing friction between tool and workpiece and high tool temperature. The result of this process is a fast development of the tool wear, the high edge temperature provokes other types of wear like crater and chipping. The degradation of the tool develops fast until it must finally be replaced. See Figure 12.

As properties like workpiece surface quality and process efficiency are strongly related to the tool condition, and this is strictly related to the tool wear progress, the study of TCM systems recently earned high attention on the field of metal cutting. As mentioned before, determining the right time for tool replacement plays an important role in order to minimize downtime and production costs. Furthermore, an accurate tool monitor can help operators to avoid possible damages on the equipment.

As methods for direct measurement of flank wear are usually expensive, and time inefficient, researches have been moving towards indirect measurement techniques. However, accessing this phenomena by indirect methods brings the challenge of dealing with the problem of noisy data.

Figure 12 – Tool edge showing allowed flank wear (a) and exceeded crater wear (b).



Source: [31]

On the review done by [4], the author approached the main methods for indirect sensing of tool wear. The first method involved cutting force measurement, usually performed through the use of table dynamometers. The paper emphasizes the problem of variability of the force signal related to the spindle position, once the signal has peaks on the entry and exit of each tooth on the material. This behavior can easily lead to false alarms when employing conventional methods like using a threshold criteria for the tool condition estimation.

Another reported method is the usage of acoustic emission (AE) sensing. As the flank wear develops, studies have reported an increase on the energy emitted by the AE signal, making it a handy tool for detecting tool wear and breakage. It was also explained that the prevalent frequency of those signals got higher on the end of tool life. On this case, mechanisms involving RMS, skew and fast Fourier transform (FFT) of such signals are usually applied in order to evaluate the output presented by those sensors.

Another source of information reported in [4] is the vibration sensing. This was usually carried out by installing accelerometers on a single tooth of the milling tool. Therefore, features can be identified when analyzing the behavior of the signal in some different frequency bands.

Other methodologies can be presented. However, the main point is to observe that analyzing those sensor outputs mostly requires developing complex methods for interpreting their information. Therefore, when coming to tool wear in milling, there is a high prevalence of statistical and smart system development for decision making. Even when mentioning a review published in 2003.

[5] raised another problem when approaching TCM using threshold criteria. The paper pointed to the fact that this methodology is highly dependent on the CNC process parameters. So, in order to produce a robust TCM system, the thresholds must have a high variety of parameters and complex programmed rules, otherwise their accuracy is heavily damaged.

The author proceeds by mentioning a strong trend in using machine learning to overcome this barrier, mainly because of its high generalization capacity. The paper reported an increasing usage of ANNs, support vector machines (SVM) and Hidden Markov Models (HMM) for the issue.

On [2] the author used Design of Experiments (DoE) – a statistic method for optimization developed by Genichi Taguchi – in order to design a dataset of 20 different process parameter combinations. The experiment plan was developed using 3 factors and 5 levels. The research used this dataset to train an ANN and fit a polynomial function for tool wear prediction using the process parameters as input. For labeling the data, a microscope measured the tool condition. The results presented 94% accuracy on the polynomial regression, although no test-set was used, which can be misleading.

A similar method was implemented by [35] where also 20 experiments were raised using DoE and AE sensing. The author applied ANOVA method in order to optimize the cutting parameters aiming to maximize tool life. Similarly, the training did not separate a test-set for network performance evaluation.

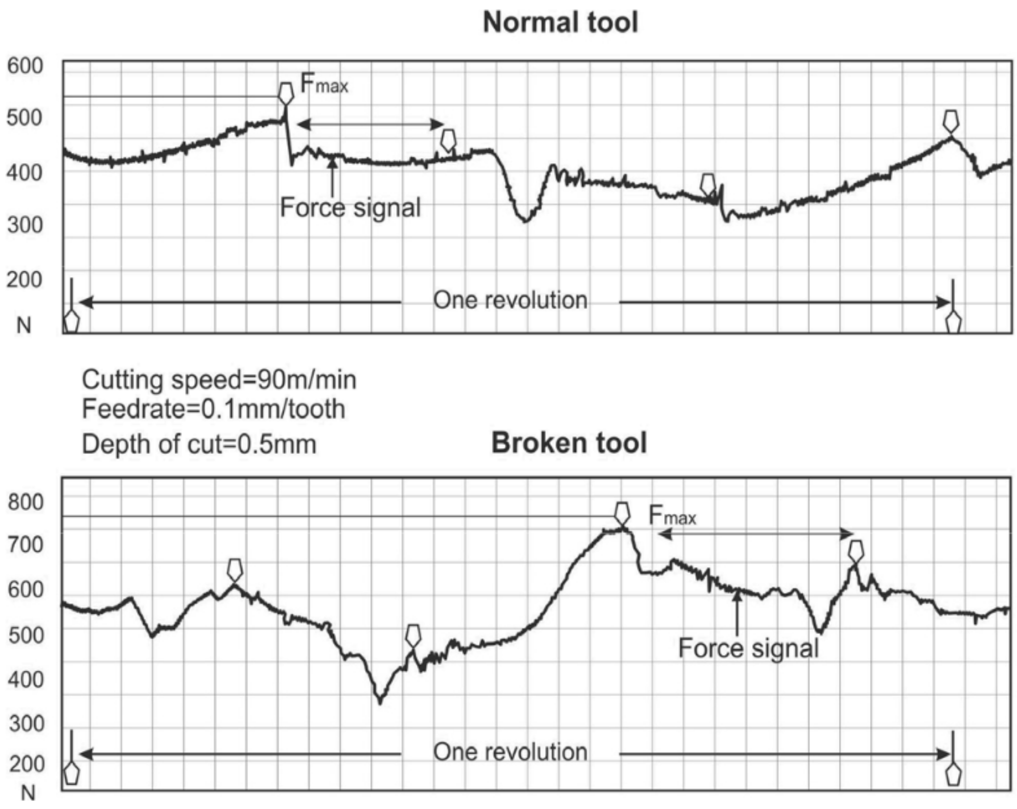
Faleh et al. [22] used the spindle power consumption to develop a TCM system for drilling. On this method, the stand-still power consumption is decremented when the spindle is moving towards the workpiece. On this way, the recorded signal can be interpreted as containing only the cutting power consumption. The experiment plan and signal analysis was also done using DoE.

[23] presented a method for TCM which uses an Adaptative Fuzzy Inference System for predicting tool wear values based on force indicators. After that, an ANN classifies them into two states: fresh or worn. The Fuzzy system looks at the peak force of each tooth in order to estimate the wear. The method reported relies on the fact that

a broken tooth will not receive load in the milling process while the next tooth will receive extra-load.

The reports in [23] and [24] showed how the forces applied to the table dynamometer are strongly related to the spindle position during the cutting process (see Figure 13). It is clear in Figure 13 how the signal changes as the tool gets worn.

Figure 13 – Force signal on a fresh and worn tool.



Source: [23]

The method developed by [25] uses an impedance layer printed on the tool surface in order to identify the wear state. The mechanism consists of measuring the impedance of the layer during machine operation. Once the wear level progresses, the paths printed in the tool edge are wiped out, changing its impedance value. Therefore, by measuring this entity it is possible to estimate the tool condition in real time. However, such method involves special tool manufacturing and delicate process setup, resulting in poor applicability for the industrial environment.

The tool wear prediction system for drilling developed in [26] employed a set of ANNs for the task. The data is distributed between the ANNs by addressing to a second network the samples which generated poor premature predictions on the first one. After addressing and training each model, a third model developed using Gaussian Distributions selects the target network based on the input under evaluation. So, the author claims that such method does not require tuning the network hyper-parameters. On this way, the procedure for training a dataset becomes simpler from an operator perspective.

[27] used a Bayesian system in order to produce a tool wear estimator through cutting force sensing. The paper examined multiple approaches to preprocess the data. A superior performance was achieved by using only the relevant features on the training step. Besides that, the paper also shows a superior performance of SVM in comparison to vanilla ANNs.

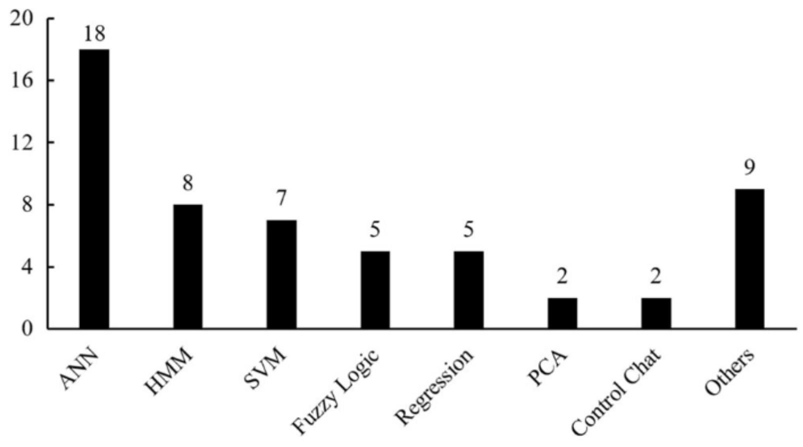
[28] and [14] applied a combination of fuzzy logic and ANNs for monitoring the tool condition. [28] applied AE on drilling while [14] applied cutting force on turning. The resulting system in both papers presented accuracy close to 100%.

The algorithms proposed on [29] and [30] use Hidden Markov Models in order to predict tool wear in milling process using force measurement. The force signal is pre-processed using wavelet transform. On [30], the approach also post-process the results of the HMM using a Gaussian distribution in order to estimate the remaining useful life of the tool. On this way, this system gives higher priority to the useful remaining time instead of the actual state of the cutters.

On the review presented in [3], the author listed the recent works regarding TCM for milling processes. It reports a drastic increase in methods using ML, mainly ANNs as shown in Figure 14.

One of the reasons pointed out by the author is the simplicity of ANNs when compared to other methods. The black-box perspective about the ANNs reduces the theoretical knowledge required for the task while simultaneously provides suitable generalization power. When concluding the text, the author states deep-learning techniques as one of the potential technologies for future works in the field of TCM for milling.

Figure 14 – Methods recently reported for TCM systems on Milling.



Source: [3]

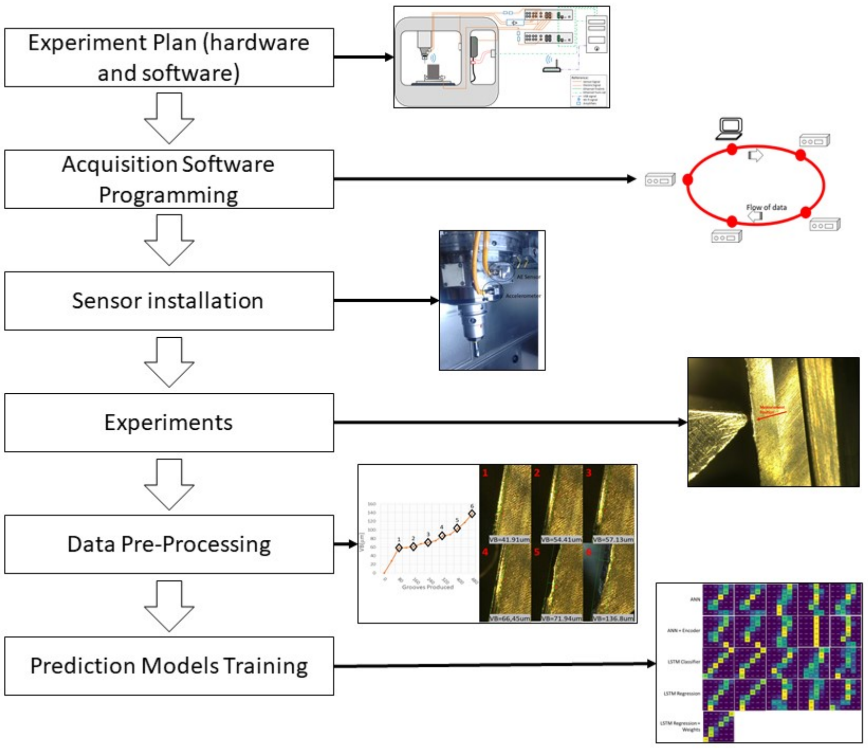
2.4. Solution Approach

The project goal is to develop a system for tool wear prediction for high precision milling. As previously described, the flank wear is the most observed type of wear in such processes. Furthermore, other types of wear usually occur only on final stages of flank wear – under normal cutting conditions. On this way, crater wear, chipping or any other wear type can be perceived as consequences of the flank wear development. Therefore, flank wear measurement will be adopted for tool wear estimation on this project.

The presented study uses DL techniques in order to interpret the acquired sensor signals. As the literature does not present massive information about the applicability of such models on tool wear prediction, multiple architectures were tested. ANNs, ANNs with auto-encoders, and LSTMs for classification and regression were employed. So, the project provides a comparison between those different approaches.

A set of experiments using different process parameter and direct measurement of tool flank wear were carried out in order to generate data for training the DL models. Once there is not a consensus about the best measurement strategy for the task, a multi-sensor system was installed in the CNC machine. Figure 15 lists the project steps for the proposed methodology.

Figure 15 – Project Steps.


Experiment Plan (hardware and software)
Acquisition Software Programming
Sensor installation
Experiments
Data Pre-Processing
Prediction Models Training
Flow of data
ANN
ANN + Encoder
LSTM Classifier
LSTM Regression
LSTM Regression + Weights

In order to compare the performance of each sensor employed on the experiments, all sensor signals were pre-processed using FFT. The generated spectrograms will serve as input for the prediction models. Each prediction model was trained using only one sensor data as input.

The study here presented explores all possible combinations between the chosen DL approaches and the pre-processed signals. So, by analyzing the performance of the trained models, it is possible identify the most suitable combination for the tool wear prediction task.

3. EXPERIMENT SETUP

Physical experiments like the ones reported in the scope of this project consume time and resources like machine tools and material. Therefore, the experiments were designed to also provide information for studying other phenomena.

So the data acquired from the experiment will be used for – besides tool wear prediction – tool breakage detection, chattering control, power consumption optimization, etc. Therefore, the experiment plan employs sensors which will not be further used for the prediction models. However, they are important for studying the mentioned problems.

The acquisition system records data provided from the following sensors: spindle force, spindle power consumption, acoustic emission, vibration, ultra-sonic emission and encoder position. Furthermore, the acquisition system must be projected to acquire all signals together. The following sub-chapters provide details about the resources applied on the experiments.

3.1. CNC Machine

The machine chosen for the scope of this work is the DMG MORI HSC 55 linear (see Figure 16), designed for high precision milling. It possesses a Heidenhain CNC system with 3 axes. The 10kW spindle rotates up to 28000 RPM, making it appropriate for steel cutting. The encoders responsible for the positioning system have $5\mu m$ accuracy. Table 1 exposes additional information about the machine.

Figure 16 – DMG MORI HSC55 linear CNC Machine.

Source: https://www.dmgmori.co.jp/en/top2/

Table 1 – CNC machine Technical information.

DMG MORI HSC 55 linear		
Component	**Description**	**Unit**
Axes	3 (X, Y, Z)	-
Working area	(X, Y Z) = (450, 350, 400)	mm
Max. Spindle Speed	28000	RPM
Max. Spindle Power	28	kW
Max. Spindle Torque	33	Nm
CNC system	HEIDENHAIN iTNC 530	-
Axes Encoders	HEIDENHAIN LC483	-
Axes Encoder accuracy	± 5	μm
Axes Encoder measuring step	100	nm

Source: https://www.dmgmori.co.jp/en/top2/

3.2. Sensors

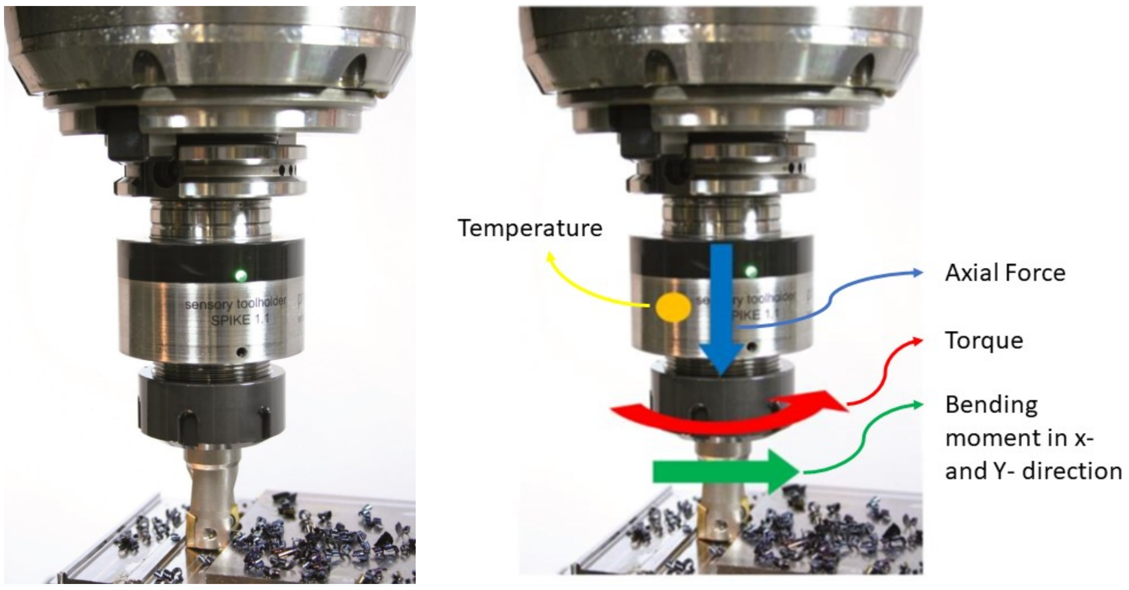
3.2.1. Cutting Force

In order to measure the cutting force regardless the tool center position (TCP), a device installed on the spindle is more appropriate than devices installed on the workpiece table. This is due to the fact that signals recorded from the spindle present more accurate information about the tool once they watch the process from a short and constant distance to the cutting point. Therefore, the Spike 1.2 tool holder was chosen for recording the cutting forces applied on the tool.

The Spike is a wireless smart tool holder which records flection, pressure and torsion applied to the tool (See Figure 17). Table 2 transcribes the main information about this device.

Although this tool holder can present important information about the process, its usage is undesired on the industrial environment. The main reasons are its cost, downtime on charging the remote unit and component force constraints. However, its usage is important once entities like shock and tool breakage can be better evaluated through force measurement.

Figure 17 – Sensory tool holder Spike 1.2 and recorded signals.



Source: https://www.pro-micron.de/spike/?lang=en

Table 2 – Spike Tool Holder Technical Information.

Spike Sensory Tool Holder 1.2		
Component	**Description**	**Unit**
Signals	Axial Force	N
	Torque	Nm
	Bending moment in X/Y	Nm
	Temperature	°C
Frequency response	1600	Hz
Axial Force measuring range	60	kN
Torque measuring range	400	Nm
Bending Moment measuring range	400	Nm
Axial Force resolution	<5	N
Torque resolution	<0,03	Nm
Bending Moment resolution	<0,03	Nm

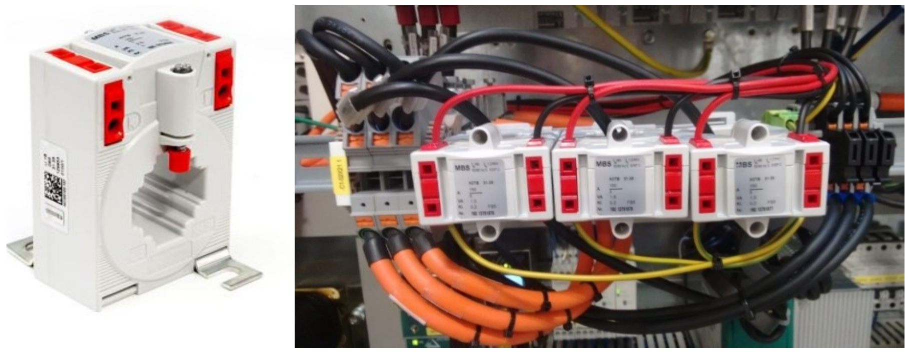
Source: https://www.pro-micron.de/spike/?lang=en

3.2.2. Spindle Power Consumption

Another factor directly correlated to the cutting force is the spindle power consumption. Once it is responsible for providing the required energy to the spindle and consequently to the tool, the electrical power consumed by the spindle is directly correlated to the cutting forces related to the machining process. Furthermore, power consumption measurement does not present most of the inconveniences caused by the direct force measurement like the ones listed in last section.

Due to high electrical current flow on the PLC cables, it is preferable to use a non-invasive method for the task. Therefore, the system uses 2 components. Primarily, a transformer reduces the amplitude of the electrical current of the cable (see Figure 18 and Table 3).

Figure 18 – MBS XCTB 31.35 transformer and its installation on the machine.



Source: https://mbs-ag.com/en/shop/xctb-31-35/

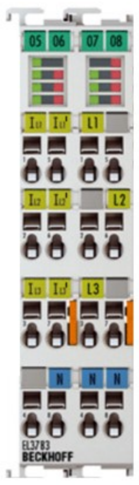
Table 3 – MBS XCTB 31.35 transformer technical Information.

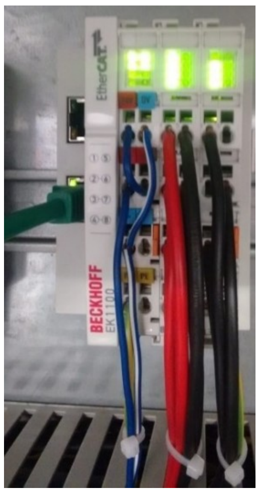
MBS XCTB 31.35 Current Transformer		
Component	**Description**	**Unit**
Max. Operating Voltage (U_{eff})	1,2	kV
Current Input range (I_{eff})	0-150	A
Current Output range (I_{eff})	0-5	A
Amplitude error (0,05-10kHz)	≤ 2	%
Amplitude error (10-20kHz)	≤ 3	%
Phase error (0,05-10kHz)	$\leq 2°$	-
Phase error (10-20kHz)	$\leq 3°$	-

Source: https://mbs-ag.com/en/shop/xctb-31-35/

After the current reduction, data is recorded using the Beckhoff EL3783 terminal. This device reads the three phase voltage and current, and records the power consumption. In order to communicate to the terminal, a Twin-cat client software is required for handling the communication with the device. The Twin-Cat terminal and its installation can be seen in Figure 19. The technical description is found in Table 4.

Figure 19 – Twin-Cat Power Monitor terminal and its installation.





Source:
https://infosys.beckhoff.com/english.php?content=../content/1033/el3783/2628174603.html&id=

Table 4 – Beckhoff EL3783 Terminal Technical Information.

Beckhoff EL3783 Power Monitor		
Component	**Description**	**Unit**
Number of inputs	3 x current, 3 x voltage	-
Measuring error	$< \pm 0.2\%$	-
Nominal Voltage Range	690	V_{rms}
Voltage Resolution	22.5	mV
Input Resistance Voltage circuit	1,5	$M\Omega$
Nominal Current Range	5	A_{rms}
Current Resolution	281	μA
Max. permitted Overvoltage	± 1270	V
Max. permitted Overcurrent	± 10A peak or $7A_{rms}$	

Source:
https://infosys.beckhoff.com/english.php?content=../content/1033/el3783/2628174603.html&id=

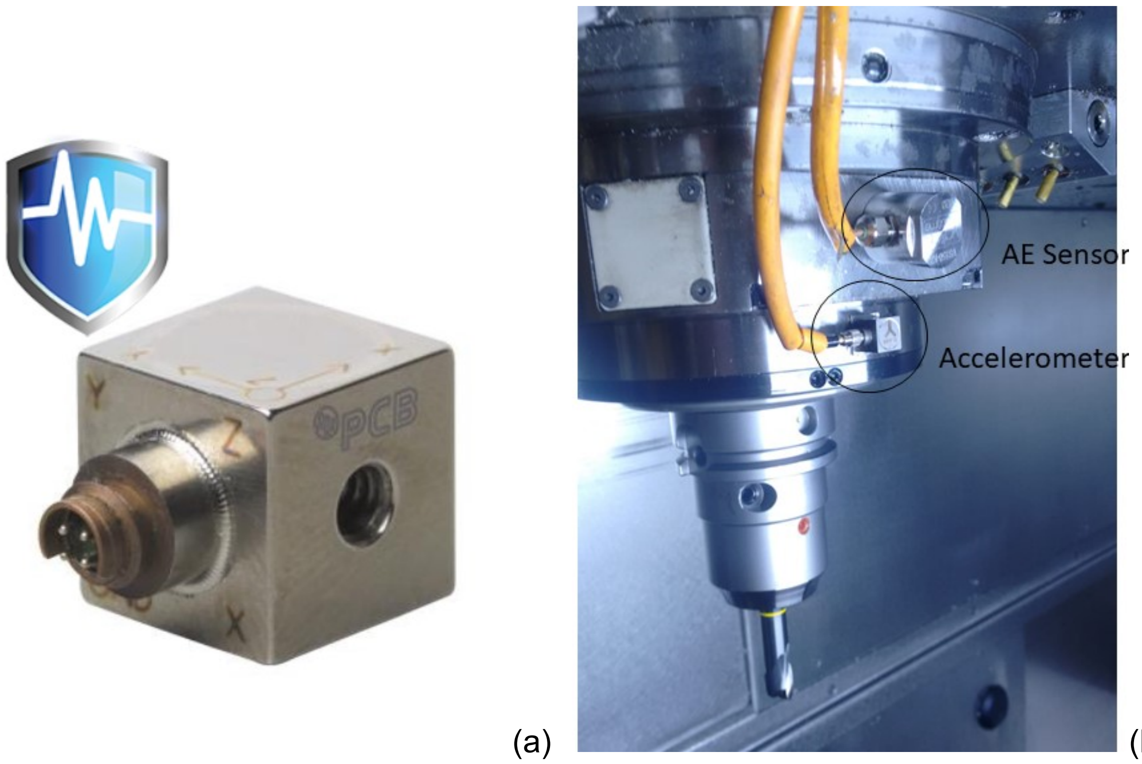
3.2.3. Vibration

The microscopical events provoked by removing material from the workpiece – shock, rubbing, abfraction, etc. – also generates vibration and noise. With the progress of tool deterioration, the behavior of those entities changes. So the experiments also observed the process vibration.

Unlike conventional methods, the accelerometer was installed in the spindle body instead of the workpiece. Besides the setup time, another reason for this choice, as previously mentioned in section 3.2.1, is the constant distance to the TCP. Sensors installed on the workpiece present more intense signals when the tool cuts material close to them. Fact which can be translated as measurement noise.

Table 5 presents the technical description of the accelerometer chosen for the experiment. The sensor designed for industrial operation (Figure 20) records the vibration on 3 axes (X, Y and Z) separately. The placement of the component can be seen in Figure 20(b).

Figure 20 - PCB 356B21 Accelerometer and its installation.



Source: https://www.pcb.com/products?model=356b21

Table 5 - PCB 356B21 accelerometer technical Information.

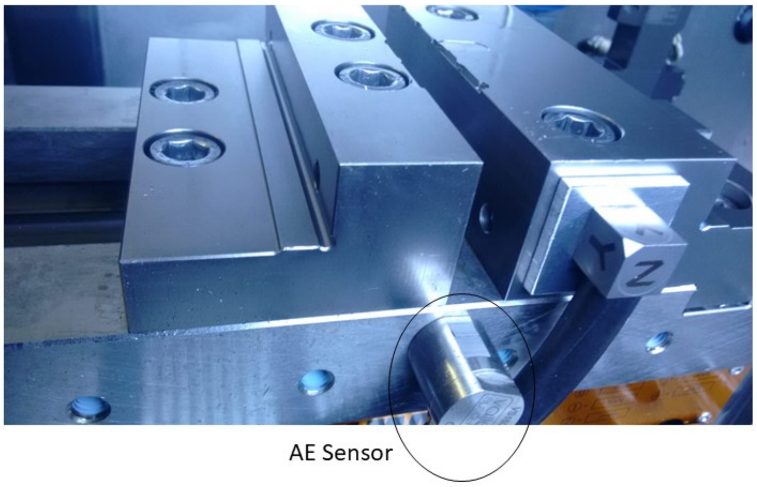
PCB 356B21 Accelerometer		
Component	**Description**	**Unit**
Axes	3 (X, Y, Z)	-
Frequency range (Y and Z)	2-10000	Hz
Frequency range (X)	2-7000	Hz
Measurement Range	± 4905	$\mathrm{m/s^2}$
Sensitivity	1,02	$\mathrm{mV/(m/s^2)}$
Broadband Resolution	0,04	$\mathrm{m/s^2}$
Output impedance	≤ 200	Ω

Source: https://www.pcb.com/products?model=356b21

3.2.4. Acoustic Emission

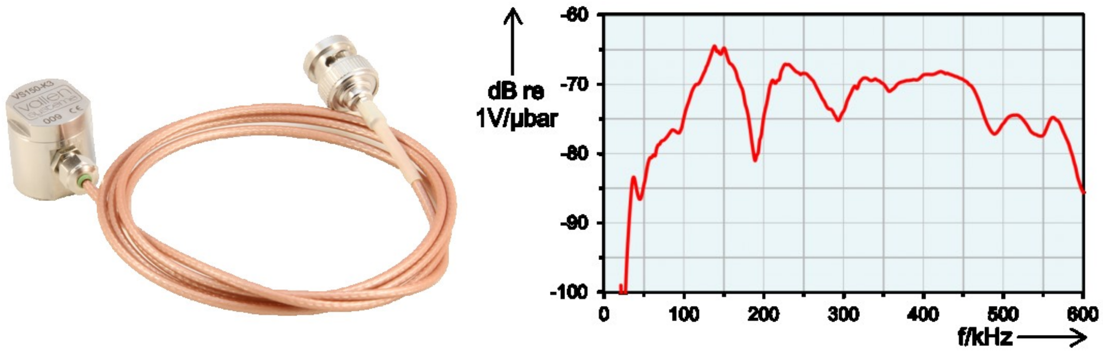
Along with vibration, acoustic emission (AE) is also generated during the cutting process. Past projects carried out the AE sensing on the workpiece side. In the referred project the acoustic emission will be measured in both workpiece clamp (Figure 21), and spindle body (see Figure 20). With the aim of correlating and comparing signals gathered from both of them during the process.

Figure 21 – AE sensor installation on workpiece.



The AE sensor chosen is the Vallen VS150K3. The sensor presents a strong non-linear behavior and high frequency response (Figure 22). Therefore, the data processing units must record information with a frequency $freq \geq 450KHz$ (see Table 6).

Figure 22 – Acoustic Emission Sensor and its response curve.



Source: https://www.vallen.de/sensors/watertight-sensors/vs150-k3/

Table 6 – Vallen VS150K3 Technical Information.

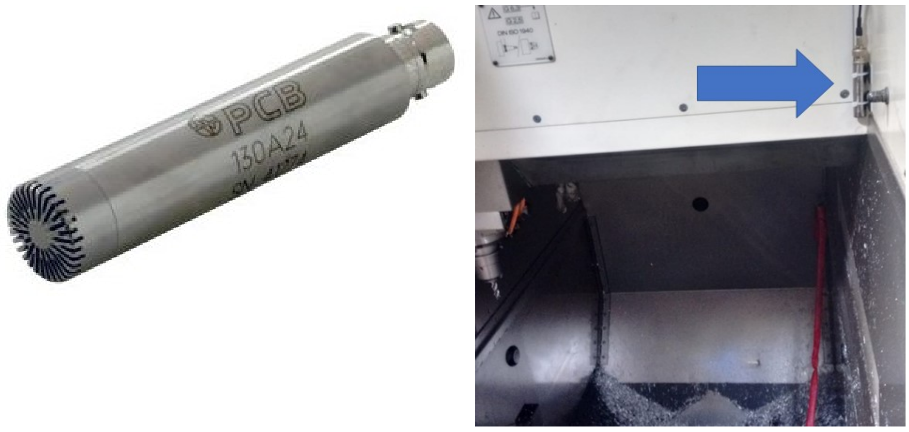
Vallen VS150K3 AE sensor		
Component	**Description**	**Unit**
Frequency Range	100-450	kHz
Capacity	450	pF

Source: https://www.vallen.de/sensors/watertight-sensors/vs150-k3/

3.2.5. Ultra-sonic Acoustic Emission

An ultra-sonic sensor was also installed in the machine in order to record the ultra-sonic emission generated in the process (Figure 23). The PCB 130A24 is an electret array microphone and its description is found in Table 7.

Figure 23 – PCB a30A24 microphone and its installation.



Source: https://www.pcb.com/products?m=130a24

Table 7 – PCB 130A24 microphone technical information.

PCB 130A24 microphone		
Component	**Description**	**Unit**
Frequency response ($\pm 3\mathrm{dB}$)	20-16000	Hz
Sensitivity ($\pm 3\mathrm{dB}$)	1	V/Pa
Inherent Noise	20	$\mu\mathrm{Pa}$
Output impedance	< 52	Ω

Source: https://www.pcb.com/products?m=130a24

3.2.6. Encoder Position

Another important source of information on the project is the Encoder positions of the machine. The actual position can present information about the speed, and positioning errors presented in the process. It is also a good source of information for filtering out the air-cut time of the path plan, i.e., the time when the machine is moving, but not cutting material.

The machine tool position encoders will be used in order to gather the axes positions. The signals are obtained by using a signal splitter on the encoder cables plugged on the machine PLCs. The DMG machine has an LC483 absolute positioning encoder. The acquisition system will record information from the 3 orthogonal axes and the spindle. See Figure 24 and Table 8.

Figure 24 – Heidenhain LC483 encoder.

Source: https://www.heidenhain.de/de_EN/products/linear-encoders/sealed-linear-encoders/for-numerically-controlled-machine-tools/lc-400-series/

Table 8 - Heidenhain LC483 encoder technical information.

Heidenhain LC483 linear Encoder		
Component	**Description**	**Unit**
Calculation time	≤ 5	μs
Encoder accuracy	± 5	μm
Encoder measuring step	100	nm

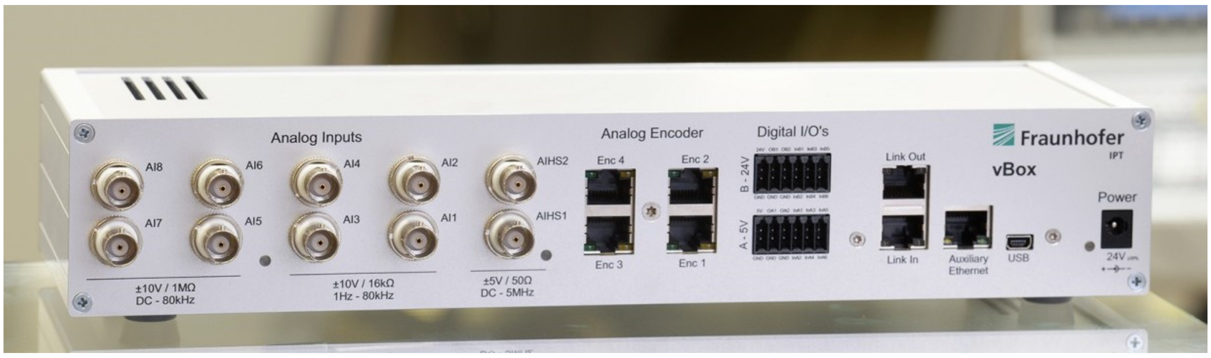
Source: https://www.heidenhain.de/de_EN/products/linear-encoders/sealed-linear-encoders/for-numerically-controlled-machine-tools/lc-400-series/

3.3. Fraunhofer V-Box

The V-Box is a DAQ developed by Fraunhofer for high frequency data acquisition. It is aimed to provide support for the usage of multiple sensors. Although, until the realization of this project, this hardware did not have a stable Client software. Therefore, the acquisition software for V-Box was also developed in this project as it will be reported in Chapter 4.

The V-BOX (Figure 25) has 10 voltage analog inputs. From those, 8 channels are tuned for 80kHz while 2 channels for high speed data acquisition up to 5MHz. This last – called High Speed Analog Input (AIHS) – records data by using an integrated FFT hardware followed by an Analog-Digital converter. On this way, the data provided from this channel is encrypted in a 32 bits integer where the first 16 bits represent the amplitude and the last 16 bits represent the frequency coefficient.

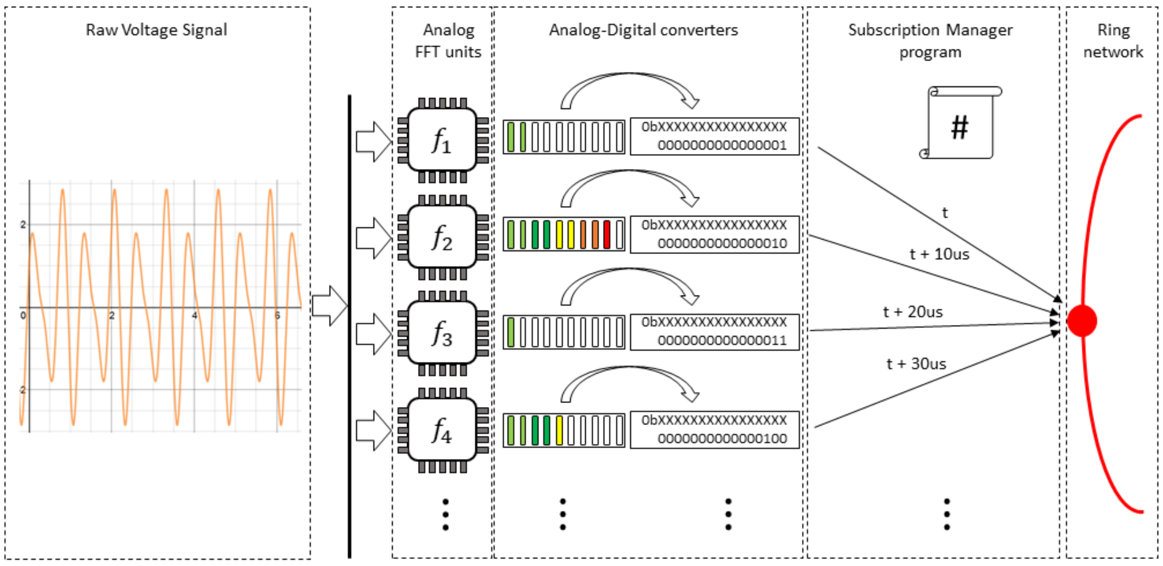
Figure 25 – Fraunhofer V-Box.



Source: https://www.ipt.fraunhofer.de/de/kompetenzen/Produktionsmaschinen/praezisionstechnik-und-kunststoffreplikation/vbox.html

So, data coming from the sensor feeds the FFT units, which will present on its output the analog band amplitude of the signal. This amplitude bus is then converted and encrypted with its corresponding band as explained above. After that, an algorithm loops through this new generated values and sends them to the network sequentially at 100kHz. See Figure 26.

Figure 26 – High speed data processing representation on V-BOX.

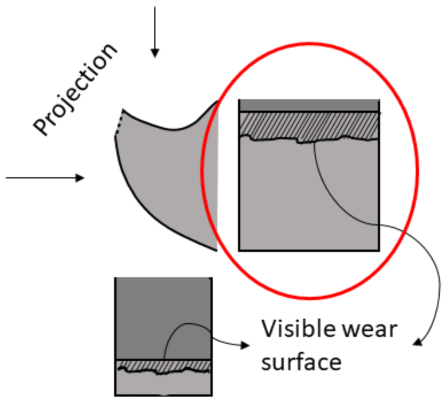

Raw Voltage Signal
Analog FFT units
Analog-Digital converters
Subscription Manager program
Ring network
f_1
f_2
f_3
f_4
0bXXXXXXXXXXXXXXXX 0000000000000001
0bXXXXXXXXXXXXXXXX 0000000000000010
0bXXXXXXXXXXXXXXXX 0000000000000011
0bXXXXXXXXXXXXXXXX 0000000000000100
#
t
t + 10us
t + 20us
t + 30us

3.4. Tool Wear Measurement

As explained in [31], the flank wear is one of the most appropriate wear types to indicate the actual state of the tool. Firstly, because it presents a gradual progress along the usage of the milling tool. Secondly because other types of wear like crater and chipping can be understood as consequences of flank wear progress under normal cutting conditions. In other words, crater and chipping happen on the tool only after the cutter edge achieves an advanced level of deterioration.

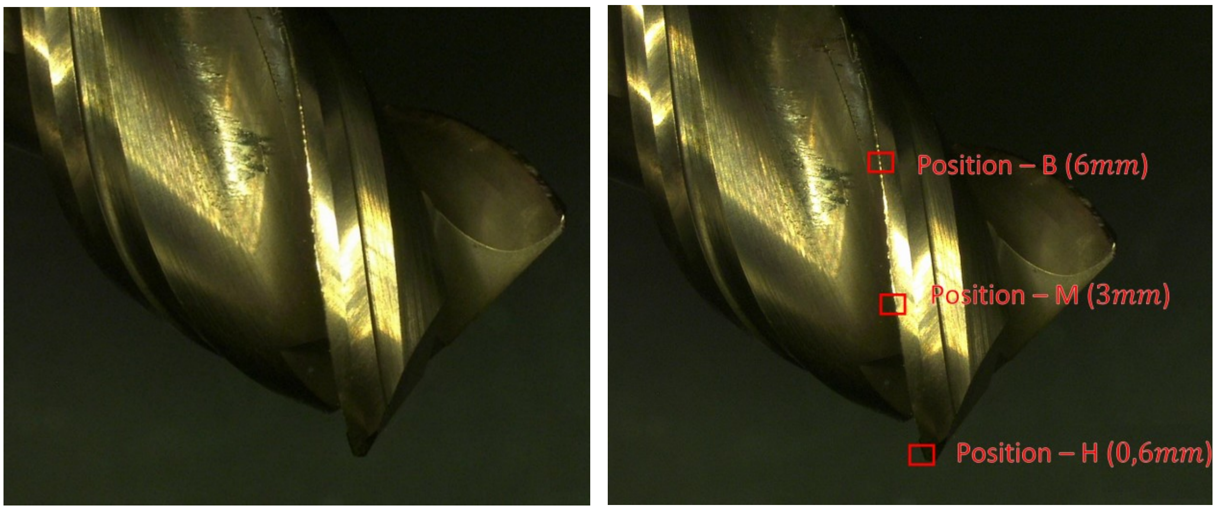
Therefore, the flank wear measurement will be adopted in order to assess the actual condition of the milling tool. The procedure adopted observes the wear surface length on the edge of the tool parallel to the cutting surface. See Figure 27. The main reason for such choice resides on the fact that it is possible to determine precisely the measurement positions and edge loss when accessing the parallel view of the surface. Furthermore, the external surface of the tool has extra marks which helps the measurement procedure once the tool edge starts to lose material.

Figure 27 – Different perspectives for accessing the flank wear surface.



For labeling the data, tool flank wear was measured using Keyence VHX-500F electronic microscope. As it will be presented in section 3.6, the employed tool has an helicoidal geometry (see Figure 28). Therefore, the procedure involves some additional steps in order to determine the measurement position.

Figure 28 – Tool on microscope and measurement positions.

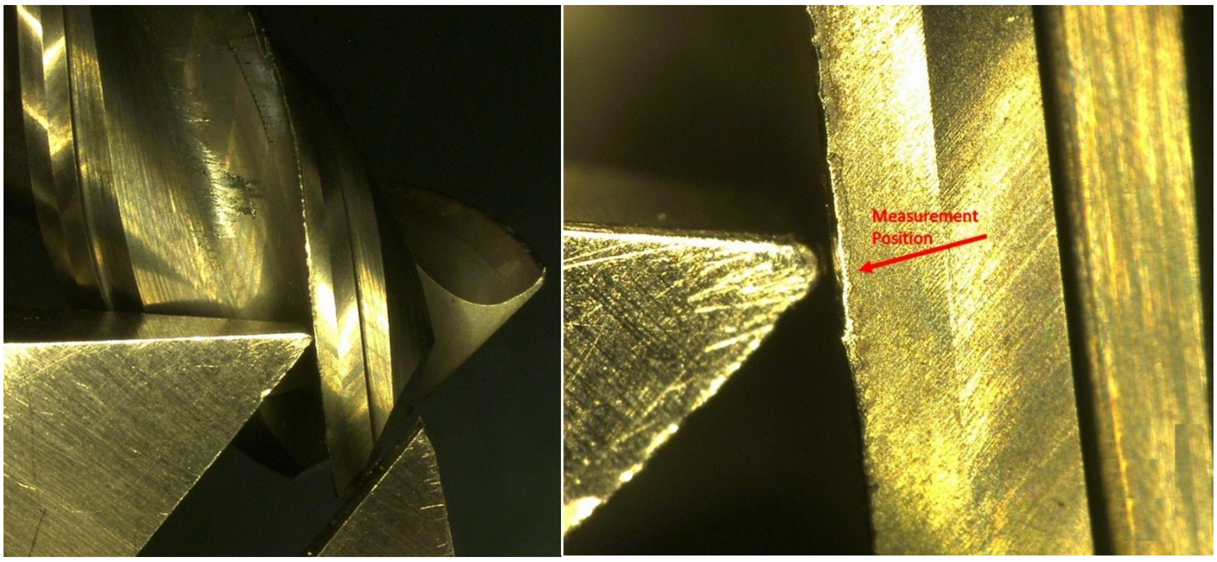


On each tooth, 3 measurement points are accessed: the first one close to the tool tip (called Head or H), approximately 0,6mm from the visible tooth tip; the second at a distance of 3mm from the tip (Middle or M); and the third at 6mm (Base or B). See Figure 28. On this way, if a tool has 3 tooth, the flank wear is measured in 9 points at total. As the literature recommends, the adopted tool wear is the maximum value from those 9 measurement points.

For determining the two last positions, a pachymeter is used in order to measure the distance from the tool tip to the stablished measurement points (See

Figure 29). This step guarantees that the flank wear will always be accessed on the same position of the tool during all its lifetime.

Figure 29 – Pachymeter method for finding measurement position.



At last, the flank wear is measured by looking at the worn surface length. Another important element is the light reflection on the measurement point. Every time when accessing the required position, the light emitted from the microscope must reflect besides the referred position. This adjustment has the purpose of fixing the tool angle for all measurements, once a different angles can distort the worn surface length. See Figure 30.

To validate the described procedure, 5 test measurements were carried out in a trashed 3 teeth helicoidal tool similar to the ones used on this project. Table 9 shows the resulting measurement values and metrics.

Figure 30 – Flank wear measurement and light positioning.

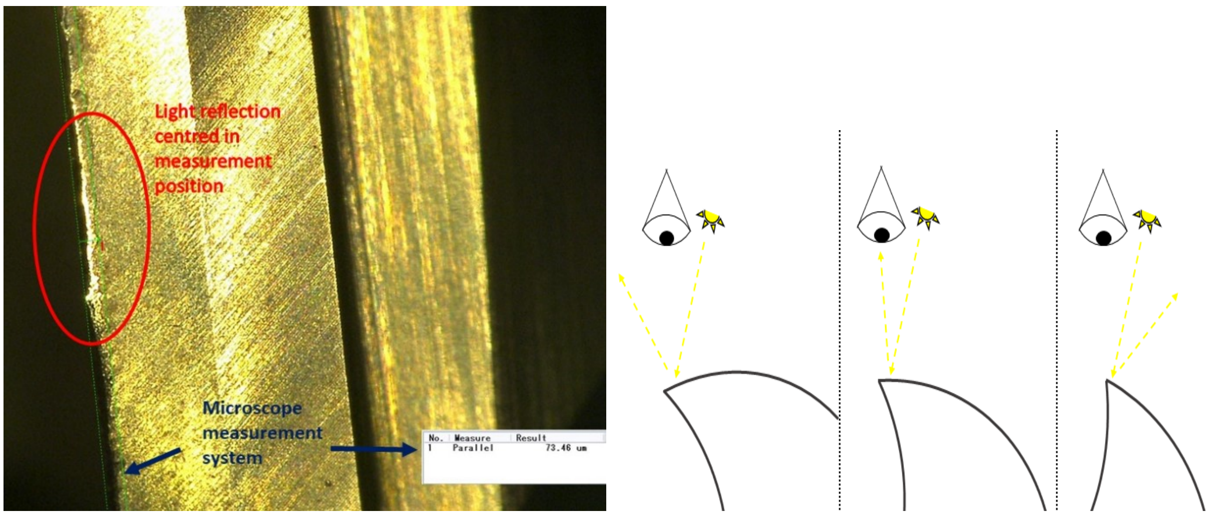

Light reflection centred in measurement position
Microscope measurement system
No. Measure Result
1 Parallel 73.46 um

Table 9 –Flank Wear Test Measurement Results (in μm).

Meas. Nr.	T1B	T2B	T3B	T1M	T2M	T3M	T1H	T2H	T3H	Max
1	86	149	105	78	114	119	62	127	121	149
2	81	124	88	80	130	110	77	137	149	149
3	84	110	92	73	120	132	66	113	135	135
4	98	96	87	78	134	122	89	139	139	139
5	85	102	89	63	143	123	67	140	126	143
Mean	86.8	116.2	92.2	74.4	128.2	121.2	72.2	131.2	134	**143**
Std. Dev.	5.84	18.89	6.62	6.15	10.24	7.08	9.74	10.21	9.84	**5.51**

The estimated measurement deviation is $\pm 6\mathrm{\mu m}$. Given the maximum allowable flank wear as VB=110$\mathrm{\mu m}$, the error corresponds to 5,46% of the tool wear range. From now on, this value will be adopted as the standard error for the described measurement procedure.

3.5. Hardware Setup

Once all sensors cannot be processed by a single unit due to hardware property limitations – different signal transmission, proprietary architectures – the sensors will be handled by 3 different data frameworks. These frameworks will run in parallel and time synchronization will be handled afterwards. Once the goal of this project is to analyze each signal separately, the time synchronization does not represent a problem.

Therefore, Spike hardware and software solutions will record the force signals gathered by the tool holder. The Power Monitor signals will be processed using a Twin-Cat client software. The remaining sensor signals will be handled using 2 V-Boxes, once only one device cannot stand all data flow in the required frequency.

Figure 31 shows the complete hardware setup for the experiments. Table 10 lists the connection to each V-Box, once they cannot be distinguished on Figure 31. In order to acquire all relevant information generated by the listed sensors, the AE signals will be recorded using the high speed channels of the V-Box. The configuration chosen records data up to 500kHz. The remaining sensors are recorded at 50kHz.

Figure 31 – Complete Hardware setup of the experiment.

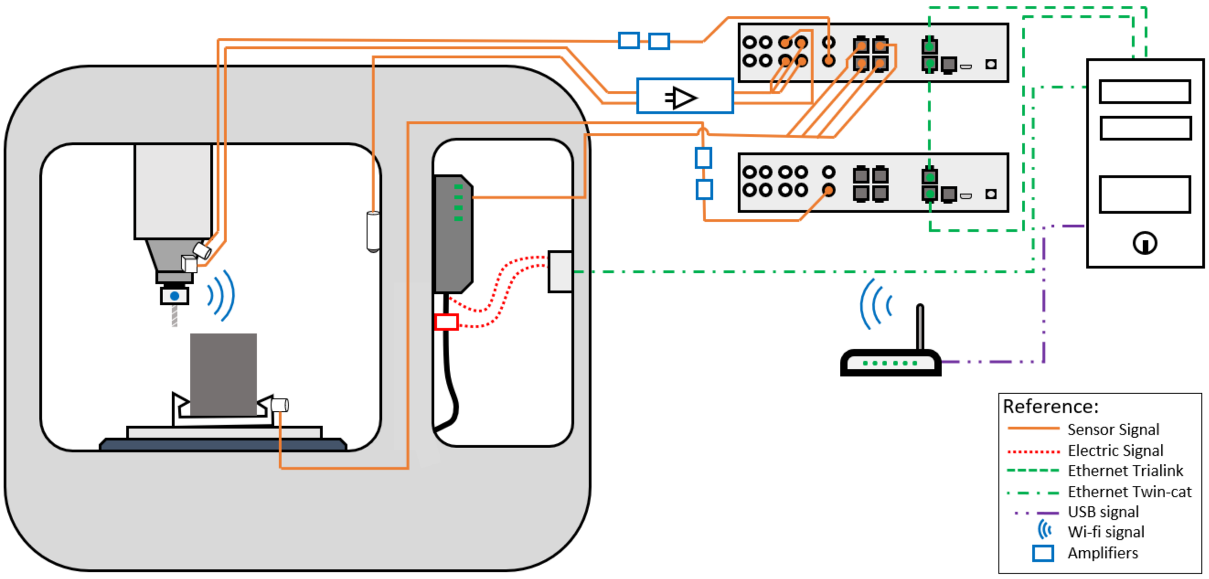


Table 10 – Sensors connected to the V-Boxes.

Source	Amplifier	Destination
AE-Spindle	DCPL2 + AEP5H (Vallen)	AIHS1-VBOX1
AE-Workpiece	DCPL2 + AEP5H (Vallen)	AIHS1-VBOX2
Accelerometer X	482C (Piezotronics)	AI1-VBOX1
Accelerometer Y	482C (Piezotronics)	AI2-VBOX1
Accelerometer Z	482C (Piezotronics)	AI3-VBOX1
Microphone	482C (Piezotronics)	AI4-VBOX1
Encoder X		Enc1-VBOX1
Encoder Y		Enc2-VBOX1
Encoder Z		Enc3-VBOX1
Encoder Spindle		Enc4-VBOX1

3.6. Process Setup

Once the hardware configuration for the experiments is set up, it is necessary to define the process characteristics of the experiments. The mechanical aspects of this project will be handled in a similar way as presented in [32].

In [32], a simpler hardware and software layout was used for the milling experiments. The main goal was to create a dataset for future studying the tool flank wear phenomena. So, the project was focused on studying the mechanical properties of the tool wear in order to determine the best process parameters for a plane overview of the phenomena.

Therefore, the project provides valuable information about the CNC parameter determination, path planning and material used to approach the problem. Serving as a reference guide for the experiment plan performed on this work.

The tool chosen for the experiments is the Pro-Steel solid carbide roughing end mill HPC 8mm. Further information about the tool is presented in Table 11. The tool has 3 teeth and it is designed for rough or finishing milling using high spindle speed. The workpiece block is a steel plastic mold 40CrMnNiMo8-6-4.

Table 11 – Solid Carbide HPC 8mm tool technical description.

Tool Description	
Name	Pro-Steel solid carbide roughing end mill
Article Nr.	2024148
Cutter Diameter	8mm
Cutting Length	19mm
Overall Length	63mm
Number of Teeth	3

The path planning consists of successive groove making. The cutting width used is 5mm and the cutting depth is 8mm as recommended by the tool manufacturer. Considering the workpiece size, 20 grooves can be produced in each depth level (see Figure 32). After milling 2 complete levels – 40 grooves – the tool is taken to the microscope for the measurement of the flank wear (VB) according to the procedure described in section 3.4.

In order to determine the optimum process parameters, [32] performed a battery of trial experiments. Starting from the recommended parameters, those tests consisted in completely use a tool in order to determine the flank wear curve for the tested configuration. Then, the parameters were tune until a good combination for tool wear progress was found. The trial tests performed for the referred tool are presented in Table 12.

Figure 32 – Superior view of the process path planning.

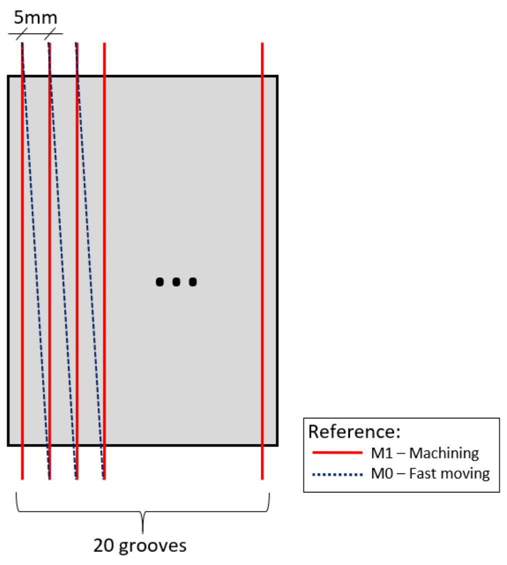


Table 12 – Trial Experiments for process parameter determination.

Trial Nr.	Spindle speed [1/min]	Feed rate [mm/min]	Cutting width[mm]	Cutting depth[mm]	Life time [s]	VB [μm] (flank wear)
1	7440	1300	6	8	185	Broken
2	7440	1160	6	8	83	Broken
3	**7440**	**1160**	**5**	**8**	**>3818**	**96**
4	**6760**	**1160**	**4**	**8**	**>7055**	**90**

Source: [32]

Based in the trial experiments presented in Table 12, the experiment plan will consist of using 4 tools and 2 different parameter combinations. The process parameter plan for the experiments is shown in Table 13.

Table 13 – Experiment Plan.

Ex. ID	Spindle speed [1/min]	Feed rate [mm/min]	Cutting width[mm]	Cutting depth[mm]	$VB_{max}[\mu m]$
11	6760	1160	5	8	110
12	6760	1160	5	8	110
21	7440	1160	5	8	110
22	7440	1160	5	8	110

4. ACQUISITION SYSTEM PROGRAMMING

For recording data coming from the sensor system listed in Chapter 3, an acquisition system must be programmed. Once Fraunhofer V-Box, the Beckhoff Twin-cat and Spike communicate using different architectures, the approach consists in running those 3 acquisition modules in parallel.

The main element of this system is the Tama Client – as named by the author. This system is responsible for communicating and storing the signals coming from the Fraunhofer V-Box. Once most of the sensors are connected to the V-Box, this device has higher priority for the reported experiment plan.

On the following sub-chapters it will be explained the project and implementation of the Tama Client system using C# programming language. The remaining modules required for the experiments were programmed by other team members. So they will not be reported in the scope of this document.

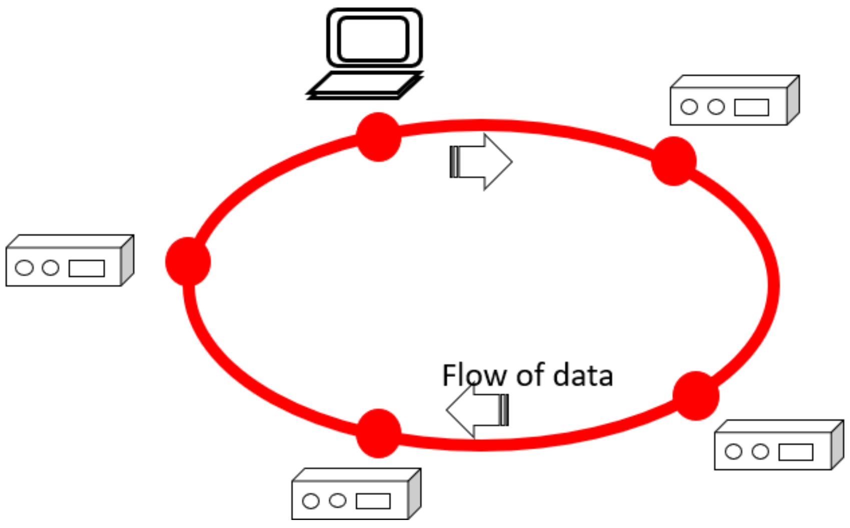
4.1. Trialink Network

Tama Client is a Software Solution to perform data acquisition on Trialink Networks. Even though it was initially projected to connect to Fraunhofer V-BOX, the system is able to connect and acquire real-time data from any device which can communicate via the Trialink Network schema [33].

Once the client software does not know the devices connected to the network beforehand, the client software must be able to identify the device configuration of the accessed network automatically. On this way, the operation avoids reprogramming the software every time a new set of components must be accessed.

As explained in [33] the Trialink network is a ring network designed for real time communication (see Figure 33). It is mainly used for synchronizing operation between drives like axis motion control units in a CNC machine. Therefore, the protocol is extensively used in order to program software systems for metal cutting machines mainly in high precision field.

Figure 33 - Example of Trialink network with 5 devices connected.



All the devices on this type of network have 2 Ethernet entries: a link-in, and a link-out. Therefore, the link-out of a device k is connected to the link-in of the device k+1, the link-out of device k+1 to the link-in of device k+2, and so on. Until the last device is reached, which is connected to the link-in of the first device closing the ring.

Once all messages have time constraints to reach their destination, it is impossible to prevent data loss. Therefore, there are situations when overloads on the communication flow of data bottlenecks the system causing Queue Overflows, or simply communication timeouts on the network.

Each device on the ring behaves like a station node, receiving and sending data as requested. Internally, the hierarchy of a device is organized in a tree perspective, where the root node corresponds to the device and all the leaf nodes correspond to readable/writable registers. The structure on the middle of this tree is strongly dependent on the type of device and cannot be determined beforehand.

For example, a DAQ system like V-Box will have an internal tree consisting of Analog-Inputs, Encoders, etc. and under each node, the exact number of inputs or outputs the DAQ offers. On the other hand, a Motor Drive for Motion control will possess a node which points to Writable controller parameters, and other register providing the readable position, speed, acceleration, positioning error, etc.

Each readable register inside this tree can be acquired using a Subscription/Publisher schema, which is used on the Tama Client system in order to acquire data from the devices. All the Subscription Mechanism is abstracted to the user and implemented internally on the client module.

So, once the system connects to the Trialink Network, it requests all the devices on the link for their respective Register Tree, listing all the readable registers of the system. On the GUI, the devices are also organized in a tree perspective, so the user must have a basic knowledge about the internal structure of the target devices in order to select the right registers for the acquisition.

4.2. System Requirements and Software Project

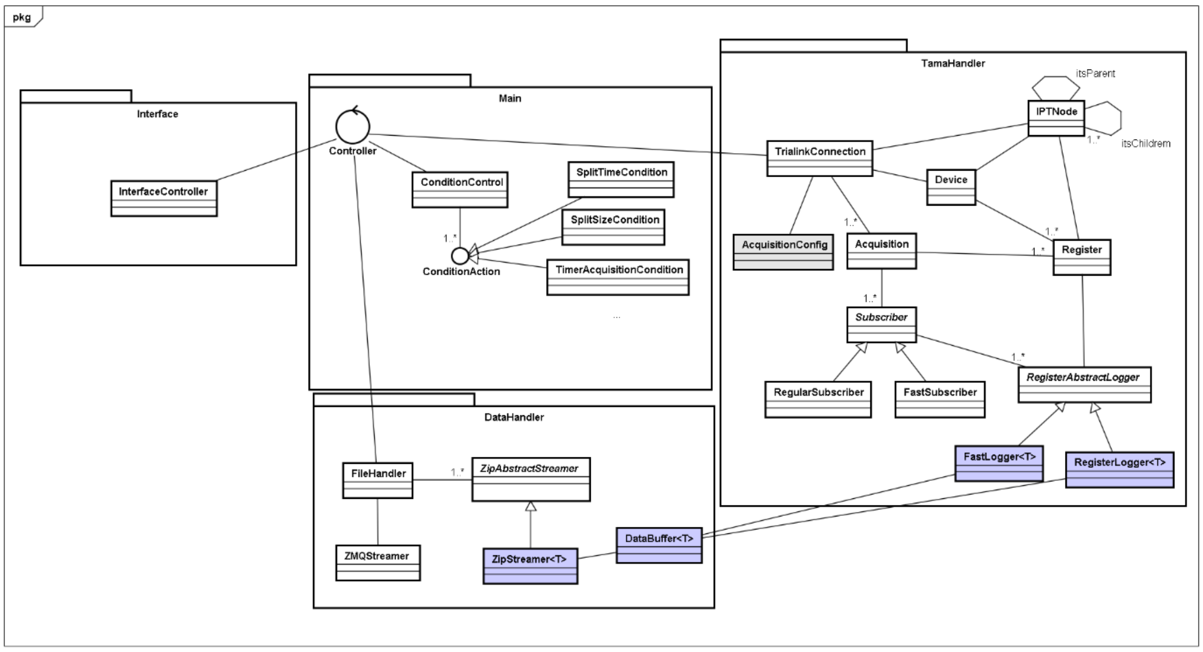
In order to assist the software implementation, a software project was developed. At first, a document specifying the functionalities of the software was produced. This document includes functional and not-functional requirements. The purpose of this first step was to develop a guideline to further planning of the class diagram to be implemented in the Tama Client software. The document can be seen in Appendix A.

Based on this document, a class diagram was developed. The resulting project paid attention to fulfill all mandatory functionalities while preserving important properties like:

- Modularity: the tasks on the project were distributed between classes with specific functionalities. So, for instance, if a change in how the module saves data is required, only a specific region of the code must be edited;
- Scalability: The software ensures high performance in simple and complex tasks, allowing a fast way of programming improvements or new functionalities;
- Robustness: the project controls a wide variety of possible problems which can happen during operation. Therefore, it possesses protective mechanisms to assure resilience and minimize losses while keeping the operation active.

Figure 34 shows the resulting class diagram of the Tama Client. The attributes and methods were ignored in order to make the diagram visualization clear. A more detailed explanation about its components, interface appearance and implementation steps can be accessed in Appendix B.

Figure 34 - Class Diagram for Tama Client Software.



4.3. Implementation

This sub-chapter will explain some of the functionalities of the software. The overall software implementation was initially separated in steps, and coded partially. Starting from the Main package.

The module is projected to acquire in the maximum frequency of the V-BOX, which is 100kHz. Therefore, if the quantity of registers is too big, the network cannot stand all incoming flow of data. In order to avoid this problem, the system was implemented to support multiple acquisition sampling rates.

By doing so, high frequency sensors can be processed on maximum frequency while sensors with lower dynamic will be recorded using smaller frequency rates. This approach releases the network flow of data, minimizing the chances of data loss during acquisition.

Therefore, the back-end and front-end were programmed to stand multiple acquisition. Although each acquisition has a different sampling rate, all register data is synchronized using the same reference timestamp system.

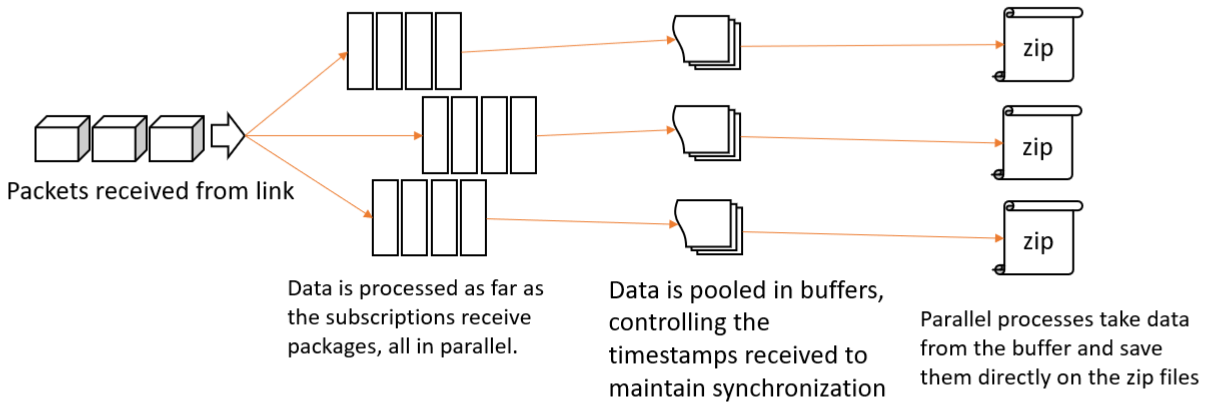
Once the communication on the networks follows a real time standard, the processes (threads) on the Tama Client have short time constraint to process the data.

The system was implemented in order to maintain the minimum amount of buffered values possible.

This is another reason why the network readers and file writers work in parallel. On the same meaning, the system has a different data processing unit – readers and writers, also called data pipeline – working independently for each register recorded.

In order to accomplish the task of minimum data type conversion (NF5.2 in Appendix A), the pipeline was kept generic. On this way, each register is processed using the specific variable type which comes from the ring network. Figure 35 presents a representation of the data pipeline.

Figure 35 – Representation of data processed in the pipeline.


zip
zip
zip
Packets received from link
Data is processed as far as the subscriptions receive packages, all in parallel.
Data is pooled in buffers, controlling the timestamps received to maintain synchronization
Parallel processes take data from the buffer and save them directly on the zip files

In order to synchronize the start, stop and normal operation, all the threaded instances possess state machines. The function of the state machines is to: provide a status control mechanism; ease data exchange between different threads; synchronize the timestamp on the beginning and end of acquisition file; and provide information to the GUI.

Another important task of the state machines is to provide status information to the Controller (NF5.3 on Appendix A). So during operation, the controller accesses all threaded instances and check if the system is running without problems. In the cases when some failure happens, the Controller is notified. After that, the Controller thread forces all machine states to go to failure state, preventing overflows and minimizing information loss. When possible, the Controller tries to restart the acquisition.

For saving the data, Tama Client uses the Deflate Stream C# library. Therefore all files are saved in zip format. Each zip file contains an array for one register. In order to identify the files, an XML header file is generated detailing the information saved in each deflate file.

However, those files cannot be read immediately after saved as requested by F9 (Appendix A). In order to fulfill this functionality, after a file is closed, the Tama Client invokes a process to convert all zip files to one HDF file [34]. This process runs independently from the acquisition and does not damage the software performance. After the conversion is finished, the file can be accessed and read normally using – for instance – HDF Viewer software.

This approach was chosen mainly because the C# library for HDF is unstable. In past versions of the Trialink client software, the system suffered frequent crashes due to lack of performance of the referred library. Dealing with deflate stream in runtime proved to be more efficient in comparison with directly persisting data using HDF format.

4.4. Performance

As a result, the software can achieve all listed requirements when performing acquisition on the ring network. A wide battery of tests was employed in order to prove the performance of the software, which fulfilled the stablished functionalities listed in Appendix A. Those tests will not be reported since they are not the aim of the incurrent document.

On Figure 36, it is shown the memory and process resource consumption of the Tama Client performing an acquisition on 9 registers at a frequency of 100kHz. It can be noticed that the number of data enqueued is fewer when compared to the old version of the software. This result was achieved by the parallel pipeline procedure to process each register on the acquisition.

By using multiple sampling rates, it was possible to increase the number of registers gathered in the acquisition. Figure 37 shows an acquisition file with 13 registers gathered from 2 V-Boxes using the Tama Client. By decreasing the sampling rate it is possible to increase even more the number of registers selected.

Figure 36 – PC resource consumption in old and new version of the Client.

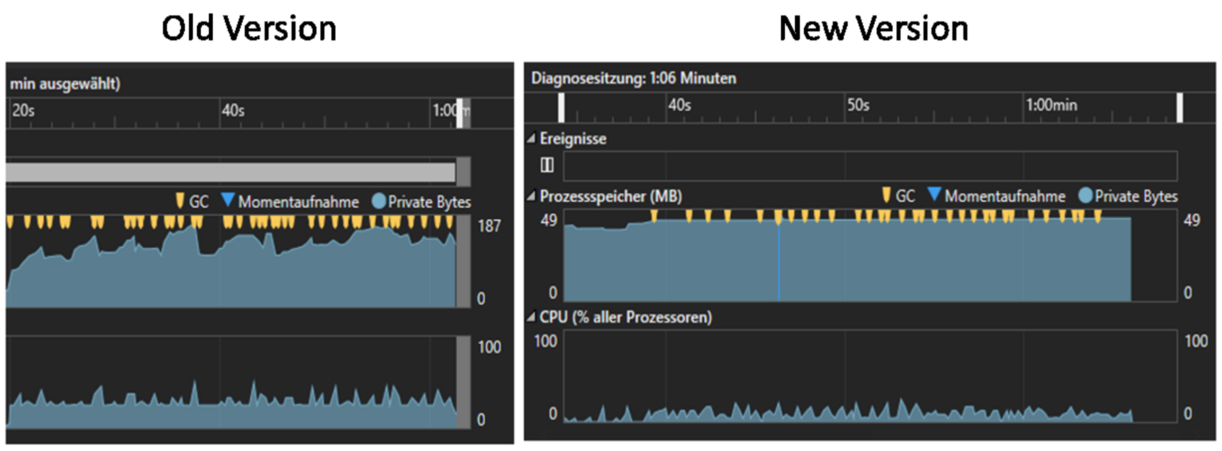
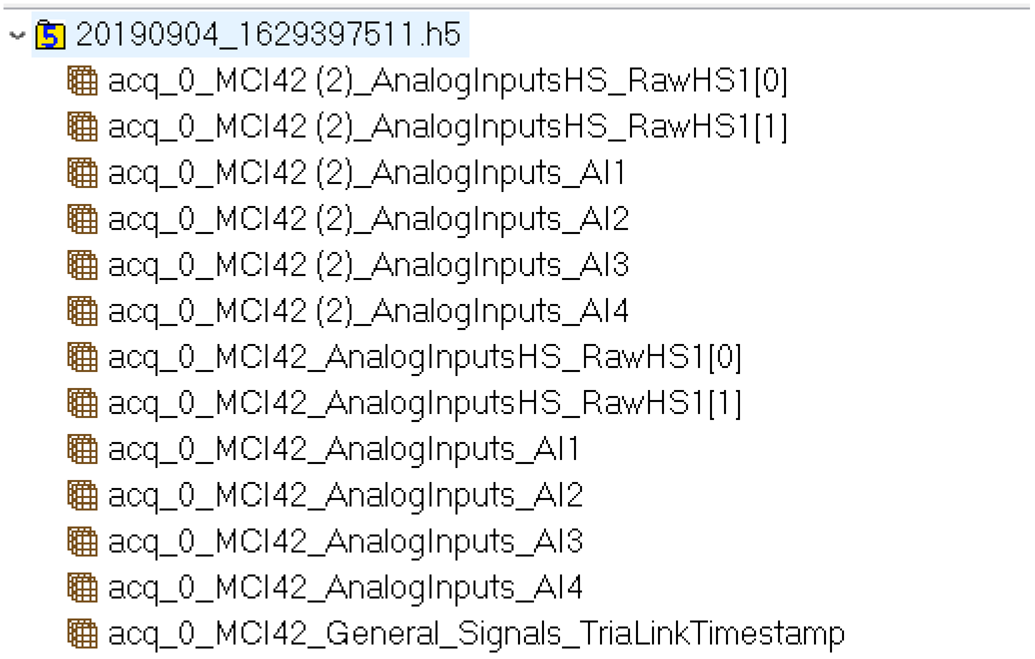


Figure 37 – HDF file generated by Tama Client with 13 registers.

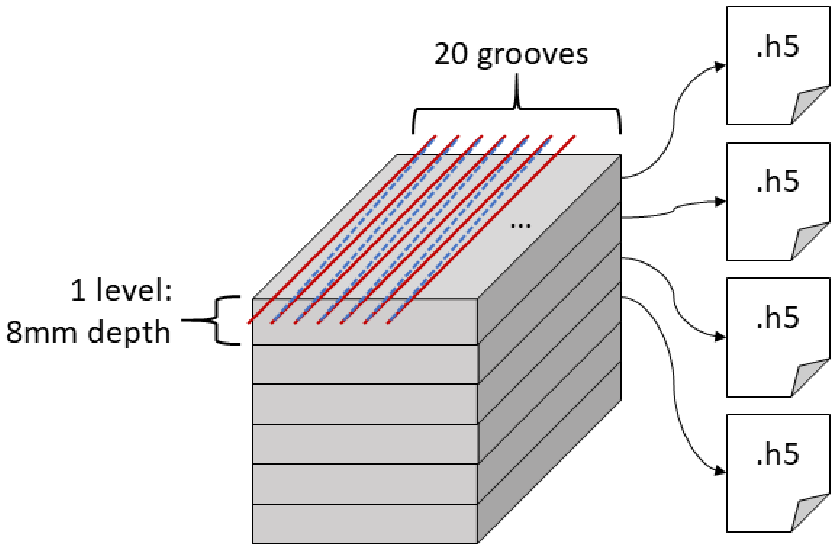


20190904_1629397511.h5
- acq_0_MCI42 (2)_AnalogInputsHS_RawHS1[0]
- acq_0_MCI42 (2)_AnalogInputsHS_RawHS1[1]
- acq_0_MCI42 (2)_AnalogInputs_AI1
- acq_0_MCI42 (2)_AnalogInputs_AI2
- acq_0_MCI42 (2)_AnalogInputs_AI3
- acq_0_MCI42 (2)_AnalogInputs_AI4
- acq_0_MCI42_AnalogInputsHS_RawHS1[0]
- acq_0_MCI42_AnalogInputsHS_RawHS1[1]
- acq_0_MCI42_AnalogInputs_AI1
- acq_0_MCI42_AnalogInputs_AI2
- acq_0_MCI42_AnalogInputs_AI3
- acq_0_MCI42_AnalogInputs_AI4
- acq_0_MCI42_General_Signals_TriaLinkTimestamp

5. DATA PRE-PROCESSING

This section is destined to explain the data information extraction from the acquisition files. The experiments were performed such that each file contains information about the material removed from one level of the workpiece, summing 20 grooves per file. See Figure 38. After 2 complete levels of material were removed, the tool was taken to the microscope for the flank wear measurement.

Figure 38 – Representation of grooves and levels machined on the experiments.



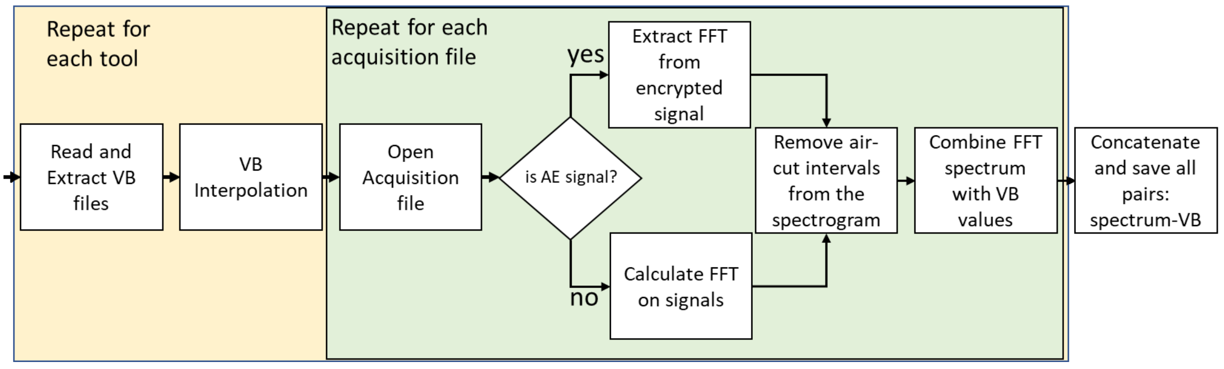
Therefore, the pre-processing module is responsible for extracting information gathered from the signals recorded during acquisition, as well as extracting information about the flank wear measurements performed in the microscope in order to label the data.

On Figure 39 it is presented the sequence of operations performed in the pre-processing module in order to extract information from the acquisition signals and flank wear (VB) measurement files. The result of the module is the datasets which will be used for training the prediction models in the next step of the project.

Besides that, the algorithm processes information from each signal separately. Therefore, the procedure shown in Figure 39 is repeated for each of the signals gathered on the experiments.

At first, the algorithm analyzes the flank wear measurement files for one tool. Then, from the flank wear measurement points acquired along the usage of the tool, the algorithm can estimate the flank wear curve for the referred tool. In order to do so, a polynomial interpolation is employed.

Figure 39 – Graph of operation sequence for data pre-processing.


Repeat for each tool
Repeat for each acquisition file
Read and Extract VB files
VB Interpolation
Open Acquisition file
is AE signal?
yes
no
Extract FFT from encrypted signal
Calculate FFT on signals
Remove air-cut intervals from the spectrogram
Combine FFT spectrum with VB values
Concatenate and save all pairs: spectrum-VB

After this step, the target signals are read from the acquisition files in order to extract the input array for the dataset. As previously mentioned, each dataset contains one sensor signal as inputs, and all signals are extracted using FFT. Therefore, the algorithm must extract the FFT coefficients from AE-signals – once these signals are recorded using the fast acquisition mechanism of V-Box – or calculate the FFT for the remaining signals.

On the following step, the air-cut process time is removed using a combination of two methods. The signal intensity and encoder positions are analyzed. The signal interval gathered when the machine tool is far from the workpiece is then deleted, as well as the intervals when the energy of the signal is below the threshold – which means the tool was not cutting material.

From the resulting spectrogram containing the sensor signal for one tool, the system combines the interpolated tool wear curve. On this way, there will be one label value – VB calculated from the interpolation – for each time instant of the extracted signal.

This procedure is repeated for the 4 tools, resulting in one dataset for the referred signal. Furthermore, the procedure is repeated for each one of the recorded signals: AE, vibration and microphone. So, the process results in multiple datasets, each one containing the input vector extracted from one of the sensor signals gathered in the experiments.

On the next sub-chapters, the implementation of each one of the transcribed steps of the pre-processing will be explained in details. The last sub-chapter presents an overview of results achieved in this step of the project.

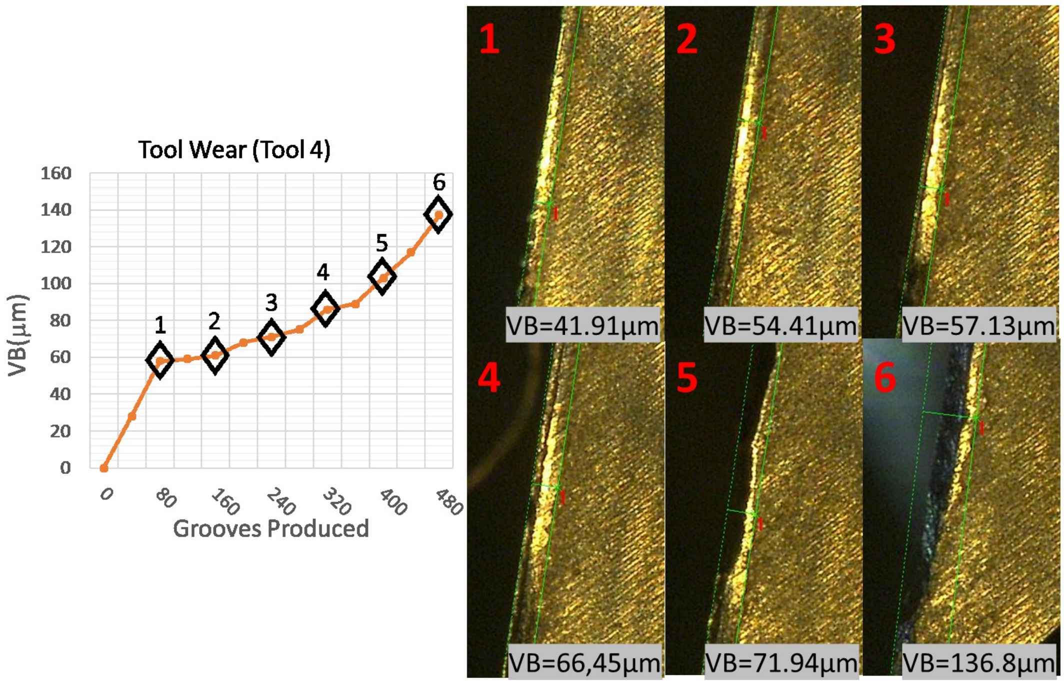
5.1. Flank Wear Measurement and Extraction

As explained in section 3.4, the measurement procedure chosen for the project assesses 3 different regions of the tool for each tooth of the tool. Resulting 9 measurement points.

By performing the experiments, it is possible to confirm the pattern reported by [31] for the flank wear development. On early stages – until 50μm – the wear increases rapidly due to material loss on the top of the tool. After this stage, the curve stabilizes, and the flank wear develops gradually until it reaches around 70μm. On this stage, the tool starts to present chipping, and therefore, the surface starts to present faster degradation.

When VB passes 80μm, the tool starts losing material on the edge, chipping and crater wear phenomena is intensified. As a result, the wear develops fast again. It is possible to notice an increase in the noise produced by the process and machine loses efficiency. See Figure 40.

Figure 40 – Flank wear curve and presentation of different stages of tool wear.

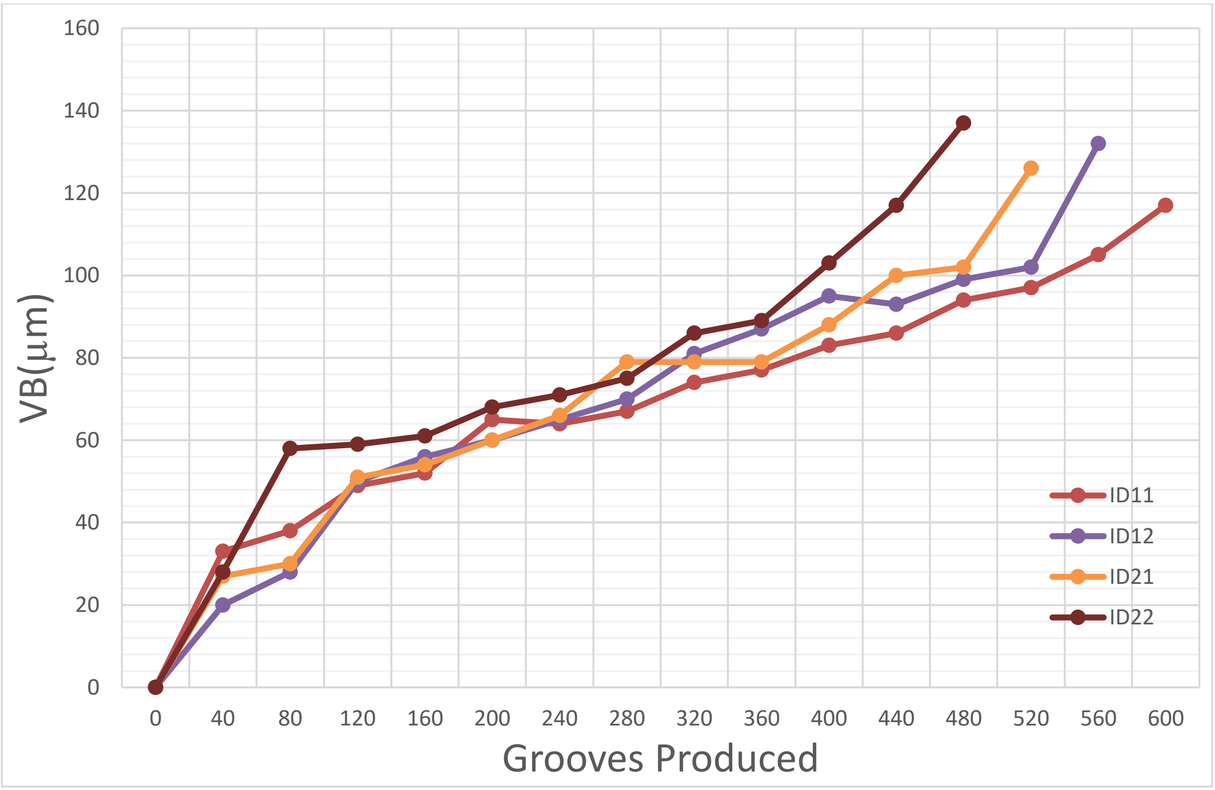


The losses presented by the tool on the last stages of its use increase the difficulty of measuring the wear. As presented in Figure 40, when the tool edge is heavily deteriorated mainly by chipping, the edge lines used as reference for the worn surface assessment are lost. Excepting the region close to the tool tip, this damages the reliability of the measured values once the guide-line is not visible anymore.

To overcome this problem, not only additional lights but also other reference mechanisms are used to try capturing the right values for the entity. The commonly used method consists in observing the chamfer localized 1mm from the tool edge when the tool is new, so the lost edge can then still be indirectly accessed.

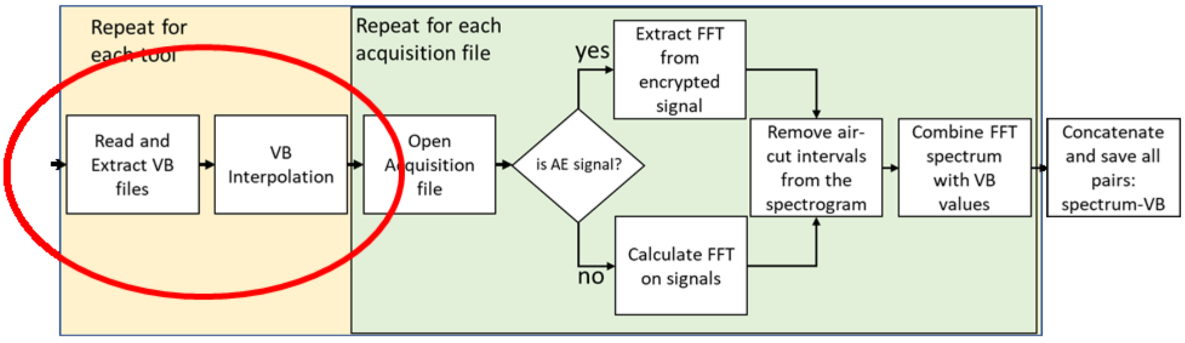
Figure 41 present the tool flank wear curves of the 4 tools. During the lifespan of the tool it is possible to notice 3 distinct regions. At the beginning of curve the wear develops very fast until it reaches approximately 50µm. After this point, there is a region presenting a gradual progress until the VB reaches 80µm. After this point, the effects previously described are responsible for speeding up the deterioration of the tool.

Figure 41 – Flank wear curves of the gathered experiments.



The first step performed by the pre-processing module is to obtain the flank wear curve of each tool. As previously described, from the 9 measurement points performed in each measurement procedure, the chosen value is the maximum between those. Therefore, the first step is to read and calculate the maximum flank wear from the 9 measured values. The complete algorithm for the flank wear extraction is presented in Figure 42.

Figure 42 – Algorithm for flank wear extraction.


Repeat for each tool
Read and Extract VB files
VB Interpolation
Repeat for each acquisition file
Open Acquisition file
is AE signal?
yes
no
Extract FFT from encrypted signal
Calculate FFT on signals
Remove air-cut intervals from the spectrogram
Combine FFT spectrum with VB values
Concatenate and save all pairs: spectrum-VB

```
For each tool repository:
        Open Wear measurement File
        Calculate Maximum VB for each measurement
        calculate polynomial interpolation relating the flank
        wear values with the grooves registered:
                              $VB = P_5(gr)$
        **
        perform further steps
        **
```

As previously mentioned, there are 40 grooves produced between each microscope measurement. However, the output dataset must contain VB values for all operation time of the tool. Therefore, processing the tool wear measurements requires an interpolation method for the VB curve. The usual procedure is to use a “holder” approach, i.e., the variable is maintained constant between each 2 measurement points.

The approach chosen for this project uses a polynomial interpolation to generate a smooth curve for the tool wear progress. Such technique, which produces a smooth curve during the tool lifetime is a closer representation of the physical phenomena. Furthermore, the interpolation helps filtering errors inserted by the microscope measurement procedure.

Based on the typical behavior of the wear curve – rapid increase, gradual increase and finally rapid increase – an odd degree must be chosen for the polynomial function. Since the 1st degree interpolation does not capture the required smooth behavior, the 3rd degree was employed, however the resulting curve did not approximate to all measurement points. On this way, the pre-processing program uses a 5th degree polynomial. Figure 43 presents the polynomial regression for Tool 1 and 3 respectively. The polynomial results are presented in Table 14.

Figure 43 – Polynomial Interpolation on flank wear for tool 1 and 3 respectively.

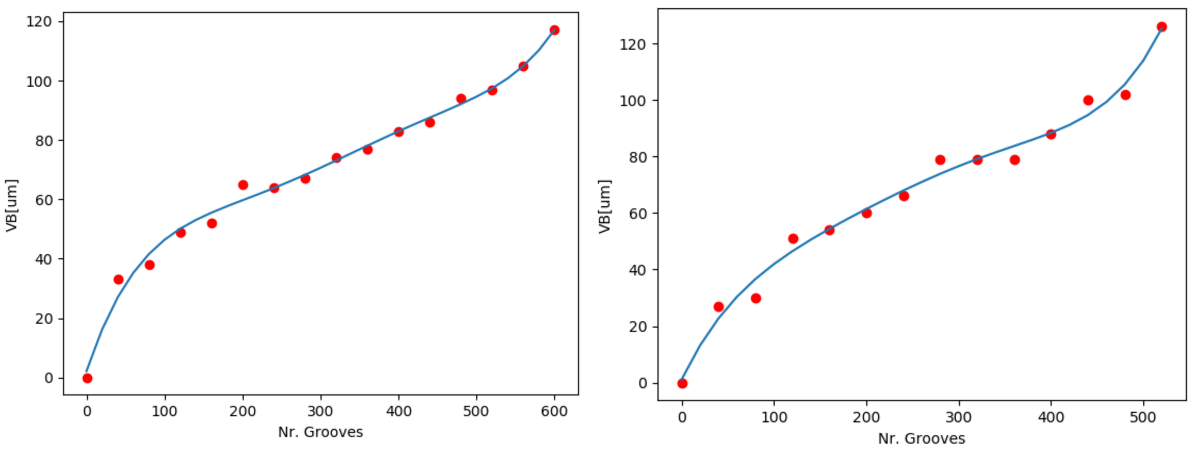


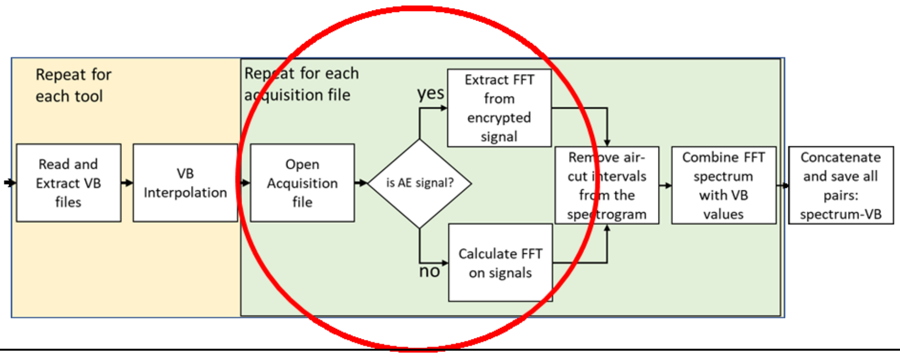
Table 14 – Polynomial interpolation results.

Interpolation Results								
Tool	**ID**	**coefficients:** $VB = a_1x^5 + a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6$						**Mean Squared Residual [µm²]**
		a_1	a_2	a_3	a_4	a_5	a_6	
1	11	$1.598 \cdot 10^{-11}$	$-2.63 \cdot 10^{-8}$	$2.164 \cdot 10^{-5}$	-0.00492	0.793	2.113	6,55
2	12	$3.296 \cdot 10^{-11}$	$-4.409 \cdot 10^{-8}$	$2.165 \cdot 10^{-5}$	-0.00498	0.736	-1.594	13,09
3	21	$2.712 \cdot 10^{-11}$	$-3.388 \cdot 10^{-8}$	$1{,}64 \cdot 10^{-5}$	-0.00402	0.679	1.036	12,83
4	22	$3.562 \cdot 10^{-11}$	$-5.187 \cdot 10^{-8}$	$3.032 \cdot 10^{-5}$	-0.00836	1.159	-1.373	7,98

5.2. Sensor Data Extraction

After this step, as previously described, the sensor signal must be extracted. When reading the AE-signal, which is already recorded using FFT units, the algorithm needs to separate frequency from amplitude on the encrypted values sent by V-Box. The complete algorithm for signal extraction is described in Figure 44.

Figure 44 – Algorithm for data Reading and FFT extraction.


Repeat for each tool
Read and Extract VB files
VB Interpolation
Repeat for each acquisition file
Open Acquisition file
is AE signal?
yes
no
Extract FFT from encrypted signal
Calculate FFT on signals
Remove air-cut intervals from the spectrogram
Combine FFT spectrum with VB values
Concatenate and save all pairs: spectrum-VB

```
**
For each Acquisition File on tool repository:
        Open HDF File
        Load signal array into memory
        If signal is AE:
                amplitudes = AE.arr >> 16 (shift values right-wise)
                frequencies = AE.arr & $(2^{16} - 1)$
                        (eliminate 16 most significative digits)
                spectrogram = Matrix(amplitudes X frequencies)
                remove from spectrogram 64th coefficient and further
        else:
                calculate_fft(array)
##
```

The AE-signal acquired using the V-Box fast acquisition channels records 100 coefficients of the raw signal. Each encrypted value has 16 bits corresponding to the frequency and 16 bits corresponding to the amplitude. The frequency band for recording the AE signal is $0 < freq < 500kHz$. Therefore, the spectrogram for such signal is extracted by using Boolean operations over the encrypted values. From the 100 recorded coefficients, no emission was recorded on the higher frequency band ($freq > 320kHz$), i.e., the values were zero. Therefore, all coefficients above 64th position were deleted from the dataset.

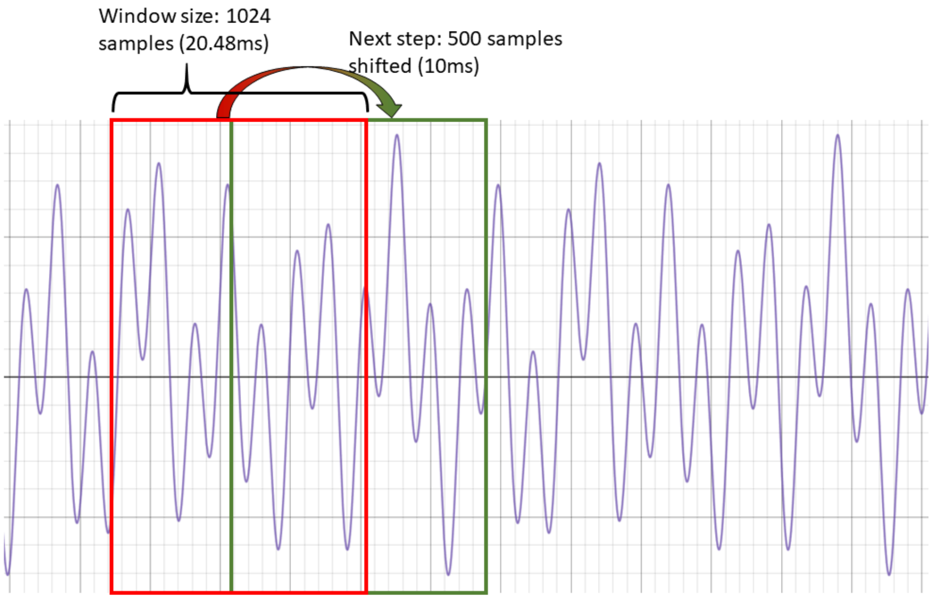
On the other hand, the spectrograms from the accelerometer and microphone must be calculated since those variables are recorded as time-series arrays. This procedure is performed by using a discrete Fourier Transform recursively on the time-series data. Figure 45 presents the algorithm for the computation of the FFT on the time-series arrays. Figure 46 shows a representation of the parameters used for computing the FFTs.

The raw signals were acquired at a frequency of 50kHz. The window size for the FFT chosen is 1024 samples (20,48ms). Between each computed harmonics, there are 500 values (10ms). Therefore, the resulting spectrogram possesses 50 coefficient values on the frequency band $0 < freq < 25kHz$ sampled at a frequency of 100Hz.

Figure 45 - Algorithm for computing the FFT on time sires arrays.

```
i = 0, w_size=1024; overlap = 500; n_freqs=50;   initialize params
While i < end_of_signal:
    x = signal [i : i + w_size] ——————→get from array values for the FFT computation
    for j in w_size: ——————→runs over the array
        $fft[j] = \sum_{k=0}^{w\_size} x[k] \cdot e^{\frac{-2\pi\sqrt{-1}\cdot j\cdot k}{w\_size}}$ ——→computes the complex coefficients of the FFT
    $scaling = \frac{2}{w\_size^2}$ ——————→determines the scaling factor
    $y = fft[:] \cdot conjugate(fft[:])$ ——————→multiplies the conjugate in order to obtain the real frequency values
    $y *= scaling$ ——————→scales the signal
    $y = y\left[:\frac{w\_size}{2} + 1\right]$ ——————→deletes the negative frequencies
    $r = \frac{size(y)}{n\_freqs}$ ——————→determine the reduction size
    for j in n_freqs:
        $coef[j] = sum(y[r \cdot j : r(j + 1)])$ ——→reduces the number of coefficients by summing up neighbours
    $spectrogram[i] = coef$ ——————→append the computed values on the spectrogram
    i += overlap ——————→overlaps the window
return spectrogram
```

Figure 46 – Representation of the parameters used to calculate the FFTs.


Window size: 1024 samples (20.48ms)
Next step: 500 samples shifted (10ms)

Using the maximum speed of the Trialink network (100kHz), the V-Box hardware FFT unit provided one complete spectrum each 1ms (1kHz). Figure 47 presents the signal gathered from the EA-sensor on the workpiece clamp. Figure 48 presents the signal gathered from the EA-sensor on the spindle body. On the figures, it can be observed when 5 complete grooves are removed by the tool.

Figure 47 – Sample data gathered from AE workpiece sensor.

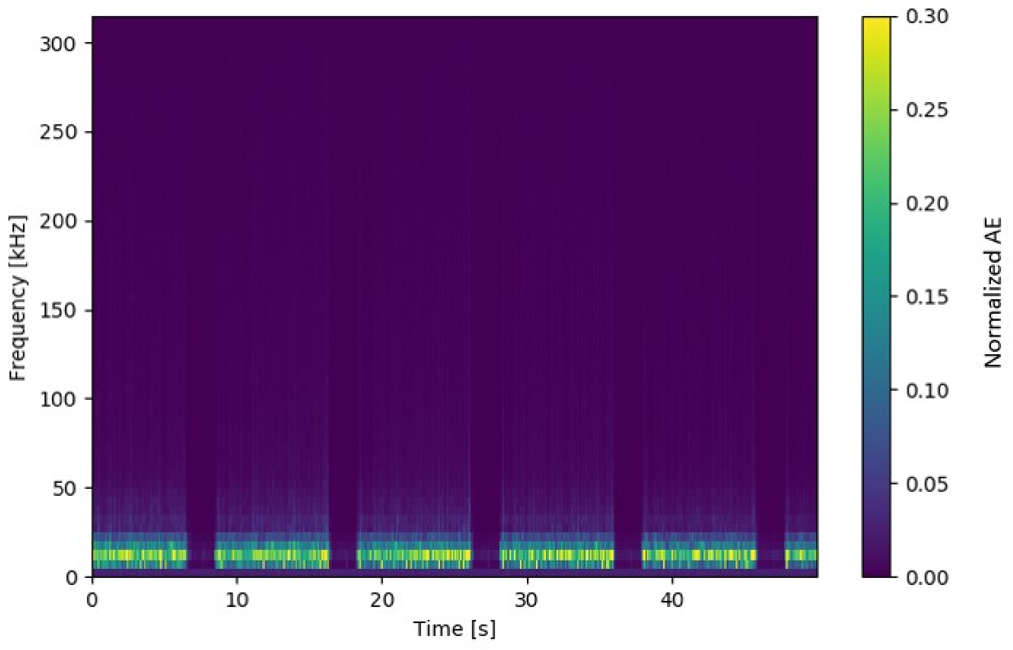
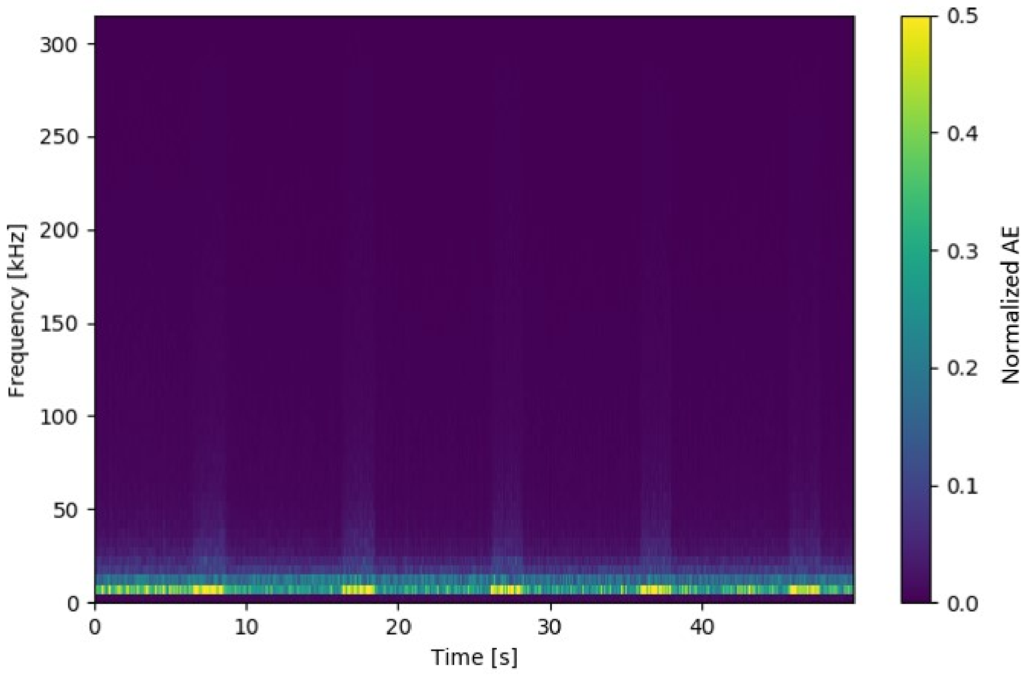


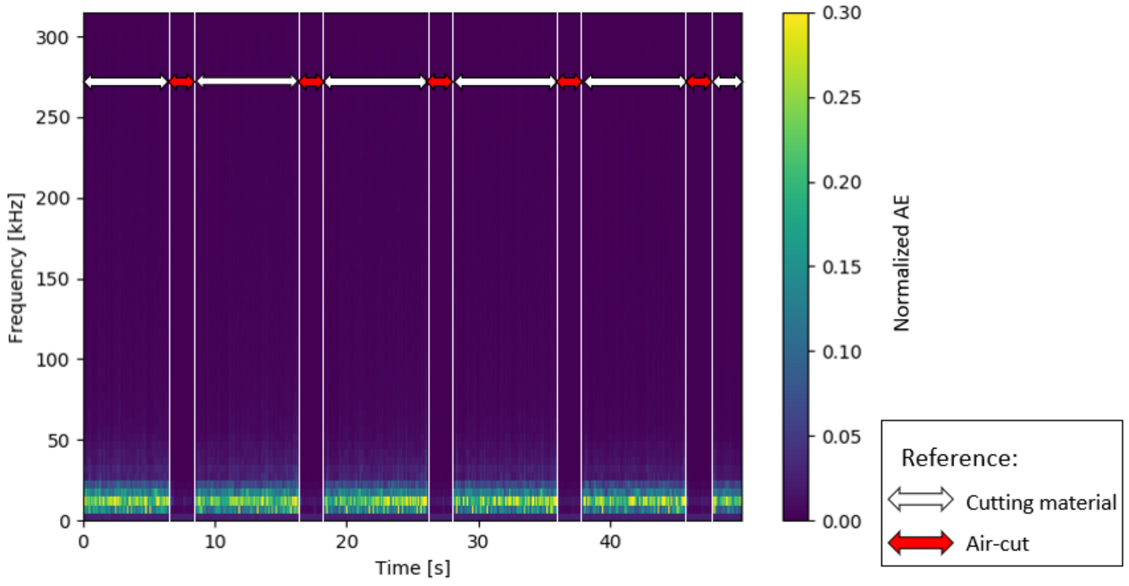
Figure 48 – Sample data gathered from AE spindle sensor.



In both spectrums the signal captured around 5kHz has the strongest emission among the spectrogram bands. Such behavior is observed in all stages of the experiment.

Another aspect to observe is the difference in the signal dynamics when the tool is machining, or it is in air-cut. See Figure 49. As expected, the AE captured in air-cut presents a weaker intensity for the sensor installed in the workpiece clamp. Unlike the first, but still distinguishable, the emission found in AE-signal from the spindle body is stronger when air-cutting. Such behavior suggests the spindle has stronger vibration when out of process.

Figure 49 – AE-signal workpiece, air-cut visualization.



The accelerometer data presents a similar but clearer difference in the 2 different stages of the process (Figure 50). Considering that the spindle speed on all experiments is 6760 min^{-1}, the emissions captured by the sensor are related to the natural frequency of the spindle during air-cut. Once the cutters start removing material from the workpiece, the peak frequency rapidly switches to the process frequency, figured around 25kHz.

Figure 50 – Accelerometer sample data (X axis).

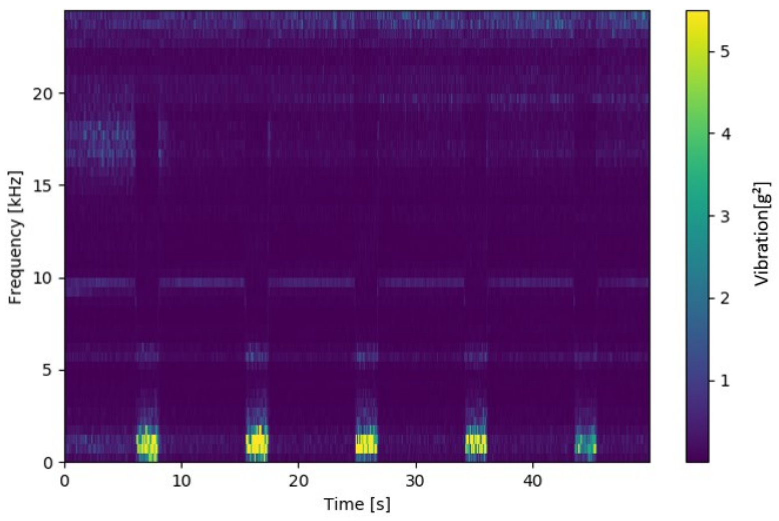
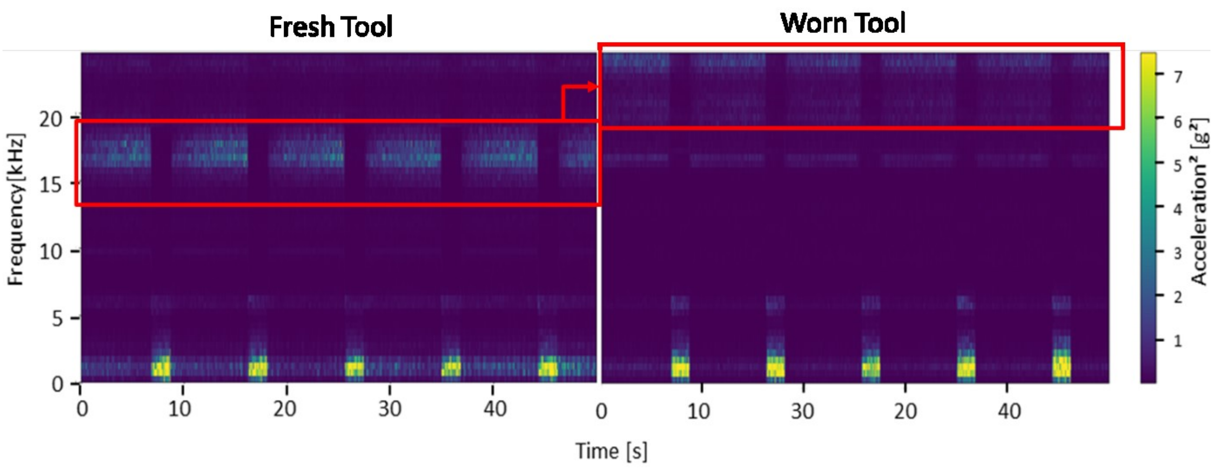


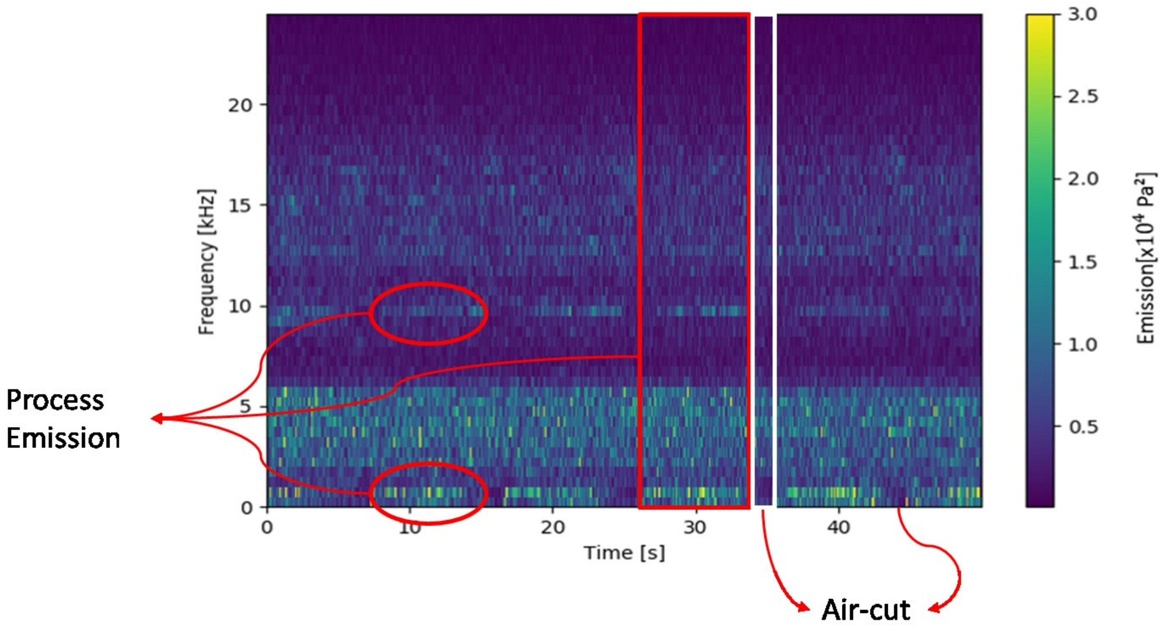
Figure 51 shows two spectrums from the beginning and end of a tool life. As reported in the literature, a worn tool vibrates in higher frequencies when compared to a fresh one. On a fresh tool, the signal presented higher intensity around 17kHz, changing to 25kHz when the tool is worn. On the other hand, not significant changes were observed in the amplitude.

Figure 51 – Vibration X: samples of a fresh and worn tool.



Finally, on the microphone data, the regions of process and air-cut are also visible. However, differences between signal and noise are weaker when compared to other sensor data presented above. See Figure 52.

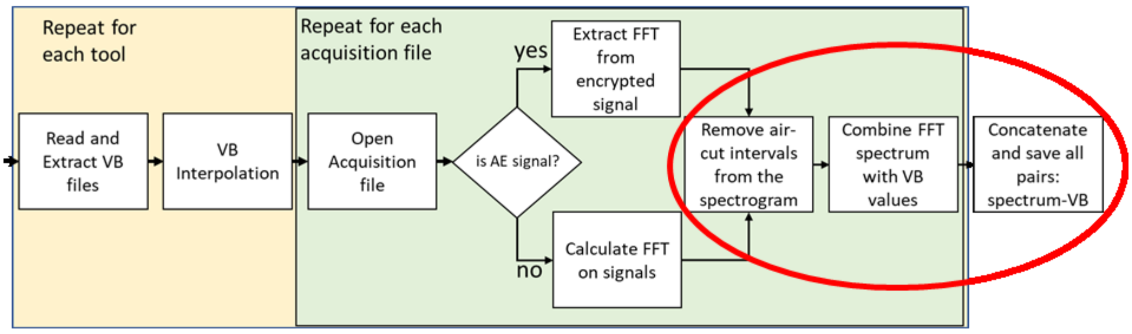
Figure 52 – Microphone sample data.



5.3. Air-Cut Removal and Dataset Persistence

Before the dataset is ready for the prediction module, 2 additional steps must be performed. The algorithm for this steps is presented in Figure 53.

Figure 53 – Algorithm for air-cut and persistency step.

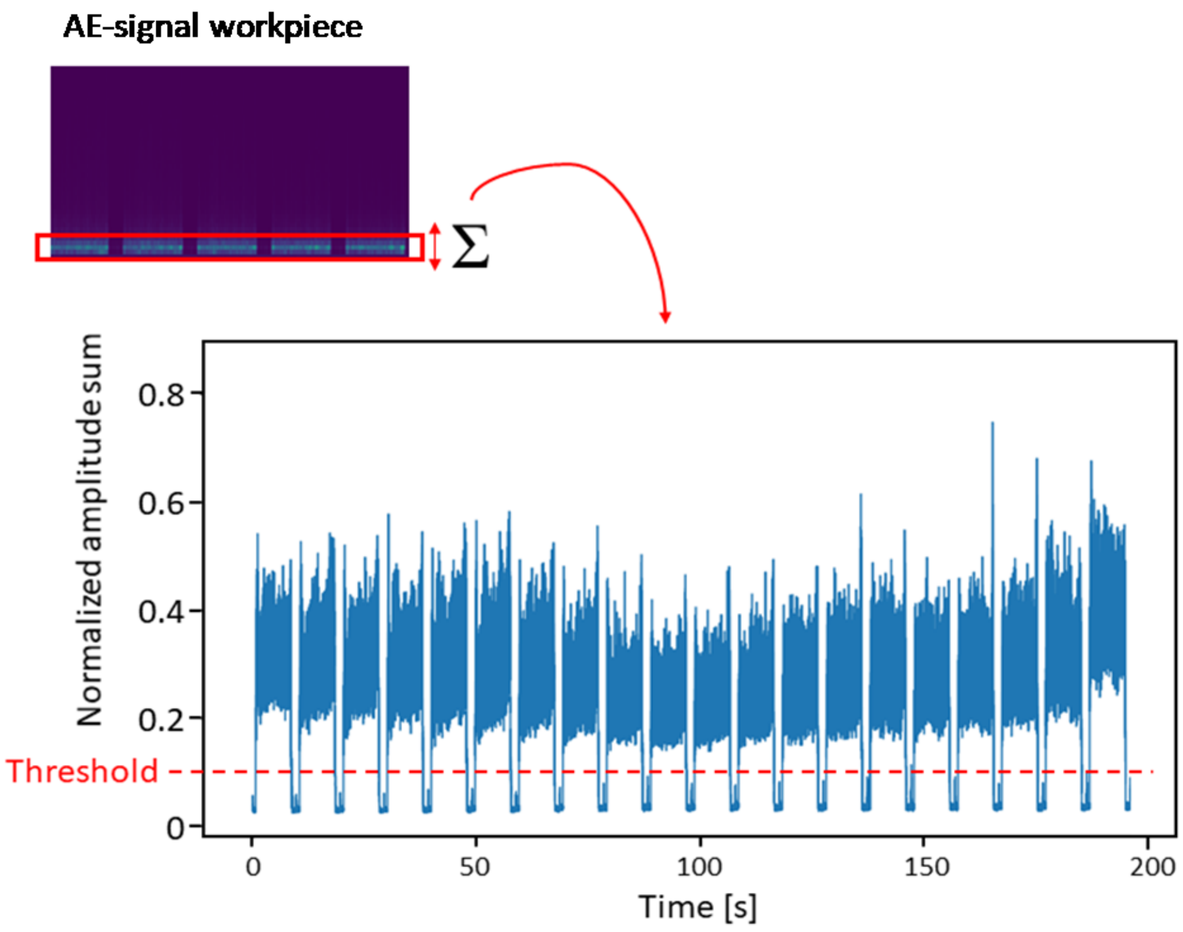

Repeat for each tool
Read and Extract VB files
VB Interpolation
Repeat for each acquisition file
Open Acquisition file
is AE signal?
yes
no
Extract FFT from encrypted signal
Calculate FFT on signals
Remove air-cut intervals from the spectrogram
Combine FFT spectrum with VB values
Concatenate and save all pairs: spectrum-VB

```
##
temp_b = spectrogram.bands where air-cut ≠ process
sum_arr = sum bands (spectrogram[temp_b])
delete from spetrogram where sum_arr < threshold
determine groove interval of file: $[g_0, g_1]$
Create array containing the curve region for the file:
                                    vb_arr = $P_5(g_0, g_1)$
Enqueue resulting arrays: [spectrogram, vb_arr]
##
dataset = Concatenate [queue]
save locally(dataset)
**
```

Firstly, the air-cut time interval must be removed from the spectrogram. This is done in 2 steps:

- The encoder positions are checked, and the time interval of the spectrogram acquired when the machine is far away from the workpiece is removed;
- The module sums up the frequency coefficients which present higher distinction between in and out of process. From the resulting array of sums, a threshold criteria is used in order to identify and remove air-cut. See Figure 54.

Figure 54 – Sum of spectrum amplitudes for one acquisition file.



The result of such operation is the clean spectrogram, containing only information gathered on the cutting process for all 4 tools. This spectrogram, will be used in order to generate the inputs for the further prediction models.

The last operation is to create an additional array, which will contain the labels of the datasets. For this purpose, the 4 interpolated flank wear curves are included in an array of the same size as the spectrogram. Once the algorithm knows how many grooves were removed, it can find the corresponding VB values on the interpolated curve. See Figure 55.

5.1. Pre-Processing Results

The experiments gathered from the 4 tools resulted in 5,71 hours of data, summing 4,52 hours of active operation (tool cutting material). Then, the raw acquisition files contain approximately 28,3 GB. From which, Around 7 workpieces were used in the process.

The module saved two datasets for the vibration sensor. On the first one, the axes amplitudes were summed up, resulting in 50 coefficients containing the summed amplitudes for the three axes of the sensor – X, Y and Z. On the second one, the axes were appended, resulting in a 150 coefficients. Both datasets will be used for the training step. The resulting datasets are shown in Table 15.

Figure 55 – Representation of the air-cut removal and dataset persistency.

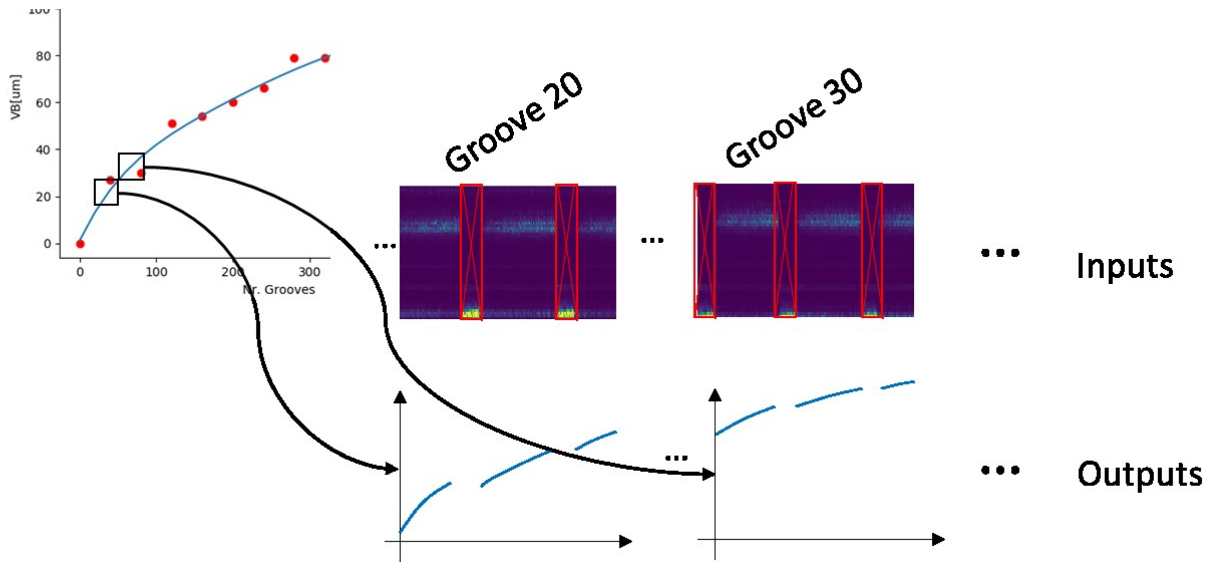


Table 15 – List of resulting datasets.

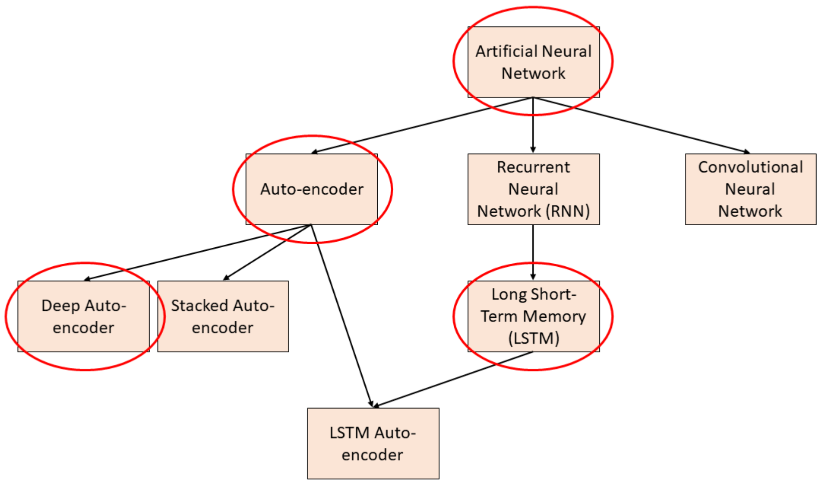
Pre-Processing Datasets		
Name	**Sensor**	**Columns (frequency coefficients)**
vib_sum	Accelerometer (axes summed)	50
vib_x3	Accelerometer (axes appended)	150
mic	Microphone	50
ae_workpiece	AE (sensor on workpiece clamp)	64
ae_spindle	AE (sensor on spindle body)	64

6. Prediction Models

The last step for the tool flank wear prediction system involves the project and implementation of the prediction module. This module has the goal of creating and training the prediction models based on the datasets obtained on the pre-processing step. Therefore, The target prediction models must be able to interpret the FFT spectrograms presented by the sensor and infer the actual state of the tool wear.

Based on those characteristics, the prediction module will train 3 different network architectures for the spectrogram interpretation task. See Figure 56. For comparison reasons, the first chosen architecture for the prediction model are the ANNs, due to its popular use on tool wear prediction [3]. For the second approach, the technique chosen is aimed to select appropriate features from the spectrogram which represent the most deterministic information about the signal. Therefore, in order to select this feature and consequently discard the less relevant regions of the spectrogram, the second architecture uses a combination of auto-encoders for feature selection followed by an ANN responsible for predicting the flank wear values.

Figure 56 – Representation of some DL techniques for supervised learning.


Artificial Neural Network
Auto-encoder
Recurrent Neural Network (RNN)
Convolutional Neural Network
Deep Auto-encoder
Stacked Auto-encoder
Long Short-Term Memory (LSTM)
LSTM Auto-encoder

The last approach chosen for solving the problem consider the tool wear prediction as a sequence of indicators or events presented by the spectrogram regarding the signal behavior. Therefore, the prediction models should possess some long-term memory structure in order to identify such indicators and infer the tool condition. Therefore, the model should be able to memorize features found in the spectrogram in such a way that these features can help determining the state of the tool. So, the last approach uses an LSTM for classification and regression.

6.1. Training Algorithm

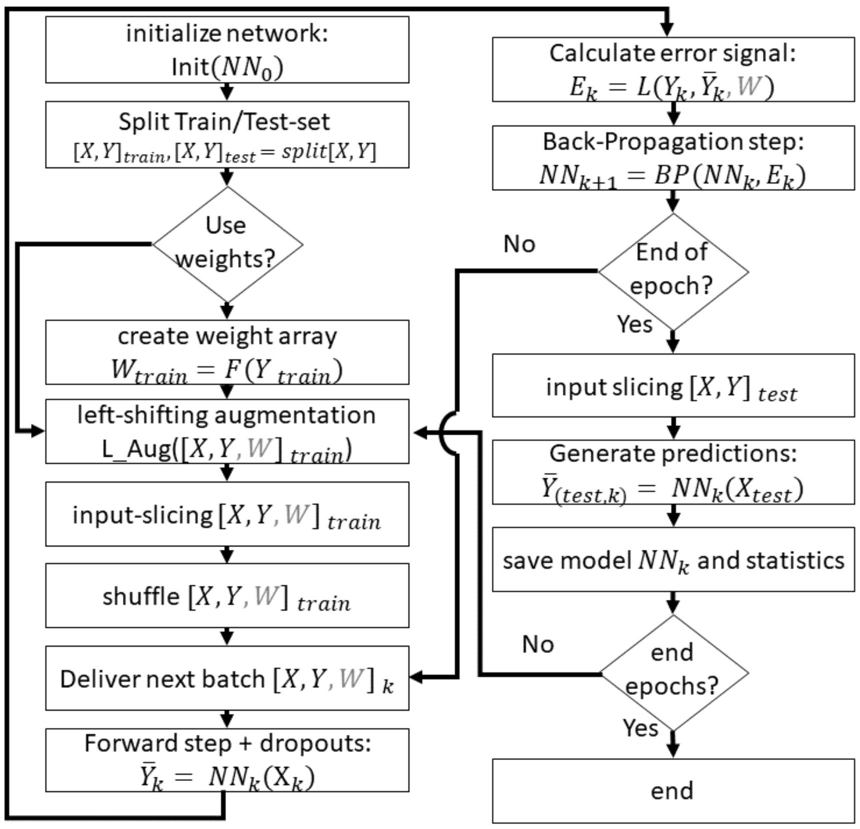
The tasks entrusted to the prediction model are: read the pre-processed dataset and prepare the data for training and evaluation; select, build and initialize the prediction model (architecture); and train/evaluate the model according to the dataset.

Figure 57 shows the algorithm for training one prediction model. Initially, the system loads and reads the dataset, then split the dataset into train and test-set. 10% of the dataset is selected for the test-set.

The training loop comes right after this step. It is important to remind that the pre-processing step does not save each input on the datasets. It saves one big array containing the information of the entire dataset. So, in order to fit the model, the inputs and targets are generated in runtime.

This approach was chosen so it is possible to perform data augmentation between each training epoch, as it will be explained afterwards.

Figure 57 – Algorithm for the Prediction Module.


initialize network: $\text{Init}(NN_0)$
Split Train/Test-set $[X,Y]_{train}, [X,Y]_{test} = split[X,Y]$
Use weights?
create weight array $W_{train} = F(Y_{train})$
left-shifting augmentation $\text{L\_Aug}([X,Y,W]_{train})$
input-slicing $[X,Y,W]_{train}$
shuffle $[X,Y,W]_{train}$
Deliver next batch $[X,Y,W]_k$
Forward step + dropouts: $\bar{Y}_k = NN_k(X_k)$
Calculate error signal: $E_k = L(Y_k, \bar{Y}_k, W)$
Back-Propagation step: $NN_{k+1} = BP(NN_k, E_k)$
End of epoch?
No
Yes
input slicing $[X,Y]_{test}$
Generate predictions: $\bar{Y}_{(test,k)} = NN_k(X_{test})$
save model NN_k and statistics
end epochs?
No
Yes
end

6.1.1. Training Weights

As described in section 5.2, the flank wear develops faster in the beginning and end of the tool life. Therefore, those regions have fewer samples when compared to the rest of the dataset.

Consequently, when training the prediction models, the algorithm tends to give higher priority on learning the regions with bigger number of samples. This behavior logically presents a better overall accuracy, firstly by the higher portion of explained data, and secondly by the fact that the networks will know those regions better once they have been presented with more samples.

However, those points are particularly important to the aim of the project. Mainly when considering the highest flank-wear samples, there is a higher demand for accuracy once the predictions will determine if the tool has to be replaced.

In order to overcome the problem, one solution is to improve the priority of those regions on the train-set. This is the goal of the weight array on the module. When programmed to, the algorithm creates an additional array of weights. All weights are set to 0,05, excepting the ones which $30 > VB > 100$, receiving 1 instead. On this way, data located on this region will compute 20 times higher loss values, forcing the gradient to move towards the direction of a better performance for this samples.

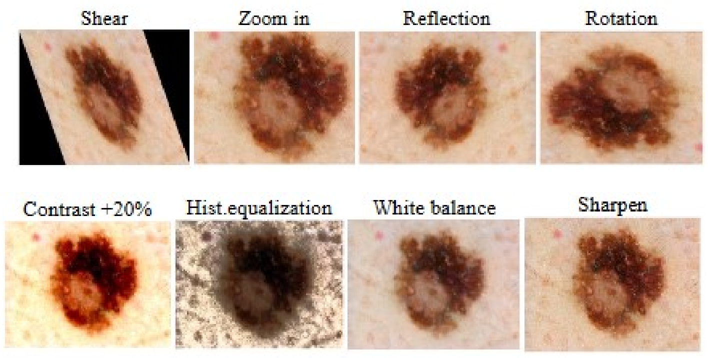
6.1.2. Data Augmentation and Input Slicing

It is important to remind that the pre-processing module saves data for the training as one spectrogram containing all inputs concatenated – all processing time of the 4 used tools – and one array of labels. So the chosen dataset must be sliced in order to generate data for the training step.

The first important method for model training optimization performed by the algorithm is data augmentation. It consists of performing small changes in the input vectors in order to raise dataset variability.

This method is popular in image classification problems, where the algorithms must interpret data on matrixes of pixels presented. Augmentation in such case perform rotation and inclination of the images, changing bright and color of the inputs [38]. See Figure 58.

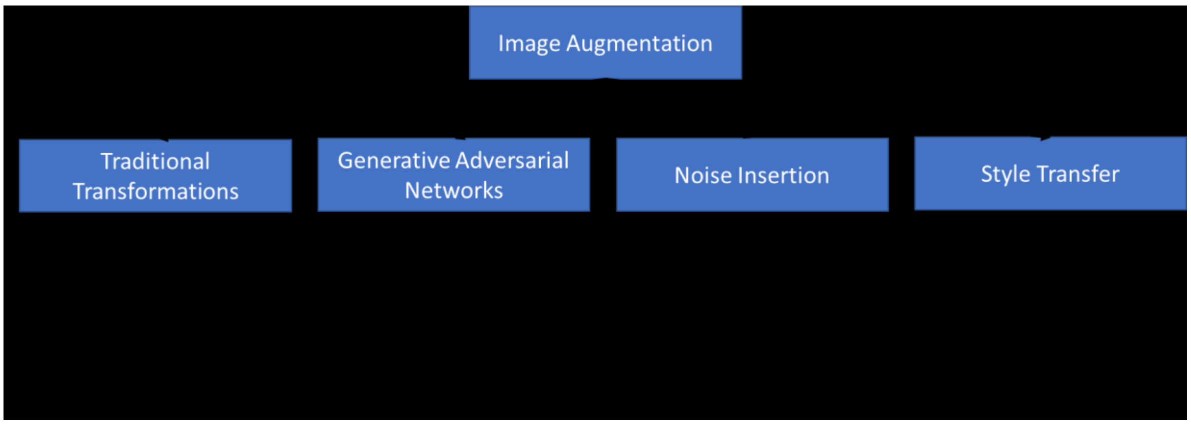
Figure 58 – Example of augmentation on image classification.



Source: [38]

This process guarantees the network will always receive different inputs on each epoch. Helping the algorithm to attenuate overfit or underfit. Figure 59 list the main methods for image augmentation recently reported in the literature.

Figure 59 – Augmentation methods for image classification problems.

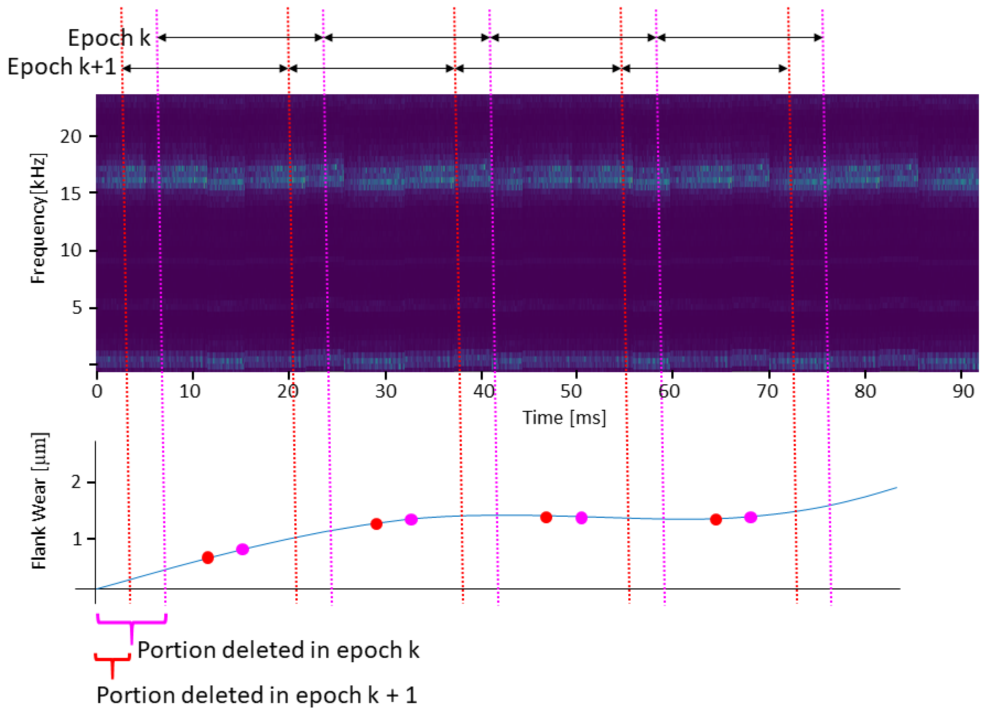


Source: [38]

Considering the nature of the problem studied in this project, the spectrograms cannot be augmented using the standard approaches for image classification. However, the method developed is somewhat similar to a traditional transformation. The procedure consists of – after each epoch – sliding the input limits on a random number of timesteps.

So, before generating the model input values used in the current training epoch, a random interval of the data is deleted from the beginning of the training spectrogram. After this step, the dataset is sliced according to the programmed number of timesteps. See Figure 60.

Figure 60 – Augmentation algorithm representation.

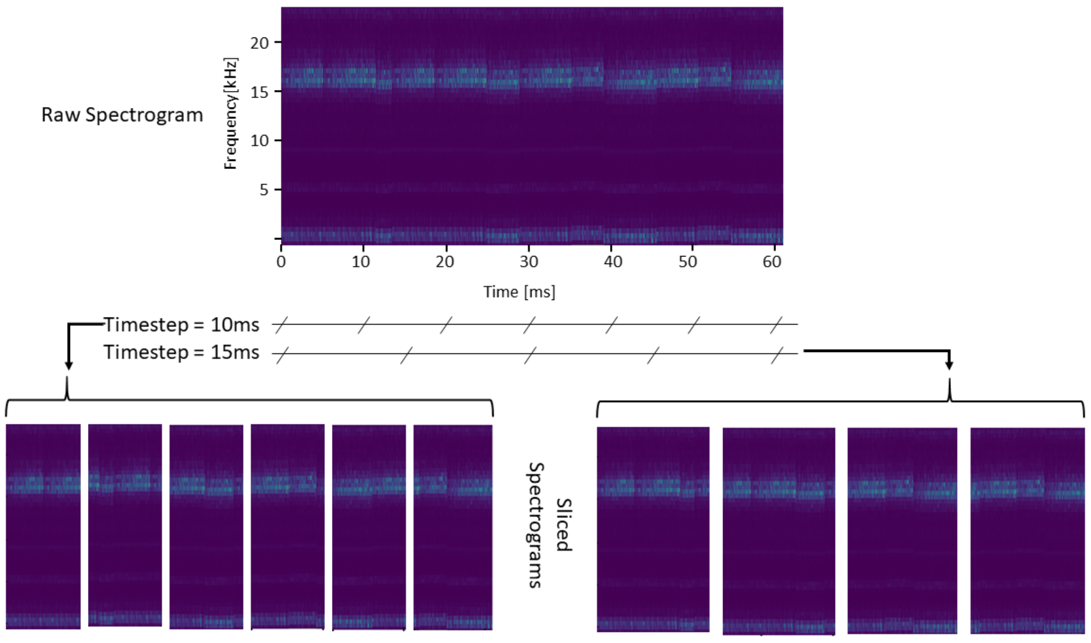


When generating the input vectors, the data reader looks firstly at the original dataset array. Once it is known that one training pair consists of one spectrogram – small slice of original spectrogram containing (for instance) 10ms of data – and one flank wear value, after "deleting" the first 3ms of data, each input will receive a small portion of what would be part of the next input.

To better explain the data slicing procedure. The number of timesteps is considered as a programmable parameter, and it can therefore be ordinarily changed. So, changing the number of timesteps changes the input size, which consequently changes the amount of data received by the model. See Figure 61. This value must be accurately tuned so that neither the network will receive too few information and therefore it will not be able to learn the signal relation to the labels resulting in underfitting, nor it will have too much information resulting in overfitting.

In order to perform such changes on the dataset, the batches generated for training are produced by the algorithm at runtime. Therefore, between each epoch, the augmentation is re-applied and the spectrogram is re-sliced. As a result, the procedure provides always a different set of inputs to the network, using the same source data. The label and (when using) weight for each input are obtained from the values sampled in the middle of each spectrogram input as shown in Figure 60.

Figure 61 – Different timesteps used for slicing the input spectrogram.



Another optimization method used in the project is the Dropouts. In this method, a random portion of the neuron connections on some layers of the NN are set to zero. The motivation for this method is the same as the proposed left-shifting augmentation: the small variations provoked by dropping values push the training algorithm to find critical properties on the data instead of memorizing inputs, attenuating the overfit effect. Each NN architecture performs dropouts on a different way. Therefore, such configuration is customized for each architecture.

6.1.3. Losses and Metrics

Regarding the prediction model training, it is important to explain the usage of loss and metric functions. On the training algorithm, the loss function is used in the forward step of the back-propagation algorithm in order to generate the error signal of the predicted value in comparison with the ground true value as also seen in Equation 2 and 4. The value generated by this loss function will be back-propagated in order to calculate the gradient values and apply the corrections on the weights of the network.

Therefore, selecting the right loss function is critical for the performance of the training algorithm. So, 2 functions were used as loss functions on the prediction module. The mean squared error (MSE) for the regression models. And categorical cross entropy for the classification approach. This last is transcribed in Equation 6.

$$L_{cce}(y,\bar{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}\left(y_{ij} \, . \log(\bar{y}_{ij})\right) \qquad (6)$$

So, categorical cross-entropy will compare the distribution of the predictions with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. In other words, as closest the algorithm gets to the binary hot code of the label, smaller will be the loss.

The accuracy metric function is a function which aims to measure the accuracy of the model. The main difference between the accuracy metric and the loss function is that the accuracy not necessarily needs to be related to the error function. Therefore, it can be a percentage measure of the explained data as it is the accuracy measure.

For the regression algorithm, the MSE is also used as metric function. Excepting for the auto-encoder training, where the accuracy function was employed. The mentioned accuracy measures the deviation to the true value. See Equation 7.

$$M_{acc}(y,\bar{y}) = \frac{1}{N}\sum_{i=0}^{N}\frac{\sqrt{(y_i - \bar{y}_i)^2}}{y_i + \epsilon} \qquad (7)$$

For the classification algorithm, the categorical accuracy function was employed. So, again considering the outputs of a classification network and labels as binary hot codes. The accuracy is calculated as shown in equation (8).

$$M_{cac}(y,\bar{y}) = 1 \; if \; (\arg_{max}(y) = \arg_{max}(\bar{y})) \qquad (8)$$

So, this measure considers the maximum value between the prediction outputs as the chosen class for the example, and compares this class to the label, returning 100% if they match, or 0% otherwise. When evaluating a batch, the metric will return the percentage corresponding to the amount of data correctly predicted by the model.

The loss and metric functions calculate an error or accuracy value for a given dataset. However, when evaluating the dataset after the training step, it may be useful to have a tool in order to measure the model performance locally.

In order to generate local error values, an additional metric was created. The Windowed Mean Error, despite the fact that it cannot be used as a metric or loss function, computes the mean error between the predicted and true values considering all data from the dataset included in a window region defined for the target values.

This algorithm processes the output error as shown in Equation 9.

$$WME(VB) = \frac{\sum_{k=0}^{n} \sqrt{(\bar{y}_k - y_k)^2}}{n} \qquad (9)$$

So, the WME first selects the vectors included in the window interval w. See Equations 10 and 11.

$$N = (\bar{y}_k, y_k)\ \forall\ |y_k - VB| \ \leq w \qquad (10)$$

$$n = \text{dimension}(N) \qquad (11)$$

In resume, the algorithm first looks for all dataset pairs which the true value y_k is close to the target VB by at most w. After this, it extract the mean error between the true and predicted values.

The proposed error metric, when running over all the flank wear curve, provides a local error estimation over the entire curve. By analyzing those values, it is possible to determine the accuracy now based on the regions under observation.

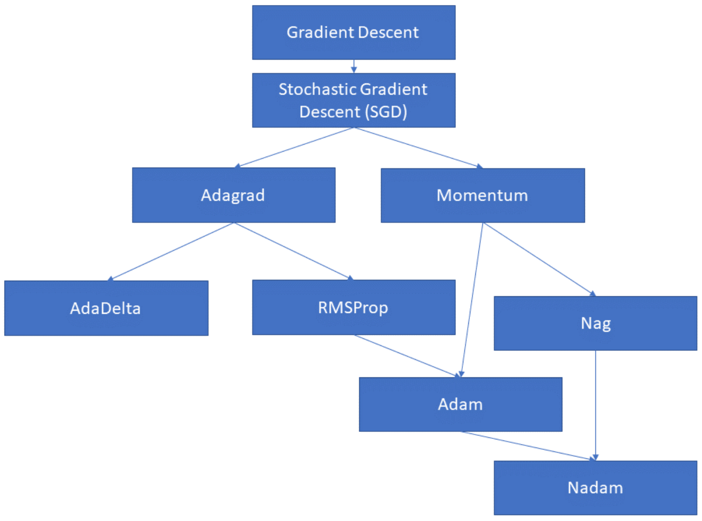
6.1.4. Optimizer Algorithm

Another important element of the model kernel is the optimizer algorithm. When dealing with big datasets, normal gradient-descent algorithms cannot be used. The first reason regards the time required to compute the forward and backward stages for each individual input and output. The second reason is the efficiency of this method, which usually is not able to find good global solutions.

In order to overcome this problem, the Stochastic Gradient Descent algorithm (SGD) was created. Which consists of a variant of the same algorithm shown in chapter 2, however it computes the gradient and weight corrections for a batch instead of a single input.

Researches proved that this approach was not the most suitable for highly complex problems like the ones studied in deep-learning [36]. Therefore, numerous variants of the SGD algorithm were created and proved in the literature afterwards. Figure 62 shows the mainly used optimizer algorithms.

Figure 62 – Optimizer algorithms for Back-propagation.



Source: [36]

The Adam is one of the most popular variants of the SGD strongly implied in ML problems as reported in [36]. It is a combination of 2 other techniques: the Adagrad, which maintains a per-parameter learning rate that improves performance on problems with sparse gradients as encountered in natural language processing and computer vision scenarios; and a Root Mean Square Propagation (RMS-Prop), that also maintain per-parameter coefficients, however it focuses on adapting weights based in all recent gradient values, being particularly useful with noisy data. Figure 63 presents the algorithm for the Adam optimizer.

Figure 63 – Adam Optimization algorithm.

```
Get params: $\alpha, \beta_1, \beta_2, \epsilon$
get initial vector: $\theta_0$
initialize moment vectors: $m_0 = 0;\ v_0 = 0;\ k = 0$
while $\theta_t$ not converged, do:

    $k = k + 1$
    $g_t = \nabla_\theta L_k(\theta_{k-1})$ (get gradients based on the objective function at iteration k)
    $m_k = \beta_1 . m_{k-1} + (1 - \beta_1) . g_k$ (update biased first moment estimate)
    $v_k = \beta_2 . v_{k-1} + (1 - \beta_2) . g_k^2$ (update biased second raw moment estimate)
    $\hat{m}_k = \frac{m_k}{(1-\beta_1^k)}$ (computes bias-corrected first moment estimate)
    $\hat{v}_k = \frac{v_k}{(1-\beta_2^k)}$ (computes bias-corrected second raw moment estimate)
    $\theta_k = \theta_{k-1} - \alpha . \frac{\hat{m}_k}{(\sqrt{\hat{v}_k} + \epsilon}$ (update parameters)

End while
return $\theta_k$ (resulting parameters)
```

The Adam receives 4 parameters, which are: α, the learning rate; β_1, the exponential decay rate for the first order moment estimation; β_2, the exponential decay for the second-moment estimation and; ϵ, a small number to prevent division for zero. Besides that, the algorithm needs an objective function f which is our loss function for the output layer or the error back-propagation signal for our hidden layers in order to compute the next optimum parameters θ_k, which corresponds to the weights of the NN.

The algorithm is also known as an exponential moving variant of Adagrad, showing smaller step sizes when the gradient moves closer to the solution. It is a popular alternative widely spread in deep-learning to solve highly noisy and sparse problems like image classification and speech recognition.

6.2. Architectures and Training

This sub-section is aimed to describe in detail the network architectures used for the prediction models, as well as presenting the results achieved for each one of the employed datasets.

As mentioned in the beginning of the chapter, 3 different architecture approaches were chosen for the prediction task: ANNs, ANN + Auto-encoders and LSTM for regression and classification. Each model presented was trained using only one sensor data as input. On this way, the project is able to present a comparison between different sensors and architectures for the flank wear prediction problem.

Appendix C lists the configuration and results of all different networks trained by the prediction module. The following sub-sections explain the details for each architecture employed, as well as the training results achieved.

6.2.1. Auto-encoders

The auto-encoders train a network to reproduce the input on the output through a compressed layer. On this sense, the dataset labels (flank wear) are not required on its training.

The chosen approach uses only the frequency bands for encoding. Therefore, the encoders must learn how to reduce or combine information of different harmonics of the signals FFT. This process eliminates redundancies between different coefficients of the spectrogram. However, it maintain redundancies in time.

Interpreting temporal relation of the sensors will then be a task for the prediction models. This approach was also chosen in order to maintain the number of samples and consequently the number of inputs, once the datasets are already relatively small.

The codes are generated by sigmoid activators on the output layer of the encoder. Therefore, all codes are normalized: $0 \leq o_j \leq 1$. The remaining units use ReLU activation. The output layers process data through linear activation in order to reconstruct the inputs on the outputs as explained in chapter 2. At last, the networks were tuned so that they can only reach accuracies around 90%. On this way, the auto-encoders attenuate data noise and perform feature selection.

As the AE data demanded higher processing power, 2 different sizes were designed for the auto-encoders: a simple auto-encoder with one layer encoder and decoder for the vibration and microphone datasets; for AE, a deep auto-encoder with 3 layer encoder and decoder was employed. All configuration used to encode each sensor signal is described in Table 16.

Table 16 – Auto-Encoder Network Configurations.

Auto-Encoder Network Configurations				
ID	**Dataset**	**Layers**	**Input shape**	**Code shape**
11	vib_sum	50-10-50	50	10
12	vib_x3	150-10-150	150	10
13	mic	50-10-50	50	10
14	ae_workpiece	64-32-16-4-16-32-64	64	4
15	ae_spindle	64-32-16-5-16-32-64	64	5

Another factor tuned for controlling the model accuracy was the code size of the networks. Even after reducing the code size, the training did not require more than 10 epochs in order to interpret and compress the datasets. Figure 64 shows the training progress for the microphone dataset (network 13). Figure 65 shows the results of the Auto-Encoder network trained for the vibration-sum dataset (network 11).

Figure 64 – Auto-encoder training curve for Network 13.

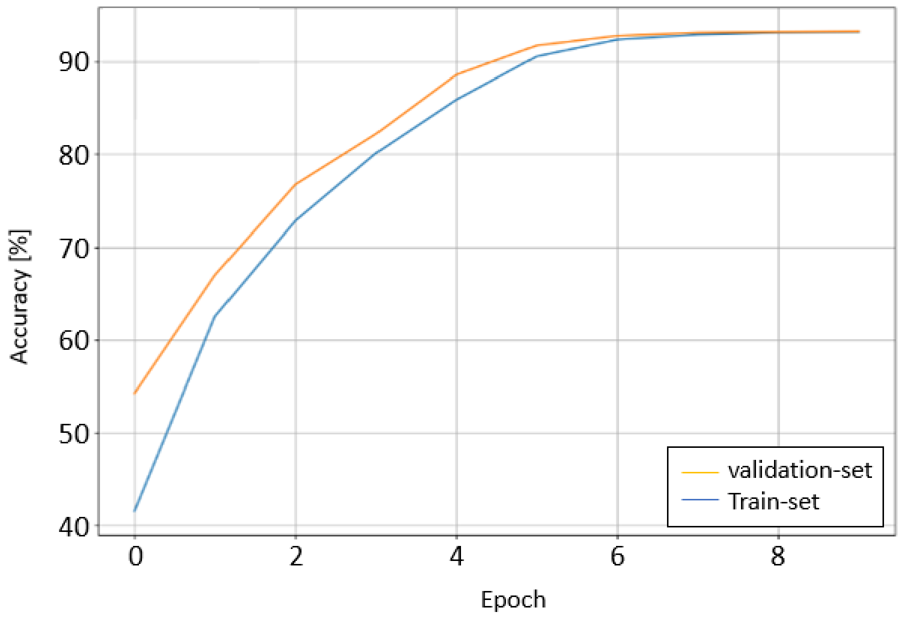
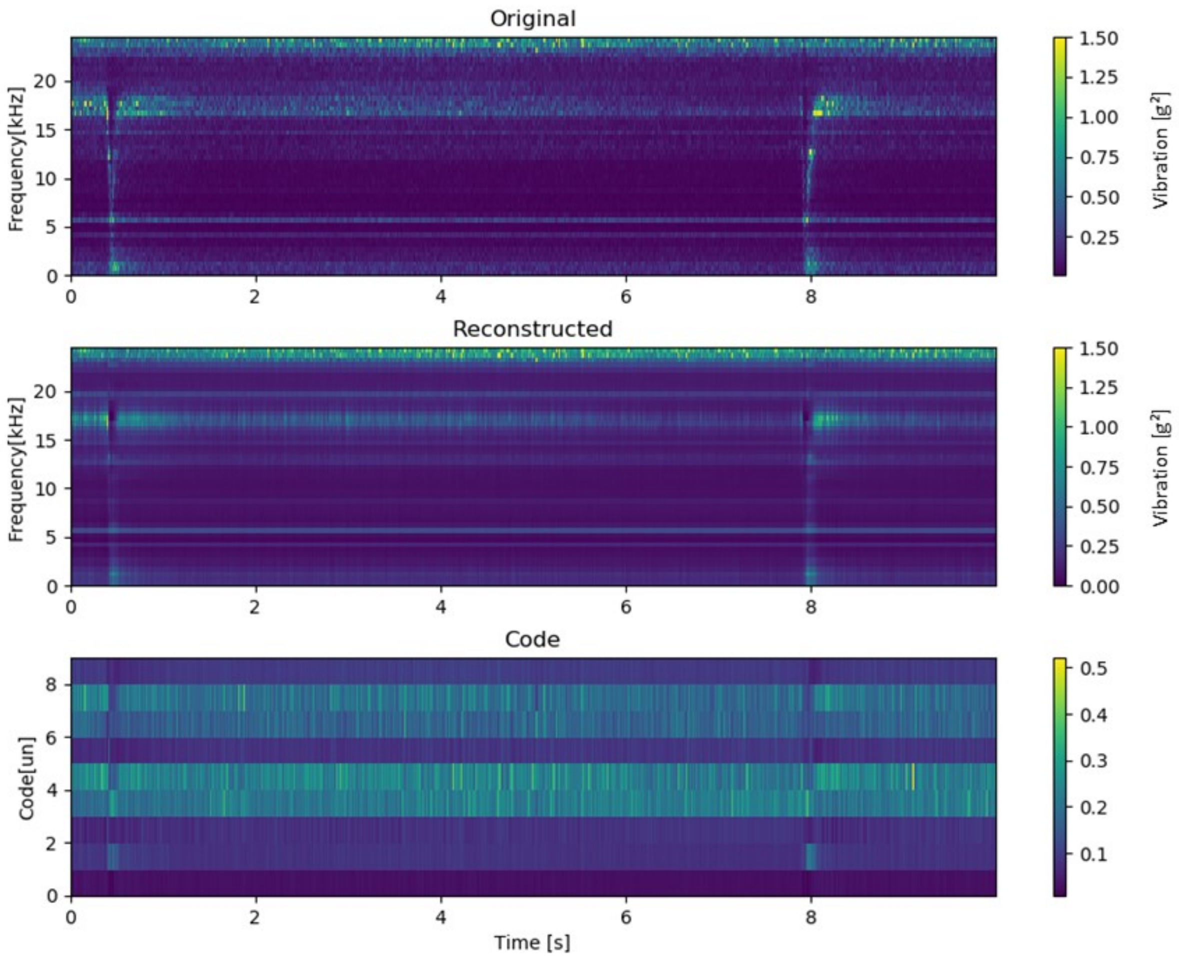


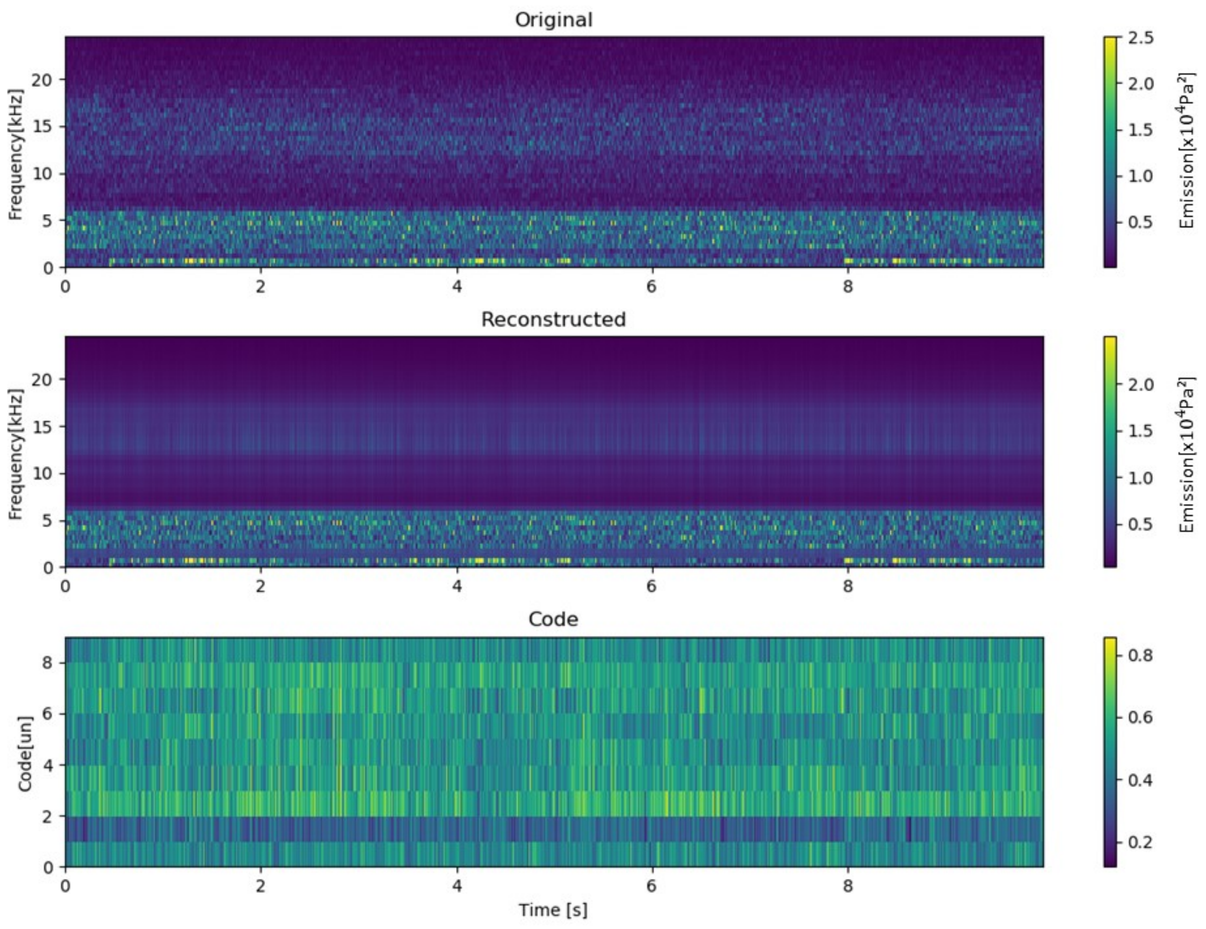
Figure 65 - Auto-encoder signal/reconstructed/code for Network 11.



The auto-encoder training results present some interesting properties of the datasets. The first element to pay attention to is the noise ratio of the sensor signals. A general behavior observed was that "clean" signals are interpreted easily by the auto-encoder. On the other hand, noisy signals make the process slowwe, once those networks must select strategically the most important features for the compression and reconstruction of the inputs afterwards.

This property can be noticed clearly in the raw signals (Figure 51 and Figure 52) in comparison with the signals reconstructed on network 11 (Figure 65) and network 13 (Figure 66). By analyzing the reconstructed signal presented in Figure 66, it is clear the algorithm prioritized the lower frequency band ($f < 7kHz$) of the sensor when compressing the data.

Figure 66 - Auto-encoder signal/reconstructed/code for Network 13.



As a last point, the network achieved unusual results for the AE(spindle) signal. The resulting model has high accuracy, however, it also present very high loss. In other words, the model is accurate but the gradient does not stabilize.

Two different diagnoses can be addressed to such behavior: the first possibility is a poor adjustment of the network hyperparameters, so the gradient is not moving in the right direction or; the referred dataset has the behavior of a random noise unit, where most of the frequency bands are constant, but the noise is intense in some specific regions.

Unfortunately, this point to some problem with the auto-encoder on spindle AE-signal. Once the similar or simpler networks were able to process the information coming from the other signals with reasonable precision.

6.2.2. Artificial Neural Networks

The first presented prediction model approach is a vanilla ANN. The network is composed by successive Dense layers fully inter-connected. The neurons on input and hidden layers of the network use ReLU activation. On the other hand, the output units have linear activation in order to reach the target values, as assigned to regression model approaches.

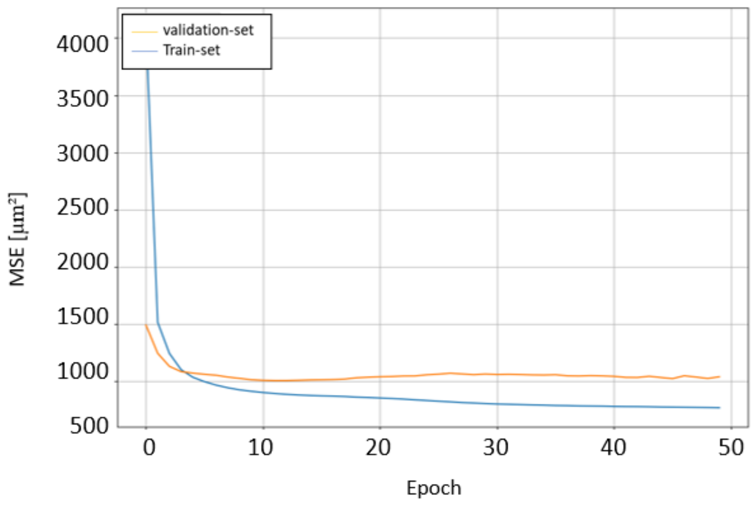
In order to better understand the representation power of this architecture, 2 different network sizes were tested. The configuration of each network employed is shown in Table 17.

Table 17 – ANN Regression Model Configurations.

ANN Regression Model Configurations				
ID	**Dataset**	**Layers**	**input shape**	**MSE test-set**
1	vib_sum	(10,50)-(10,8)-flat*-8-4-2-1	(10,50)	505.6175
2	vib_x3	(10,150)- (10,8)-flat-8-4-2-1	(10,150)	471.5322
3	ae_workpiece	(16,64)- (16,8)-flat-8-4-2-1	(16,64)	463.9799
4	ae_spindle	(16,64)- (16,8)-flat-8-4-2-1	(16,64)	305.8719
5	mic	(10,50)- (10,8)-flat-8-4-2-1	(10,50)	805.3639
6	vib_sum	(10,50)- (10,16)-flat-8-8-4-1	(10,50)	468.42054
7	vib_x3	(10,150)- (10,16)-flat-8-8-4-1	(10,150)	488.3767
8	ae_workpiece	(16,64)- (10,16)-flat-8-8-4-1	(16,64)	293.165
9	ae_spindle	(16,64)- (10,16)-flat-8-8-4-1	(16,64)	313.7809
10	mic	(10,50)- (10,16)-flat-8-8-4-1	(10,50)	822.6655

By analyzing the results presented in Table 17, the bigger networks (networks 6 to 10) present a considerable overfitting effect. This can be confirmed by looking at the training curve presented in Figure 67. However, the results collected from the smaller networks (networks 1 to 5) achieved poor accuracy rate while still overfitting.

Figure 67 – ANN training curves for Networks 1.

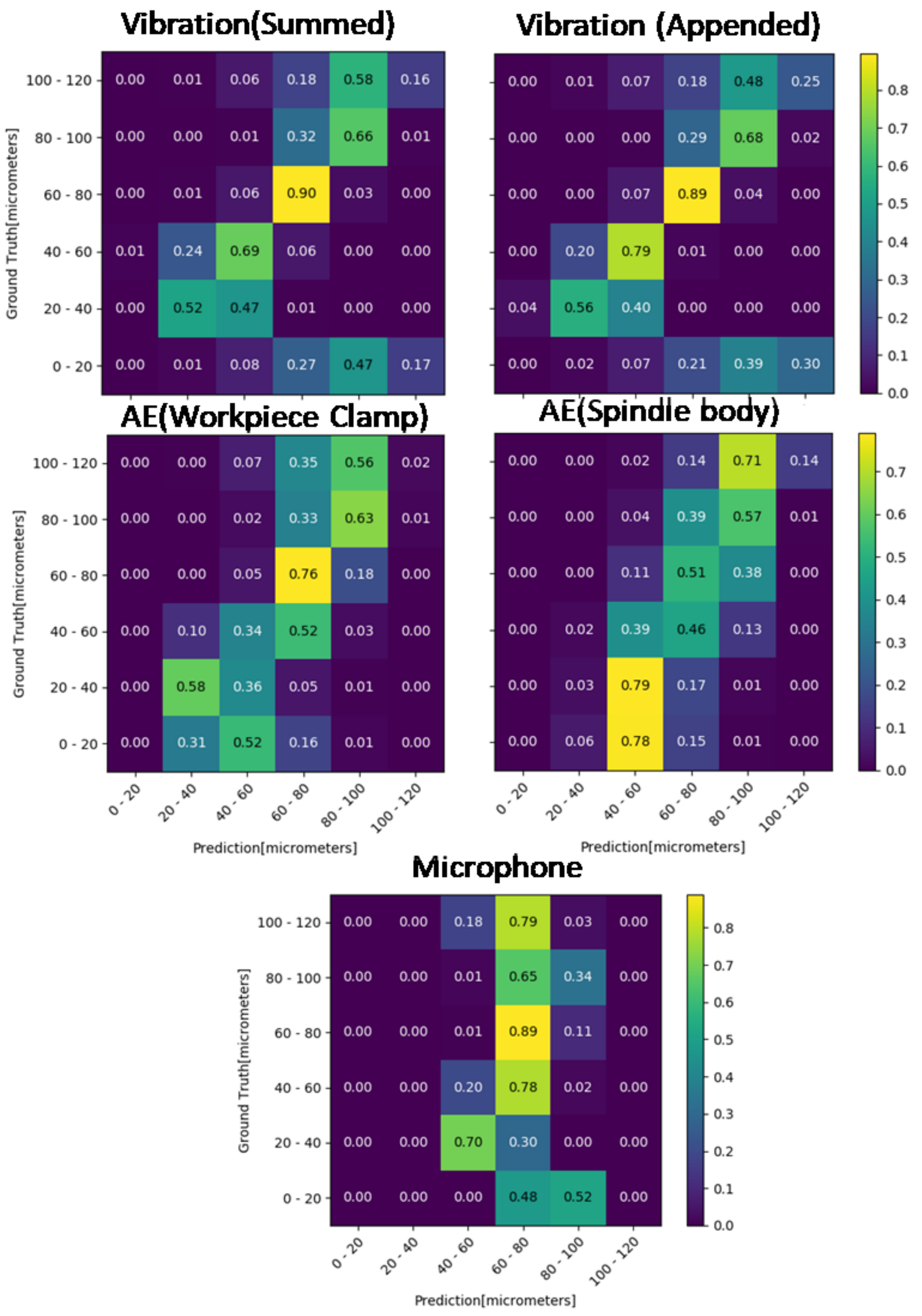


Through the use of pure ANNs, the best accuracy was achieved on the Vibration signal. See Figure 68. As the Figure shows, the model cannot predict well the extreme labels of the dataset. Performing better on the middle region, where a higher number of inputs are located.

The reasons for the range failure of the model can be better understood when analyzing the most noisy dataset used: the microphone signal. The result suggests that all the predictions done by the algorithm are biased to the mean flank wear of the train-set. Which indicates a lack of representation power of the prediction model. In other words, when the algorithm is not capable of interpreting the input spectrogram, its only choice is to average the predictions to the mean flank wear value on the dataset in order to minimize the mean value of the loss. This phenomena can also be confirmed when presenting the WME curve for the ANN models (Figure 69).

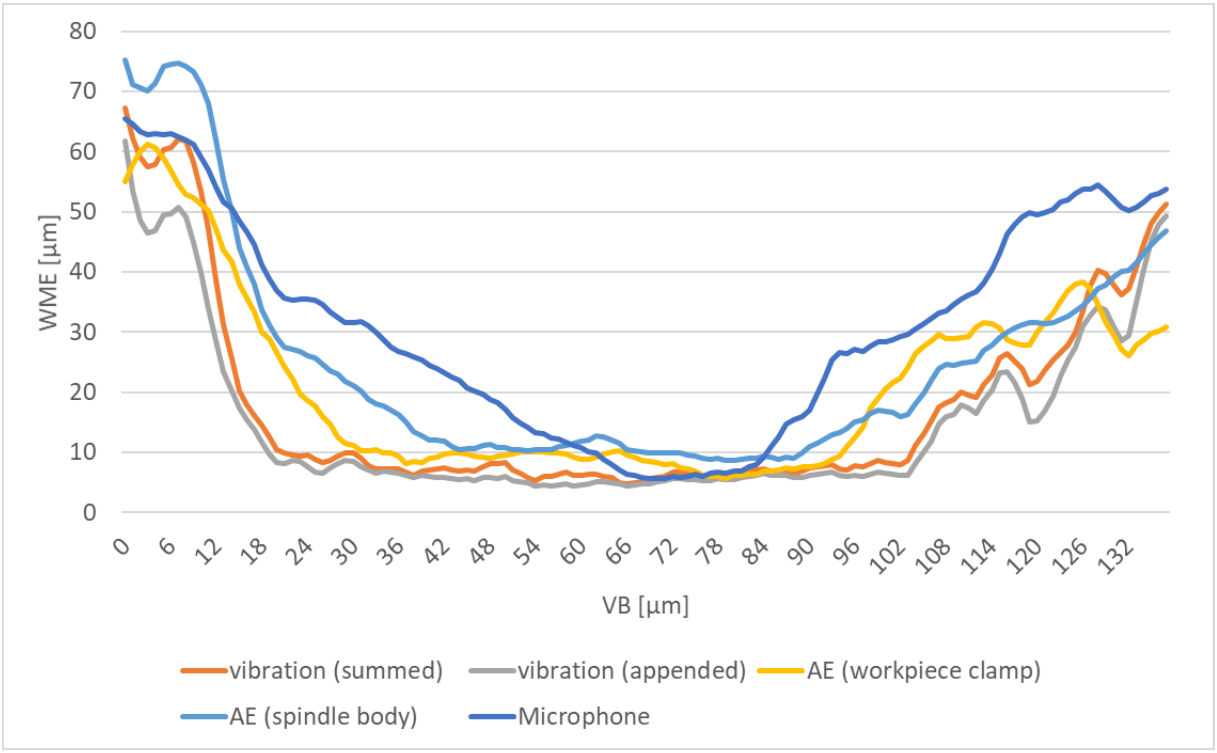
The general conclusion is that ANNs by their own are not able to generate good models for the flank wear prediction task, mainly on the beginning and end of tool lifespan. The models trained with the Vibration data were capable to distinguish different values of tool wear in the middle of the tool lifespan presenting a mean error constrained under $10\mu m$ on the interval $(20 < VB < 100)\ \mu m$ (Figure 69). On the other hand, the error rapidly increases outside this interval.

Figure 68 – Confusion matrix for ANN Regression (networks 6-10).



Datasets with lesser information or higher noise ratio presented smaller accurate prediction intervals. On this sense, after the Vibration, the AE sensor presented a better performance. The poorest performance, closer to averaging the predictions, was presented by the model trained with the microphone signal.

Figure 69 – WME curve over entire datasets for Networks 6 to 10.



Between both datasets generated from the vibration signal, the concatenated dataset presented a slightly better performance. So, summing the axes amplitudes – which could be seen as a feature selection mechanism – instead of maintaining them separated on the spectrogram caused small information loss.

Besides that, the AE signal from the workpiece clamp presented a better performance in comparison with the sensor installed on the spindle. However, for the advanced stages of flank wear, the curves presented in Figure 69 show the error curve for the spindle sensor more stable and accurate on the interval $100 < VB < 125\ \mu m$.

If the MSE on test-set (presented in Table 17) was used as the performance criteria for the model, the AE- sensors could be chosen as the best source of information for the problem. As the random error related to the flank wear measurement is $\pm 6\mu m$ (section 3.4), and the square root of the loss for Network 8 – for instance – is approximately $17\mu m$, this ANN result could be considered a good achievement. However, the accuracy value by its own does not present enough information about the performance of the model. The mean squared error, as shown in Figure 68 and Figure 69, has a performance biased to the mean flank wear value, which in resume is not the most important interval under observation.

This fact raised a big question mark related to the accuracy metric on this project. Reinforcing the importance of using the WME metric for evaluating the models. Figure 70 shows the Error curve and Figure 71 and predictions for the test-set using Network 2.

Figure 70 – WME for Network 2 – test-set.

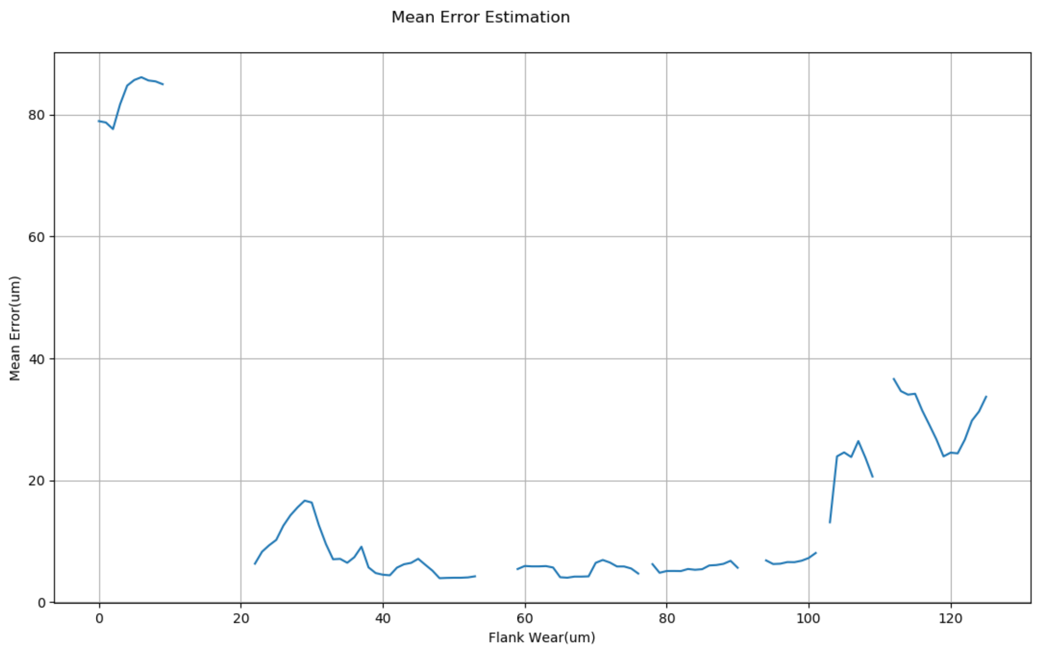
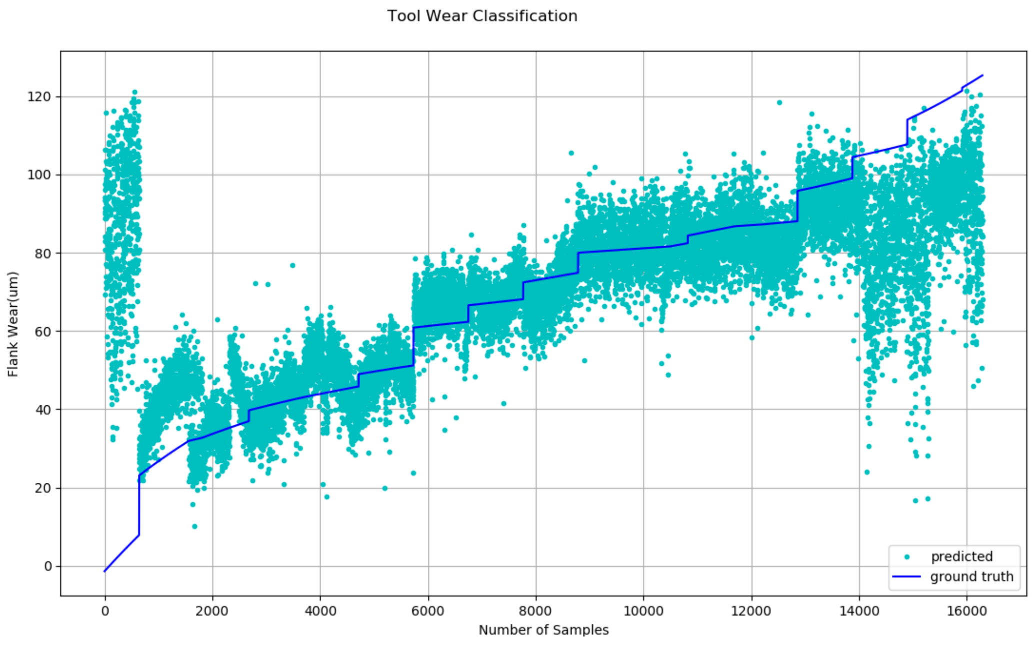


Figure 71 – Predictions generated by Network 2 – test-set.



By analyzing those figures, the performance bias of the algorithm becomes clear. Therefore, through the usage of WME, it is possible to evaluate the error in a more accurate way than only observing the resulting accuracy metric for the model.

6.2.3. ANN & Auto-encoder

Another architecture for the prediction models uses the encoded values generated from the networks presented in section 6.2.1. The architecture configuration is similar to the ANNs described on the previous section. However, the network now will receive the codes as inputs for the prediction task instead of the raw-spectrograms. Table 18 shows the network configurations employed to each dataset.

Table 18 – ANN + Auto-Encoder Regression Model Configurations.

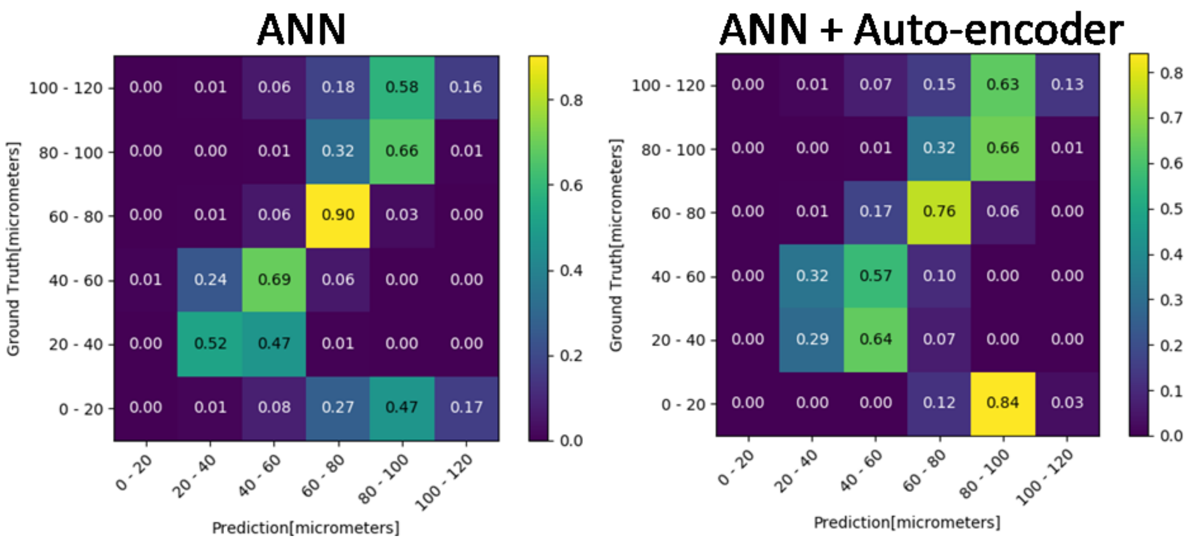
ANN + Auto-Encoder Regression Model Configurations				
ID	**Dataset**	**Layers**	**input shape**	**MSE test-set**
16	vib_sum codes	(10,50)-(10,16)-flat-drop0.5-16-8-4-1	(10,10)	509.45907
17	vib_x3 codes	(10,50)-(10,16)-flat-drop0.5-16-8-4-1	(10,10)	560
18	mic codes	(10,50)-(10,16)-flat-drop0.5-16-8-4-1	(10,10)	818.04535
19	ae_workpiece codes	(16,64)-(16,16)-flat-drop0.5-16-8-4-1	(16, 4)	461.7848
20	ae_spindle codes	(16,64)-(16,16)-flat-drop0.5-16-8-4-1	(16, 5)	742.36844

Once the auto-encoders already pre-select the potential features of the data, such ANNs are willing to overfit. In order to avoid such problem, a dropout layer was added to all networks.

However, the trained networks did not outperform the ANNs reported in the previous section. Figure 72 shows the comparison between the results gathered from the ANN and ANN + Auto-encoder for the vibration summed dataset (Networks 6 and 16).

Due to the problem reported in the end of Section 6.2.1, the model which presented the poorest accuracy rate was the spindle sensor data. The error curve (Figure 73) shows that the input codes does not present enough information for the neural network to predict the tool condition. Besides that, the best accuracy was achieved by the network trained on the Vibration (Summed) dataset. The remaining datasets presented a similar performance as the pure ANN models.

Figure 72 – Confusion Matrix for test-set on Networks 6 and 16.



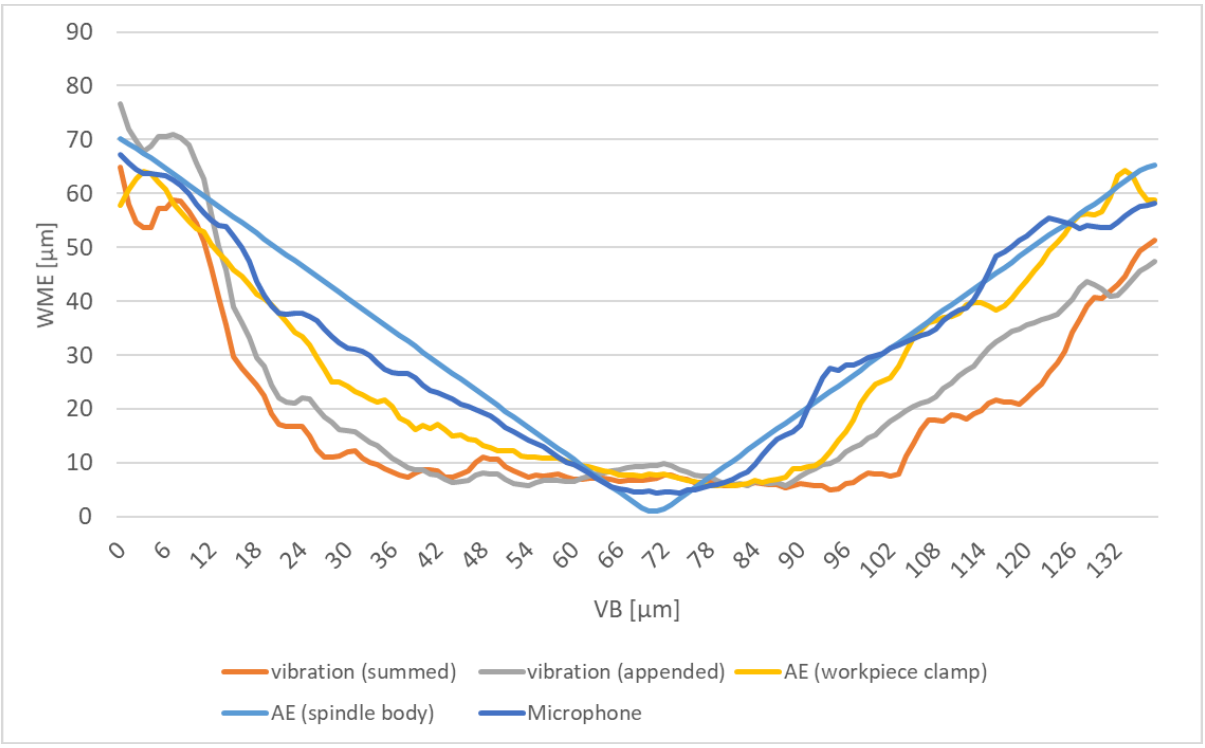
ANN (rows: Ground Truth [micrometers]; columns: Prediction [micrometers])

	0 - 20	20 - 40	40 - 60	60 - 80	80 - 100	100 - 120
100 - 120	0.00	0.01	0.06	0.18	0.58	0.16
80 - 100	0.00	0.00	0.01	0.32	0.66	0.01
60 - 80	0.00	0.01	0.06	0.90	0.03	0.00
40 - 60	0.01	0.24	0.69	0.06	0.00	0.00
20 - 40	0.00	0.52	0.47	0.01	0.00	0.00
0 - 20	0.00	0.01	0.08	0.27	0.47	0.17

ANN + Auto-encoder

	0 - 20	20 - 40	40 - 60	60 - 80	80 - 100	100 - 120
100 - 120	0.00	0.01	0.07	0.15	0.63	0.13
80 - 100	0.00	0.00	0.01	0.32	0.66	0.01
60 - 80	0.00	0.01	0.17	0.76	0.06	0.00
40 - 60	0.00	0.32	0.57	0.10	0.00	0.00
20 - 40	0.00	0.29	0.64	0.07	0.00	0.00
0 - 20	0.00	0.00	0.00	0.12	0.84	0.03

Figure 73 – WME curve over entire datasets for Networks 16 to 20.



Therefore, the auto-encoders may not be a good solution for improving performance at least on the scope of this project. On the other hand, they may be an useful tool when dealing with higher complex data formats or bigger datasets.

6.2.4. LSTM Classifier

The LSTMs are the last described architecture for the prediction modules. The method used two different approaches. The first one uses a classifier in order to predict the tool condition. Therefore, the network output layer possesses 6 sigmoid units mapping each of the assigned classes. See Table 19.

Table 19 – Classes addressed to each flank wear interval.

Flank Wear Interval (μm)	Label Binary Hot Code
0 – 20	[1, 0, 0, 0, 0, 0]
20 – 40	[0, 1, 0, 0, 0, 0]
40 – 60	[0, 0, 1, 0, 0, 0]
60 – 80	[0, 0, 0, 1, 0, 0]
80 – 100	[0, 0, 0, 0, 1, 0]
> 100 (tool failure)	[0, 0, 0, 0, 0, 1]

The architecture of the networks contain 2 LSTM layers as presented in chapter 2. In order to avoid overfitting, both those layers implement dropouts of 50%. The complete configuration for the LSTM classifiers is shown in Table 20.

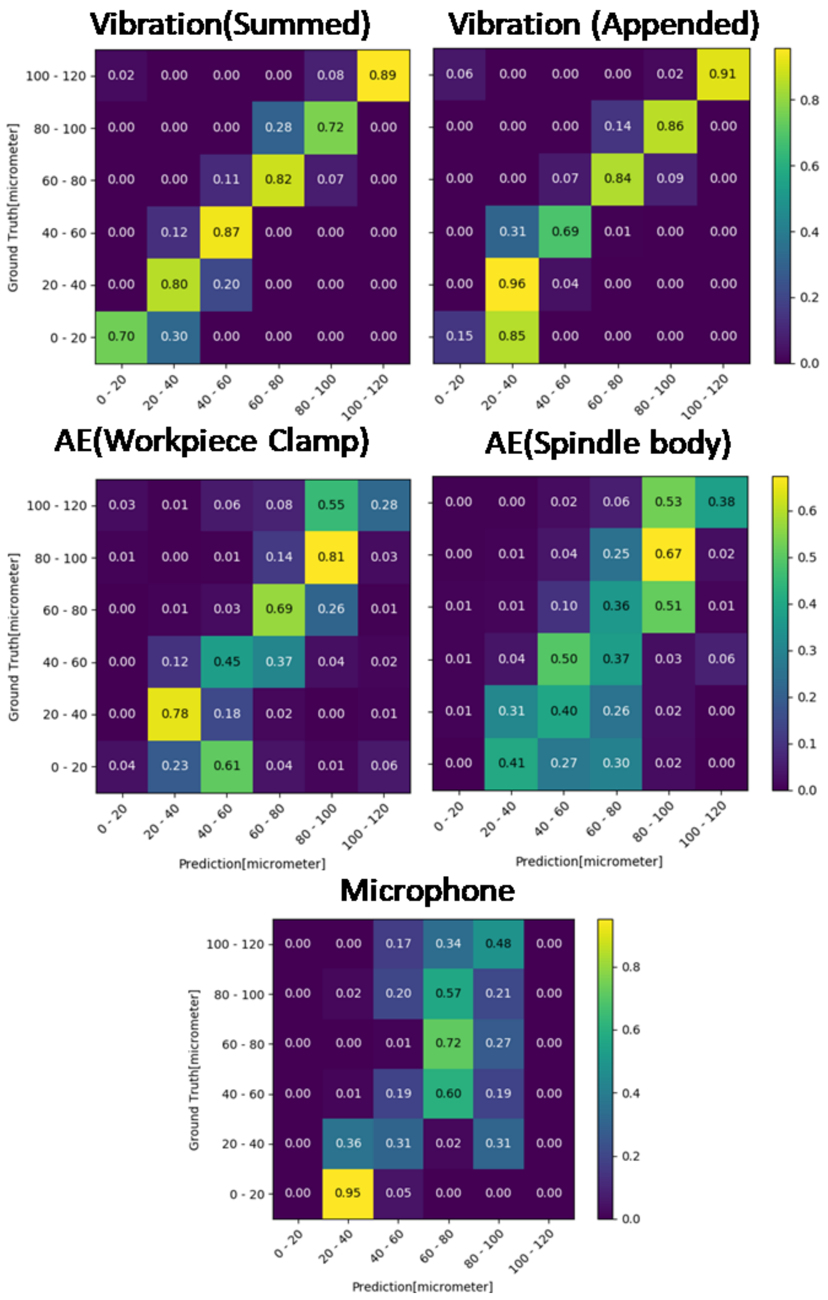
Table 20 – LSTM Classifier Model Configurations.

LSTM Classifier Model Configurations				
ID	Dataset	Layers	input shape	Categorical Accuracy (test-set)
26	vib_sum	(15,50)-64-drop0.5-64-drop0.5-32-16-6	(15,50)	0.78038
27	vib_x3	(15,50)-64-drop0.5-64-drop0.5-32-16-6	(15,150)	0.81738
28	mic	(15,50)-64-drop0.5-64-drop0.5-32-16-6	(15,50)	0.13881
29	ae_workpiece	(16,64)-64-drop0.5-64-drop0.5-32-16-6	(16,64)	0.65988
30	ae_spindle	(16,64)-64-drop0.5-64-drop0.5-32-16-6	(16,64)	0.56453

The confusion matrixes shown in Figure 74 present superior performance of the LSTM classifiers in comparison with all the previous presented regression models. The 6 different classes used, each one $20\mu m$ spaced, provided a way smaller error rate if compared to the regression networks. This fact proves that this approach is more suitable for an industrial TCM system than any of the previously presented models.

As observed in Figure 74, the performance pattern between the datasets repeated. Firstly, the models trained on vibration data provided again the best accuracy. Furthermore, the model trained on the Vibration(appended) dataset was the most accurate between all LSTM models. Unlike the regression models presented until now, the 2 classifiers trained on the vibration signal predicted well in all the stages of the tool lifespan. Which is a very important result, since the accuracy in predicting the last stages of tool wear is critical for the goals stablished for the project.

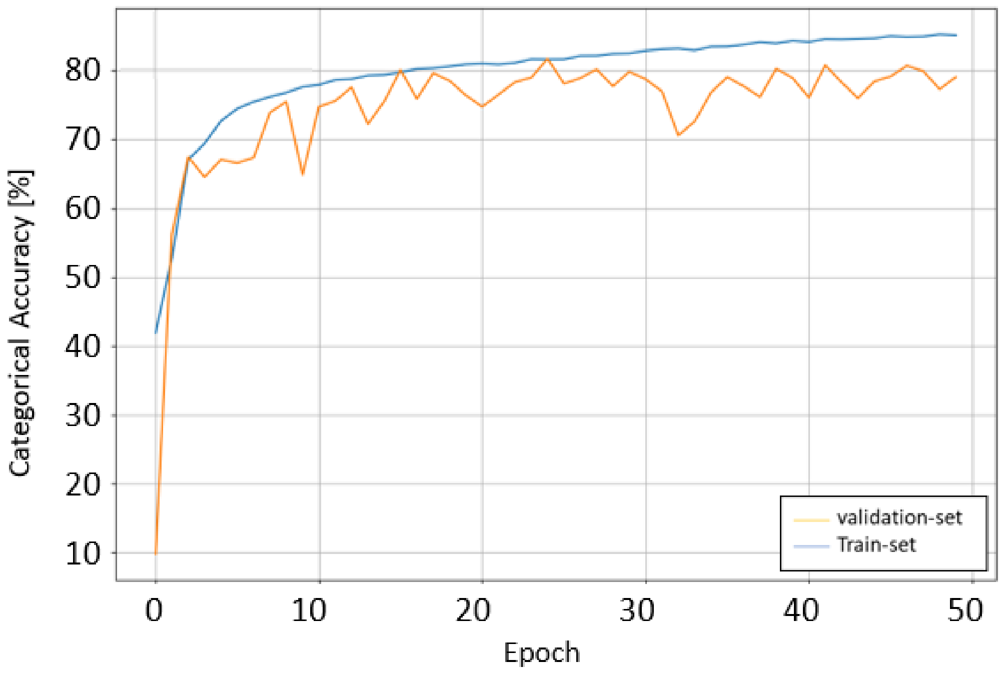
Figure 74 – Confusion Matrixes for LSTM classifiers (Networks 26-30) on test-set.



Although the AE models presented a better performance in this approach, they still present problems reaching all the flank wear range. As well as the previously reported approaches, the workpiece signal model presented a superior accuracy in comparison with the model trained with the spindle signal.

Figure 75 shows the training progress for network 27. The system accuracy develops well until the 25th epoch. After this epoch, the model starts to present overfit.

Figure 75 – Training curve for LSTM, Network 27.



6.2.5. LSTM Regression

The second approach is a regression model which, as the previously described ANNs, predicts the flank wear values through a linear output layer with one neuron. The remaining network configurations are similar to the ones presented for the Classifier approach. Table 21 shows each network configuration employed.

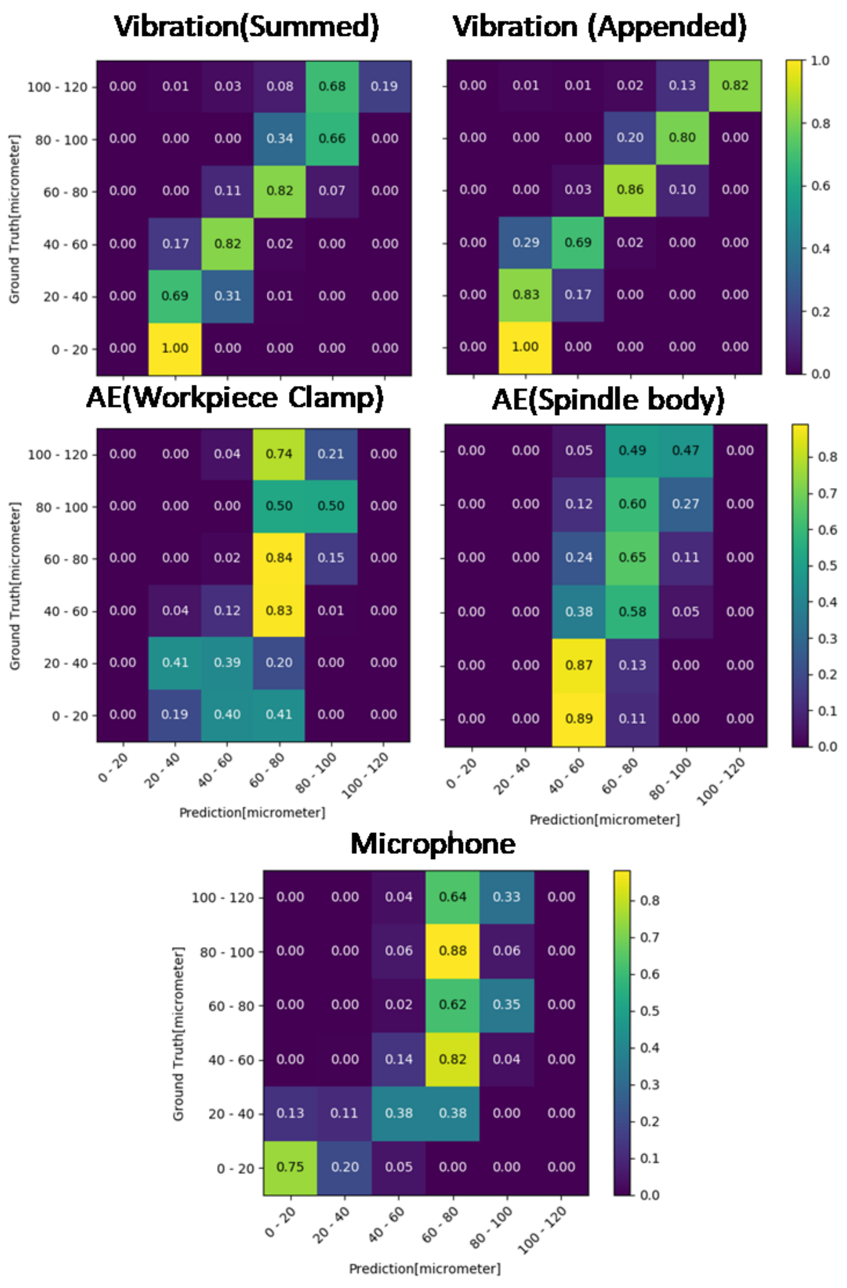
Figure 76 presents the results for the regression models trained. The charts show that the regression models are not as precise as the classification models. However, the LSTMs achieved a superior performance in comparison with the approaches using vanilla ANN and ANNs + Auto-encoders.

Table 21 – LSTM Regression Model Configurations.

LSTM Regression Model Configurations

ID	Dataset	Layers	input shape	MSE (test-set)
21	vib_sum	(15,50)-32-drop0.5-32-drop0.5-16-8-1	(15,50)	284.29948
22	vib_x3	(15,50)-32-drop0.5-32-drop0.5-16-8-1	(15,150)	176.15272
23	mic	(15,50)-32-drop0.5-32-drop0.5-16-8-1	(15,50)	771.1928
24	ae_workpiece	(16,64)-32-drop0.5-32-drop0.5-16-8-1	(16,64)	420.97011
25	ae_spindle	(16,64)-32-drop0.5-32-drop0.5-16-8-1	(16,64)	441.0024

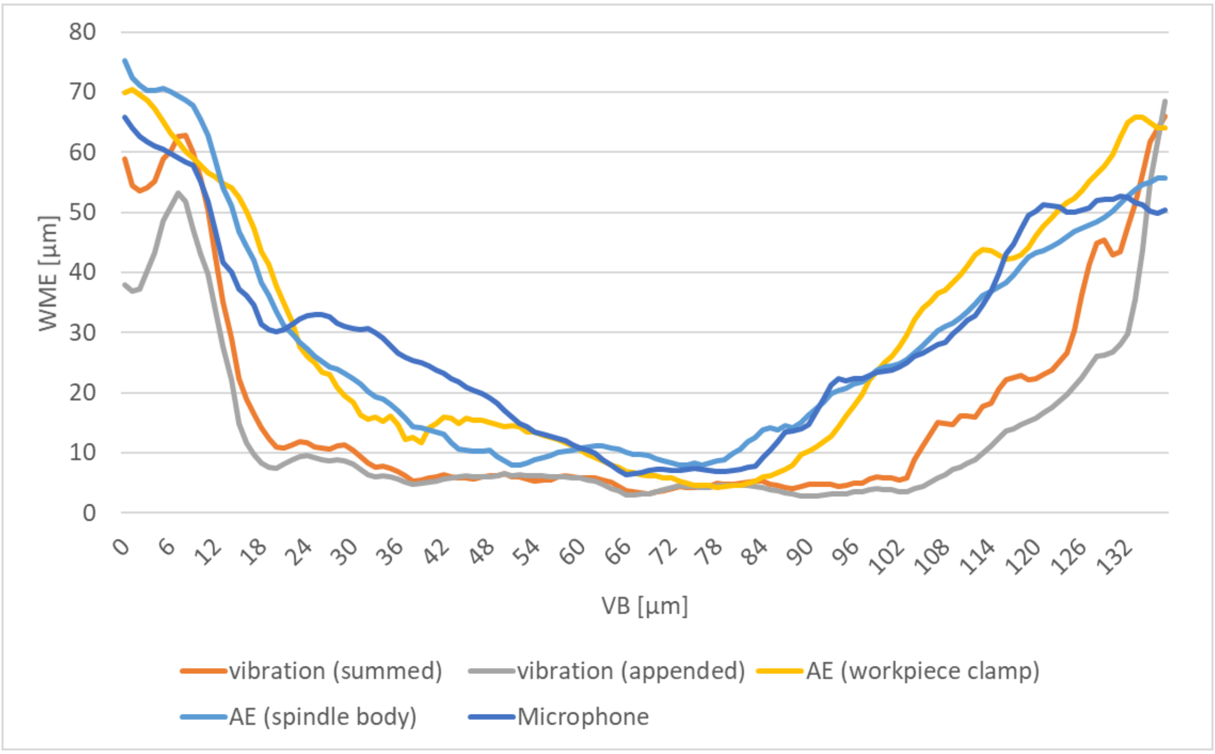
Figure 76 – Confusion Matrixes for LSTM Regression (Networks 21-25) on test-set.



The pattern among the signals repeated once more. From which, it can be concluded that the signal which presents the most relevant amount of information for predicting tool wear comes from the vibration sensor. However, as the other regression models, these networks does not present high accuracy on the extremities of the flank wear curve.

The models trained with the AE signal also presented some problem on predicting values on the beginning and end of the flank wear curve, as it can be seen in Figure 77. On the other hand, the error curve for the model trained on the microphone data followed the error curve of the AE models. Such behavior was not observed in the ANN models, where the error was bigger for the microphone signal. Such fact suggests that the LSTMs can better interpret noisy data, when compared to the other reported approaches.

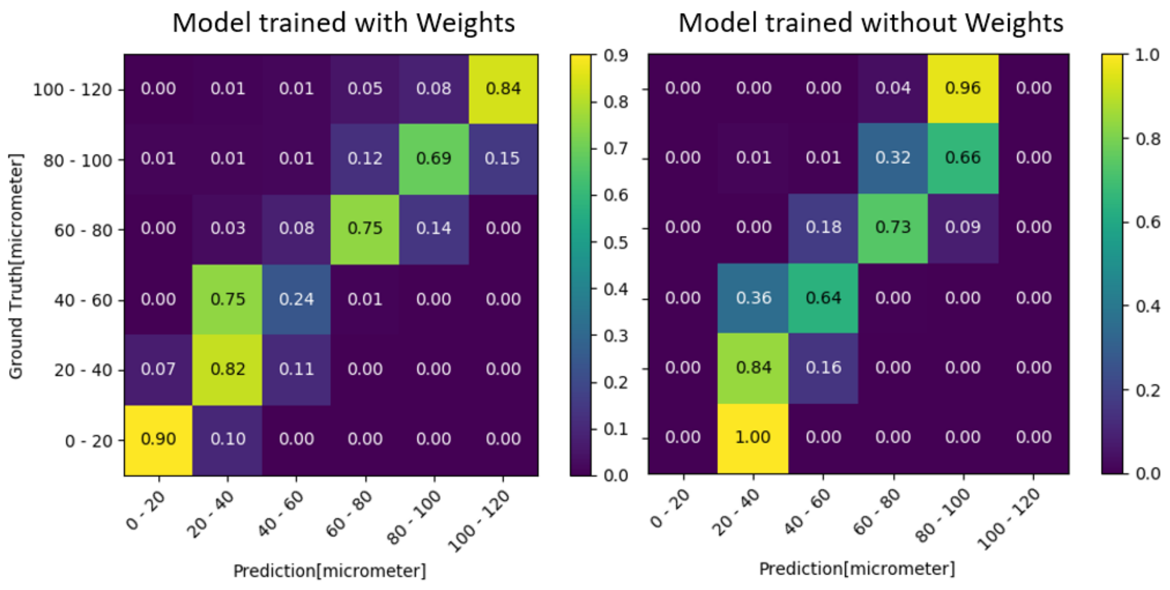
Figure 77 – WME curve over entire datasets for Networks 21 to 25.



Despite the fact that the LSTM regression models presented a superior performance, by analyzing Figure 77 it is clear how the performance of the network is still biased to the mean flank wear. In order to overcome this problem, as explained in section 6.1.4, a network model was trained with weights giving higher priorities to the beginning and end of the tool flank wear curve. Figure 78 presents the comparison between two models, one trained with weights and one without.

Figure 78 shows that, by addressing higher priority to the extremities of the tool lifespan, the resulting network has a better performance on those regions. On the other hand, the overall accuracy – on the middle region – is slightly damaged.

Figure 78 – LSTM Regression trained with and without weights for Vibration(summed) dataset.



Model trained with Weights:

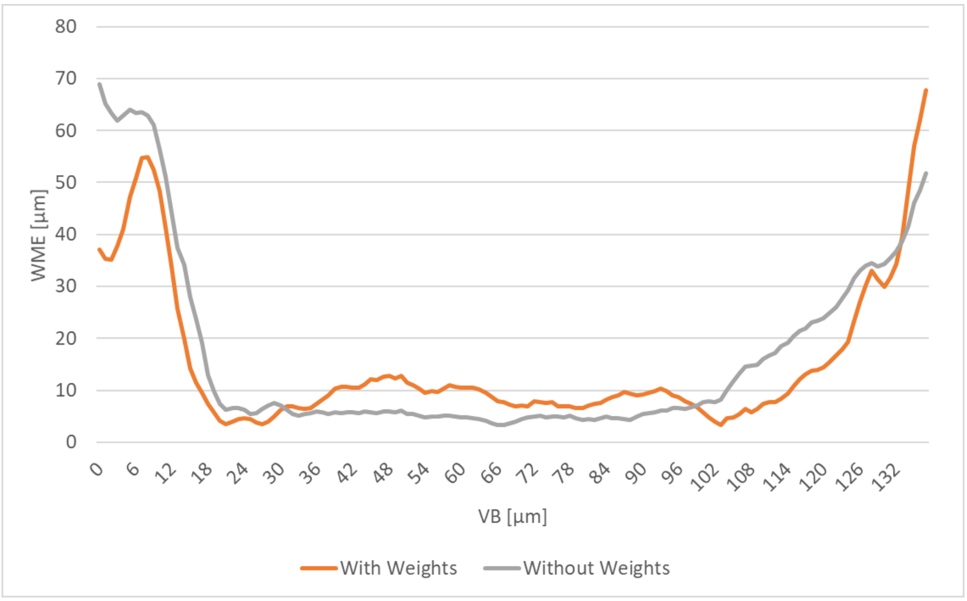
Ground Truth \ Prediction	0 - 20	20 - 40	40 - 60	60 - 80	80 - 100	100 - 120
100 - 120	0.00	0.01	0.01	0.05	0.08	0.84
80 - 100	0.01	0.01	0.01	0.12	0.69	0.15
60 - 80	0.00	0.03	0.08	0.75	0.14	0.00
40 - 60	0.00	0.75	0.24	0.01	0.00	0.00
20 - 40	0.07	0.82	0.11	0.00	0.00	0.00
0 - 20	0.90	0.10	0.00	0.00	0.00	0.00

Model trained without Weights:

Ground Truth \ Prediction	0 - 20	20 - 40	40 - 60	60 - 80	80 - 100	100 - 120
100 - 120	0.00	0.00	0.00	0.04	0.96	0.00
80 - 100	0.00	0.01	0.01	0.32	0.66	0.00
60 - 80	0.00	0.00	0.18	0.73	0.09	0.00
40 - 60	0.00	0.36	0.64	0.00	0.00	0.00
20 - 40	0.00	0.84	0.16	0.00	0.00	0.00
0 - 20	0.00	1.00	0.00	0.00	0.00	0.00

The results show an improvement of more than 50% accuracy on the extremities of the curve for the analyzed dataset. Figure 79 shows the error curve for the entire dataset for both the approaches.

Figure 79 - WME for LSTM regression comparing the usage of weighting.



6.2.6. Prediction Results

Although the spindle vibrates and produces a high amount of noise, the sensors installed on the spindle body proved to show good amount of information about the process condition. Such fact is even more clear if we consider that the best prediction models used the input from the accelerometer sensor.

By saying so, positioning the sensors on the spindle body is not only more flexible when analyzed from the production perspective, but it is also a powerful tool in order to capture the flank wear progress as well as it may be useful for studying other phenomena too.

Still related to the sensors, training the neural networks helped interpreting the performance of each of those signals on the studied problem. Firstly, the vibration signal provided by the accelerometer was the most reliable source for understanding the flank wear development. Unlike reported in the recent literature, it outperformed AE-sensors and ultra-sonic emission for capturing the flank wear. This result suggests such signals can be further studied in future works on the field.

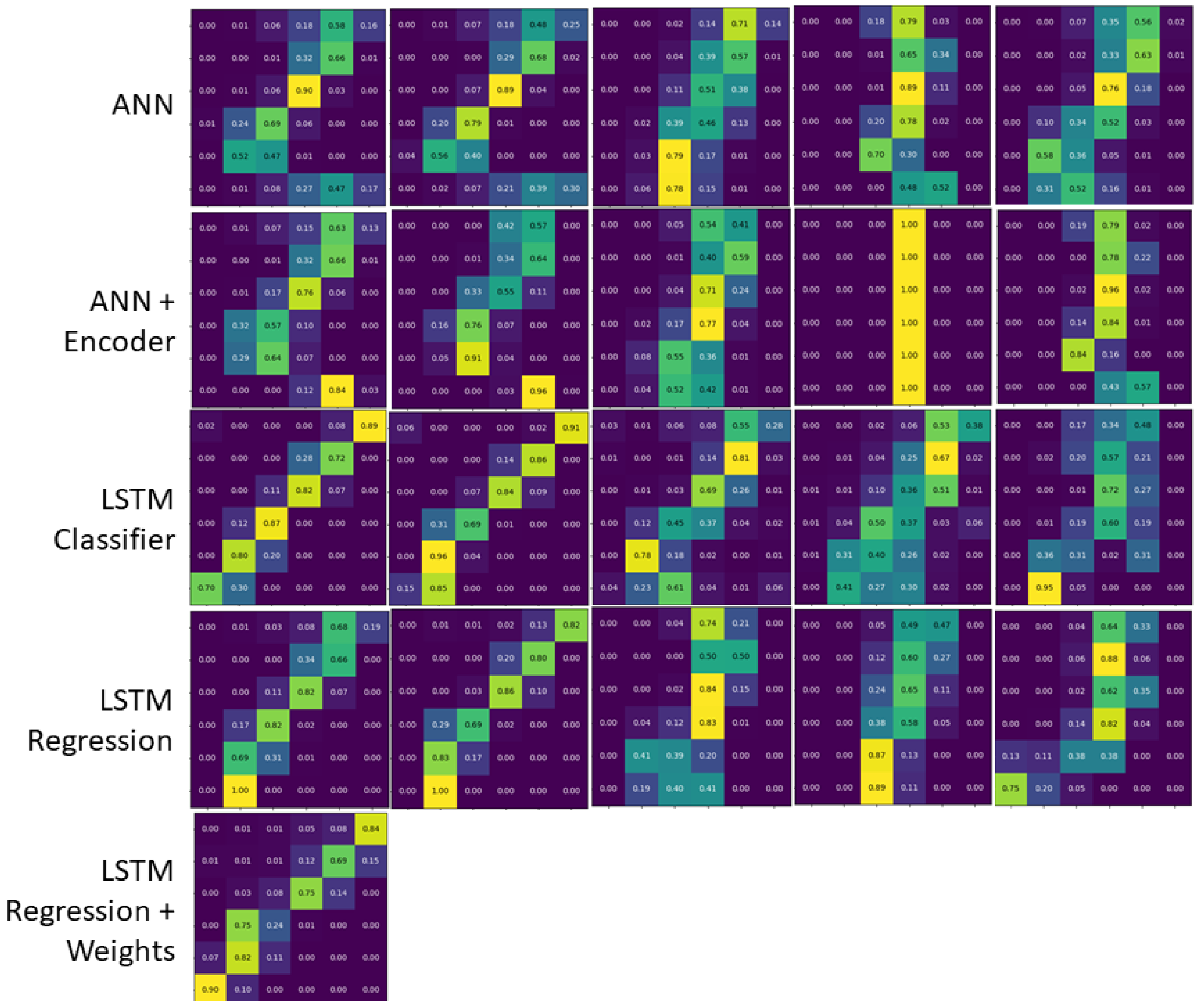
After this, the microphone data provided a very noisy behavior and not much relevant information. At least not in the current position where the sensor was installed. As a suggestion, one option could be repositioning such sensor in a way that it can capture more information about the process.

By looking at the resulting confusion matrixes, the performance of the LSTMs are clearly superior in comparison with the vanilla or encoded ANN prediction models. See Figure 80. Once those networks can memorize features found in the input vectors, they achieved – mainly on the extremities of the flank wear curve – a more accurate prediction of the cutters´ condition.

Besides this fact, the LSTM classification approach achieved the best result between all prediction models analyzed on the project. Regarding the regression approach, Figure 80 proves that the performance bias of the network can be tackled using weights in order to train the regression models. This weight training result has shown that the LSTM regression can present similar performance as the LSTM classifiers achieved. After all, both approaches can be considered suitable for an online TCM system.

Figure 80 – Confusion Matrixes (test-set) of all Prediction Models trained in the project.

[redacted]



7. CONCLUSION

This document presented a complete system for tool wear prediction in high precision milling process. The project englobed sensor setup, parameter determination, experiments and signal analysis. Through the experiments performed on the machine, various properties of the tool wear development in milling were observed.

Firstly, the acquisition program developed and reported in chapter 4 proved to be a new powerful resource available in order to capture information of processes using high spindle speed. Furthermore, milling processes on the industrial environment usually employ tools with multiple cutters and high spindle speed machine tools, reinforcing the importance of acquisition systems capable of recording high frequency signals.

The reported experiments gathered data from different sources: two AE-sensors, one installed in the workpiece clamp and the other on the spindle body; one vibration sensor installed in the spindle body; one ultra-sonic microphone installed on the machining chamber; 4 encoder position readers: X, Y, Z, spindle; the spindle power consumption; and the Spike tool holder, which recorded the Forces applied to the tool.

Related to the AE-sensors, the NN could interpret better the signals recorded from the workpiece clamp sensor in comparison with the spindle sensor. One possible reason is that such sensor does not get high influence from the spindle vibration, so it can better capture the process emission. However, both sensors achieved poor performance on the tool wear prediction task.

On the preprocessing, all sensor data was extracted using FFT. The tool flank was directly measured via microscope by accessing 3 different regions of each tooth. Once the employed tool has 3 teeth, the procedure measured the flank wear length on 9 different regions. From which, the maximum value among those 9 points was adopted for labeling the data.

In order to determine the best architecture and sensor for the problem, the system tested 3 different approaches for the prediction models: a vanilla Neural Network; a combination of ANN with Auto-encoder and LSTM architecture. The results showed the ANN and ANN with Auto-encoder did not have enough representation power for the prediction task.

A last approach used LSTM networks. On the classifier approach, the models were tuned to classify the inputs into 6 different states of tool wear divided in the range of 0 to $120\mu m$. Such method presented a high performance on the vibration data. The accuracy was higher than 80% in almost all the classes for this dataset.

On the other hand, the approach using LSTM for regression presented accuracy problem in the beginning and end of the tool wear curve, once the amount of data available to the model is smaller in this interval. Therefore, the training algorithm tends to give higher priority to the middle interval of the tool.

In order to overcome this problem on the regression networks, an approach used weights in order to raise the priority of the pairs where the flank wear $30 > VB > 100$. This procedure forced the networks to improve the performance on the referred interval. The accuracy on those regions improved more than 50% in the tested dataset.

As a result, both approaches using LSTMs could be considered for a further implementation of a TCM system. This fact shows the potential of deep-learning in analyzing sensor data. However, further researches involving new techniques must still be performed in order to improve the reported results.

For future works, a first factor to mention is the small dataset size. On the current project, 4 tools and 2 different process parameter configurations were employed. In order to better generalize this result, further works must test a higher volume of different parameters, tool paths, tool types, etc.

Besides that, other networks like convolutional networks and fuzzy networks can be explored in order to improve the achieved results. Furthermore, other techniques like training models using a combination of multiple signals can also present improvements. At least, the results here presented can serve as a guideline for future studies on the field. After all, the dataset are still not fully explored, as some signals recorded in the experiments were not used.

8. REFERENCES

[1] Institut für Produktionstechnologie - Fraunhofer IPT. (2020, January 6). Retrieved from https://www.ipt.fraunhofer.de/

[2] Palanisamy, P., Rajendran, I. & Shanmugasundaram, S. Prediction of tool wear using regression and ANN models in end-milling operation. *Int J Adv Manuf Technol* **37,** 29–41 (2008). https://doi.org/10.1007/s00170-007-0948-5

[3] Zhou, Y., Xue, W. Review of tool condition monitoring methods in milling processes. *Int J Adv Manuf Technol* **96,** 2509–2523 (2018). https://doi.org/10.1007/s00170-018-1768-5

[4] Rehorn, A., Jiang, J. & Orban, P. State-of-the-art methods and results in tool condition monitoring: a review. *Int J Adv Manuf Technol* **26,** 693–710 (2005). https://doi.org/10.1007/s00170-004-2038-2

[5] Cao, H., Zhang, X., & Chen, X. (2017). The concept and progress of intelligent spindles: A review. *International Journal of Machine Tools and Manufacture*, *112*, 21–52. doi: 10.1016/j.ijmachtools.2016.10.005

[6] Russell, S. J. (1995). *Artificial intelligence: a modern approach*. Prentice Hall.

[7] Samuel, A. L. (1988). Some Studies in Machine Learning Using the Game of Checkers. I. *Computer Games I*, 335–365. doi: 10.1007/978-1-4613-8716-9_14

[8] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu and M. Stanley, "A brief survey of machine learning methods and their sensor and IoT applications," *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Larnaca, 2017, pp. 1-8. doi: 10.1109/IISA.2017.8316459

[9] Machine learning. (2020, January 23). Retrieved from https://en.wikipedia.org/wiki/Machine_learning

[10] Al-Zubaidi, S., Ghani, J. A., & Haron, C. H. C. (2011). Application of ANN in Milling Process: A Review. *Modelling and Simulation in Engineering*, *2011*, 1–7. doi: 10.1155/2011/696275

[11] Kilickap, E., Yardimeden, A., & Çelik, Y. H. (2017). Mathematical Modelling and Optimization of Cutting Force, Tool Wear and Surface Roughness by Using Artificial Neural Network and Response Surface Methodology in Milling of Ti-6242S. *Applied Sciences*, *7*(10), 1064. doi: 10.3390/app7101064

[12] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, "Convolutional Neural Networks for Speech Recognition," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533-1545, Oct. 2014. doi: 10.1109/TASLP.2014.2339736

[13] Chen, J., Chen, J. An artificial-neural-networks-based in-process tool wear prediction system in milling operations. *Int J Adv Manuf Technol* **25,** 427–434 (2005). https://doi.org/10.1007/s00170-003-1848-y

[14] Başccedil F. “On-line prediction of tool wears by using methods of artificial neural networks and fuzzy logic” [J]. Scientific Research and Essays, 2010, 5(19): 2883-2888.

[15] Ruder, & Sebastian. (2017, June 15). An overview of gradient descent optimization algorithms. Retrieved from https://arxiv.org/abs/1609.04747

[16] Li Deng and Dong Yu (2014), "Deep Learning: Methods and Applications", Foundations and Trends® in Signal Processing: Vol. 7: No. 3–4, pp 197-387. http://dx.doi.org/10.1561/2000000039

[17] Deep learning. (2020, January 23). Retrieved from https://en.wikipedia.org/wiki/Deep_learning

[18] Yin, Wenpeng, Kann, Katharina, Yu, & Hinrich. (2017, February 7). Comparative Study of CNN and RNN for Natural Language Processing. Retrieved from https://arxiv.org/abs/1702.01923

[19] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. doi: 10.1109/72.279181

[20] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

[21] Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. Cambridge, MA: The MIT Press.

[22] Al-Sulaiman, F. A., Baseer, M. A., & Sheikh, A. K. (2005). Use of electrical power for online monitoring of tool condition. *Journal of Materials Processing Technology*, *166*(3), 364–371. doi: 10.1016/j.jmatprotec.2004.07.104

[23] Čuš, F., & Župerl, U. (2011). Real-Time Cutting Tool Condition Monitoring in Milling. *Strojniški Vestnik – Journal of Mechanical Engineering*, *57*(2), 142–150. doi: 10.5545/sv-jme.2010.079

[24] Girardin, F., Rémond, D. & Rigal, J. A new method for detecting tool wear and breakage in milling. *Int J Mater Form* **3,** 463–466 (2010). https://doi.org/10.1007/s12289-010-0807-z

[25] Sadilek, M, Kratochvil, J, Petru, J. Cutting tool wear monitoring with the use of impedance layers. Tehnički Vjesnik [Tech Gaz] 2014; 21: 639–644.

[26] Yu, J. Online tool wear prediction in drilling operations using selective artificial neural network ensemble model. *Neural Comput & Applic* **20,** 473–485 (2011). https://doi.org/10.1007/s00521-011-0539-0

[27] Dong, J., Subrahmanyam, K.V.R., Wong, Y.S. *et al.* Bayesian-inference-based neural networks for tool wear estimation. *Int J Adv Manuf Technol* **30,** 797–807 (2006). https://doi.org/10.1007/s00170-005-0124-8

[28] XIAOLI , L., YINGXUE , Y. & ZHEJUN , Y. On-line tool condition monitoring system with wavelet fuzzy neural network. *Journal of Intelligent Manufacturing* **8,** 271–276 (1997). https://doi.org/10.1023/A:1018585527465

[29] O. Geramifard, J. Xu, J. Zhou and X. Li, "Multimodal Hidden Markov Model-Based Approach for Tool Wear Monitoring," in *IEEE Transactions on Industrial Electronics*, vol. 61, no. 6, pp. 2900-2911, June 2014.
doi: 10.1109/TIE.2013.2274422

[30] Wang, M., Wang, J. CHMM for tool condition monitoring and remaining useful life prediction. *Int J Adv Manuf Technol* **59,** 463–471 (2012). https://doi.org/10.1007/s00170-011-3536-7

[31] S. Dolinšek, Mechanism and types of tool wear; particularities in advanced cutting materials, *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 19, 2006,pp.11-18.

[32] Yu Y.: "*Untersuchungen zur Entwicklung einer parametrierbaren eingebetteten Datenaus-wertung für Smart Components*". RWTH Aachen. Aachen. 2018

[33] High Performance Motion Solutions. (2020, January 26). Retrieved from https://www.triamec.com/de/TAM-SDK.html

[34] The HDF5® Library & File Format. (2020, January 13). Retrieved from https://www.hdfgroup.org/solutions/hdf5/

[35] Giriraj, B., V. P. Raja, R. Gandhinadhan, and R. Ganeshkumar. 2006. “Prediction of Tool Wear in High Speed Machining Using Acoustic Emission Technique and Neural Network.” *Journal of Engineering and Materials Sciences* 13 (4): 275–280.

[36] Kingma, P., D., Jimmy, & Ba. (2017, January 30). Adam: A Method for Stochastic Optimization. Retrieved from https://arxiv.org/abs/1412.6980

[37] Yan, J., Meng, Y., Lu, L., & Li, L. (2017). Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance. *IEEE Access*, *5*, 23484–23491. doi: 10.1109/access.2017.2765544

[38] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, Swinoujście, 2018, pp. 117-122.
doi: 10.1109/IIPHDW.2018.8388338

9. APPENDIX A – TAMA CLIENT FUNCTIONALITY DESCRIPTION

This document lists the desired functionalities to be incorporated on the Tama Client software programming for real-time acquisition. On the table below, all desired functional and performance requirements raised for the software project are described.

The functions here presented were generated by the author together with other team members who also require the use of this acquisition program on their projects.

F1: Connect to Trialink			
Description: the system must be able to connect to the main Trialink adapter (TLC-Card) in order to access the information available inside the ring network.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF1.1: Connection status	the system must show the user the status of the connection with the network.	interface	(X)
NF1.2: Release connection	Once the system finishes the tasks, or shuts down for any reason, it must – with all its effort – release the connection with the link, freeing the network to connect to another software.	performance	(X)

F2: Show Parameter Tree			
Description: Once the system accessed the link, it must show the user all the devices connected in the ring, and all the readable parameters contained inside each one of the devices found.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF2.1: Tree interface	the devices, categories and parameters must be organized in a tree mode, starting with the devices on the ring and going until the end-point parameters of the tree.	interface	(X)
NF2.2: Read-only	The system must show the user only readable parameters of a device. That is, it must find a way to eliminate all unreadable parameters and dead nodes of the device tree in order to simplify the view.	Performance	(X)

F3: Acquisition Programing			
Description: the user will be able to set one or multiple simultaneous acquisitions to be performed by the system. Using the interface tools provided in F2, the user will select a group of variables and the sampling rate for each of those groups.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF3.1: Show acquisitions	The system must show to the user a list of selected registers for each acquisition, specifying the sampling rate.	interface	( )
NF3.2: Inclusion/exclusion policy	The system must allow one option where - with at maximum - two clicks, the user selects the desired registers to be included or excluded from the acquisition group list	interface	( )
NF3.3: reset programming	The user will be able to reset the entire acquisition programing via interface, and restart selecting the channels for acquisition	interface	(X)

F4: Save Acquisition Setup			
Description: the user has the option to save and load the acquisition programing done on F3. With this, repeated acquisitions with same setup must be programmed only once and then they can be loaded back on the system multiple times.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF4.1: Loading last program	The system will always try to load the preferences programmed in the last time the software was used	performance	( )
NF4.2: External edition	The format used for the setup must allow external editions. On this way some file type like Jason or XML must be adopted for the task.	performance	(X)

F5: Acquire data on Trialink			
Description: The system will configure and perform the programmed data acquisition in the ring network. And load on software all required information			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF5.1: Real Time Constraint	The ring real-time flow of data must be preserved. This implies that the mechanism to acquire data needs to work under real-time constraints.	performance	(X)
NF5.2: Data handling	The system will preserve the types of each data received, i.e., there will be no data conversion inside the software unless critically required. On this way, avoiding overheads and performance problems due to data conversion.	performance	(X)
NF5.3: Acquisition Status	The user will receive constantly information about the status of the in-process acquisition via interface. Problems on the link or software will need to be reported as soon as possible.	interface	(X)
NF 5.4: Show acquired Values	The user, via interface, will be able to see the last values acquired from the link.	interface	(X)

F6: Save Acquired data			
Description: the system must be able to create files, handle buffered values in RAM, and save all the acquired data locally.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF6.1: Avoiding Overflow	The system must have protection mechanisms to handle data overflow on RAM memory, i.e., the system will prevent the failure by exceeding in-memory data.	interface	(X)
NF6.2: Data Conservancy	The system must avoid the loss of data due to the lack of performance on this functionality preserving, in this way, the programmed acquisition sampling rate provided by the user.	performance	( )
NF 6.3: In memory data register	The user will receive information in the interface regarding the current amount of data stored in memory.	interface	( )
NF6.4: Data handling	The system will preserve the types of each data received, i.e., there will be no data conversion inside the software unless critically required. This will avoid overhead and performance problems when saving data locally.	performance	(X)

F7: Constraint Program Insertion			
Description: The system will reserve a specific area on the code which may be used by future programmers to insert specific desired behaviors, where it is possible to observe data and perform some specific actions automatically, which could be only possible via manual operation. Ex.: stop once 10 minutes of data were recorded.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory

F8: Log data maintenance			
Description: The system will show and save, a log file containing all events which happened during the acquisition.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory
NF 8.1: In-process check-up	The user will be able to open the log file during the process to check the past events that happened on the incurrent acquisition	(performance)	(X)
NF 8.2: Interface Events	The system will maintain on the interface only the most recent events which happened on the system. (the rest will be available on the log file)	(interface)	(X)

F9: Output Data Type			
Description: The output files must be saved in a format which allows the immediate visualization by the user after the files are closed.			
Not-Functioning Requirements			
Name	Constraint	Category	Mandatory

10. APPENDIX B – TAMA CLIENT SOFTWARE PROJECT

This document is aimed to provide details about the software project developed based on the Functionality description presented in Appendix A. As described in Chapter 4, the project tried to accomplish all listed functionalities while keeping track of some important properties the software must maintain. The final version of the class diagram can be seen in Figure 81.

In order to accomplish the acquisition task, the software is multithreaded. This decision also helps avoiding interchanging problems and bottlenecks on the system. By splitting the tasks, the processes responsible for keeping control of the system status can also analyze each software module independently, easing error diagnoses and speeding up correction mechanisms.

Therefore, the acquisition software was separated into 4 modules. Each one of them is responsible for a limited number of tasks. In other words, there is no redundancy on the system functions between each of those modules.

10.1. Interface Package

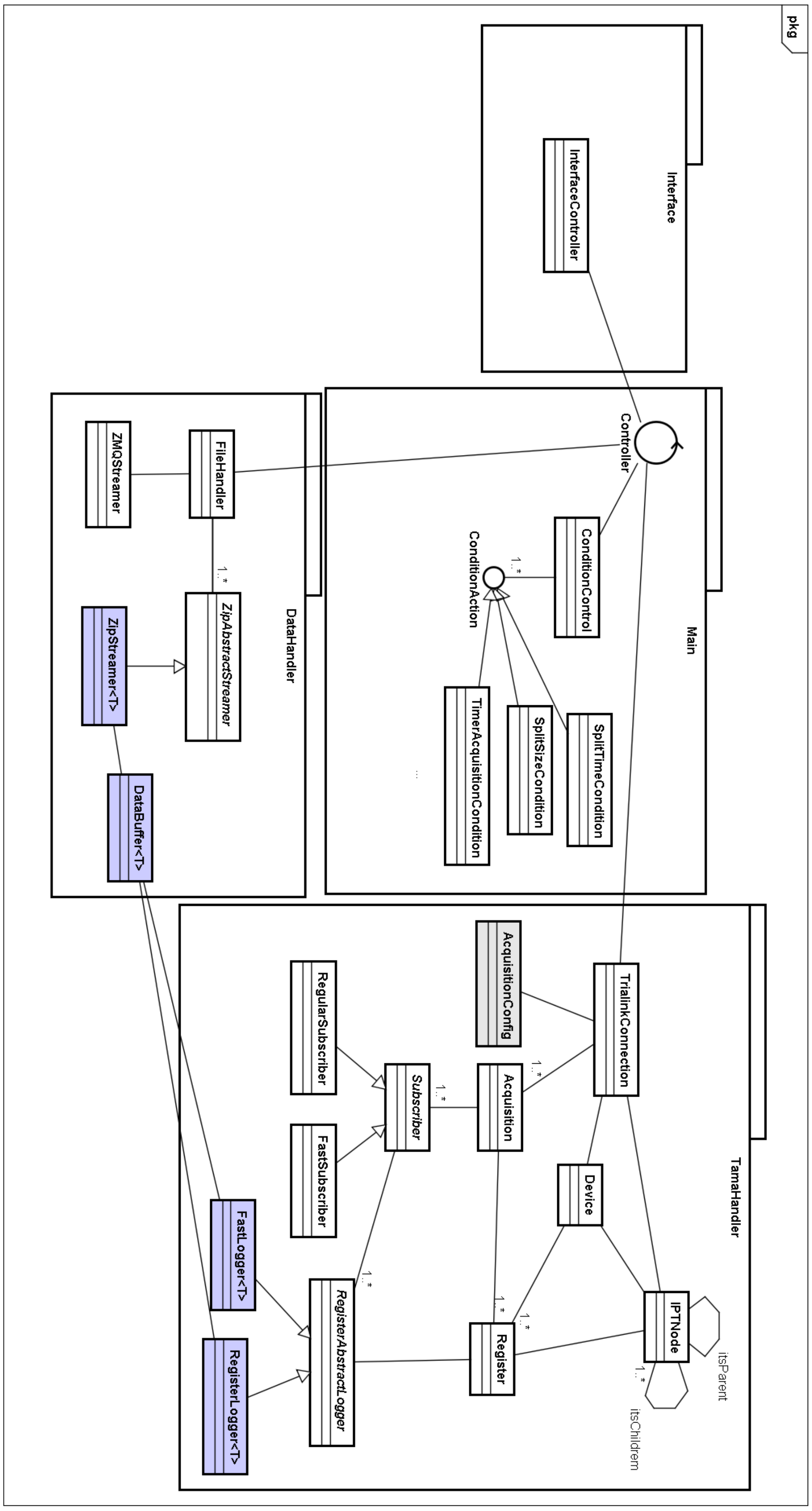
This package is responsible for handling all operations involving the user. On this package, the window and all graphical items are located. As it can be seen, the project implementation was carried out such a way that the back-end and front-end were completely separated from each other. Therefore, all requests and commands from the GUI go to the Controller class, which then access further classes and gets the required data. This allow not only the complete replacement of the GUI – if needed – but also ensures the code will not be “over-nested”.

10.2. Main Package

The Controller class is located in this package, this entity commands and synchronizes all actions on the software, therefore it is the main thread of the system. Excepting the data pipeline (DataBuffer-RegisterLogger classes), all information flow inside the packages is done through the Controller. So the controller has the task to:

- Execute user commands and provide data to the interface;
- Manage all thread instances are working during acquisition;

Figure 81 – Tama Client class diagram, final version.



- Instantiate and start all procedures on the sub-instances located in other packages;
- Handle failure in the system insuring minimum loss and maximum robustness;
- Keep track of all events which happened on the system;

The ConditionControl class can also be found on the package. This class is responsible for checking the automatic conditions programmed by the user on the “Program config” tab. Each programmed action is an implementation of the “ConditionAction” interface, which basically needs to instantiate a firing condition and a subsequent action. All possible automatic actions are then checked regularly by the ConditionControl thread, and actions are taken if the condition is achieved.

Both user via interface and the actions performed by ConditionControl thread act by calling methods on the Controller. Therefore, the system can be perceived as being commanded by two concurrent entities, performing manual and automatic actions simultaneously. So, in order to guarantee consistency on the program behaviour, the controller commands are protected with special lockers.

10.3. Tama Handler Package

This package is responsible for all actions related to the Trialink network. It is also the only package which uses the Tama libraries in order to communicate with the ring.

When the program is initialized, only TrialinkConnection is instantiated by the Controller. As the procedures go on, further classes are instantiated in the program. This eases the maintenance of runtime instances by deleting unnecessary objects when back-stepping.

Following the program execution sequence: primarily, when the software is started, the Controller instantiates the TrialinkConnection class. When commanded to connect, this class accessess the network and requests the device tree using the Tama API for the task.

After this step, the program navigates through the device tree in recursive mode identifying all nodes on the accessed network. At the same time, the program creates an internal representation of the device tree using the IPTNode class, which contains basic tree attributes and registers describing ID, name and data type. On this way, the system has a tree representation containing only the relevant information about the register tree of the Trialink network. The register IDs are unique strings organized like directory paths

This node tree is forwarded to the Controller, and then to the interface. When the acquisition program is selected by the user, the controller sends a list containing each acquisition sampling rate and the register list (IDs) of all registers selected by the user. This information is then stored in the AcquisitionConfig class.

When the Acquisition is about to start, the TrialinkConnection instantiates multiple objects of the Acquisition class. Each Acquisition instance synchronizes one register group according to its programmed sampling rate.

Each of the Acquisition instances will then instantiate the Subscribers. The Subscriber on Tama Client has the task of communicating with the Subscription mechanism of the ring network. Devices on the ring send data to each other using packages containing up to 5 register values. Configuration regarding this communication schema is set between devices using the Subscription mechanism. In other words, a subscriber device requests information from the publisher by sending a Subscription request. Therefore, The Subscriber task is to identify the right subscription type and package size. After that, it sends the request to the devices on the network and receives the data packages.

Although, the Subscriber instances receive the data packages during the acquisition, they do not check their content. Each Subscription, after receiving a list of packages, forwards those to the RegisterLoggers (instances of RegisterAbstractLogger), when then the package content will be read and checked.

Each RegisterLogger is responsible for processing information from One Register. Once they are threaded, all data registers are processed in parallel. The loggers are also responsible for checking two conditions:

- Each package has a corresponding Timestamp, and all packages are transported in the ring in a FIFO order. Once the loggers know the sampling rate, by checking sequentially the timestamp of each package, they can diagnose if data was lost on the ring. Raising the “Overload” flag to the controller immediately;
- The loggers also check data consistency, if a register bounded to some interval (value $-1 < v < 1$) or it has a certain behavior (a timestamp is always an increasing value) and the received value has some anomaly, a “Failure” flag is raised.

It worth noticing that the Logger instances are generic classes. They are the first element of – what is called in Tama Client – the Data Pipelines. So, each Logger is instantiated with the type of the variable it will process. This type is identified when first accessing the register tree of the network on the identification step.

After reading and converting the binary packages to typed values, the Loggers then forward the data to the DataBuffers, when they will be enqueued before flushed to disk by instances of the File Handler package.

10.4. File Handler Package

This package has all the tasks regarding archiving and loading files. It is responsible for checking paths, save and load configurations, open zip deflate files, write data during acquisition, buffering, buffer overflow control, streaming data for online visualization using ZMQ and invoking the converter to finally convert the zip to HDF files. This package also writes the record of all the events which happened on the system. In resume, the package handles all I/O operations with the local disk.

The ZMQStreamer is a class which uses Zero MQ protocol to stream data to another software for online visualization purpose. It is also a Thread which tries sending data using string messages. In order to access the data on the generic pipeline, the ZipAbstractStreamer has an abstract method implemented by its children which goes to the DataBuffer, dequeues the streaming values, convert them to a string array and return them.

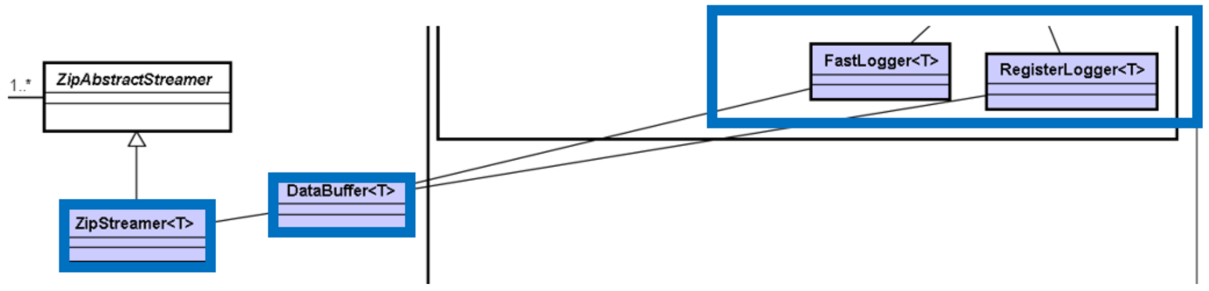
The ZMQStreamer passes one by one all the ZipStreamers, gets all the data, builds the string message and send it to the programmed port in a constant loop.

10.5. Data Pipeline

The Data Pipeline here corresponds to the Generic region of the software where the acquisition data is processed (see Figure 82). There is one pipeline for each variable acquired. Each pipe has one instance of a Logger (depending on the subscriptin type, it can be any of both), one DataBuffer and one ZipStreamer.

In order to avoid overheads related to converting data to a common type like double, this pipeline was implemented in a Generic way. So if the variable of the pipeline is a double, it is going to be processed as a double, if it is a Boolean, Boolean, if Float40, it is reduced to float and then forwarded, and so on.

Figure 82 – Data Pipeline instances of the Tama Client class diagram.



So, another task of the Loggers is to convert the Tama data type to struct (raw C# type). As well as the ZipStreamer converts this structs at the end to bytes and write them down on disk.

Since the Loggers and ZipStreamers are parallel instances, we need a way to pool data between them, and at the same time, allow both instances to add and get data concurrently. So, this is the tasks of the DataBuffer on the system: it has mainly one Queue, concurrent methods (using Mutex) to add and remove values from the Queue, counters used to control and synchronize data between different pipes, and memory overflow control – constraining queue size. This is the only place on the entire system where data is pooled.

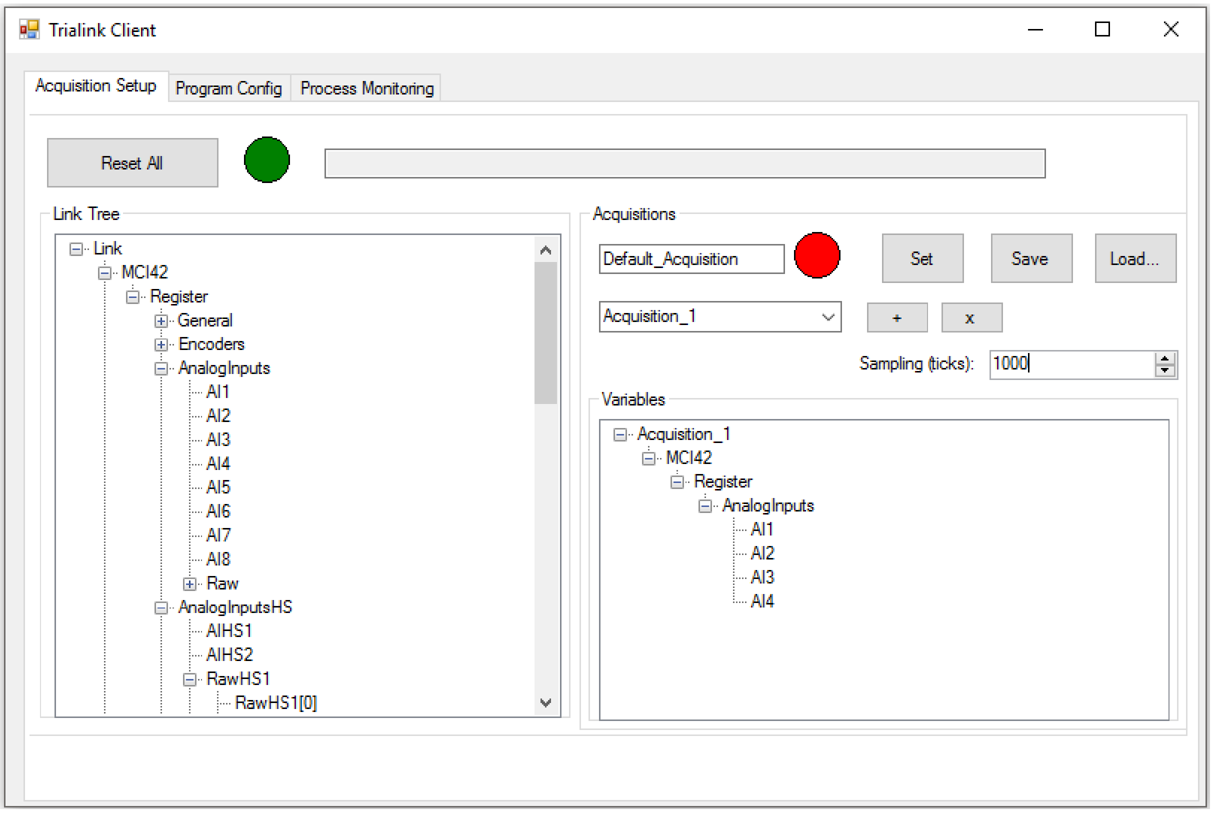
10.6. User Interface

As previously mentioned, the user operates the system using the program interface. The acquisition is configured and monitored by the usage of 3 tabs in the main window sequentially:

1. Acquisition Setup: corresponds to selecting all desired registers and sampling rates for the acquisition
2. Program Config: programs general system behavior, output folders, max file sizes, etc.
3. Process Monitoring: where user can see the status of the acquisition, diagnose errors, see current values acquired and command the acquisition.

Figure 83, Figure 84 and Figure 85 presents the screenshots of each of those tabs.

Figure 83 – Acquisition Setup tab of Tama Client Software.

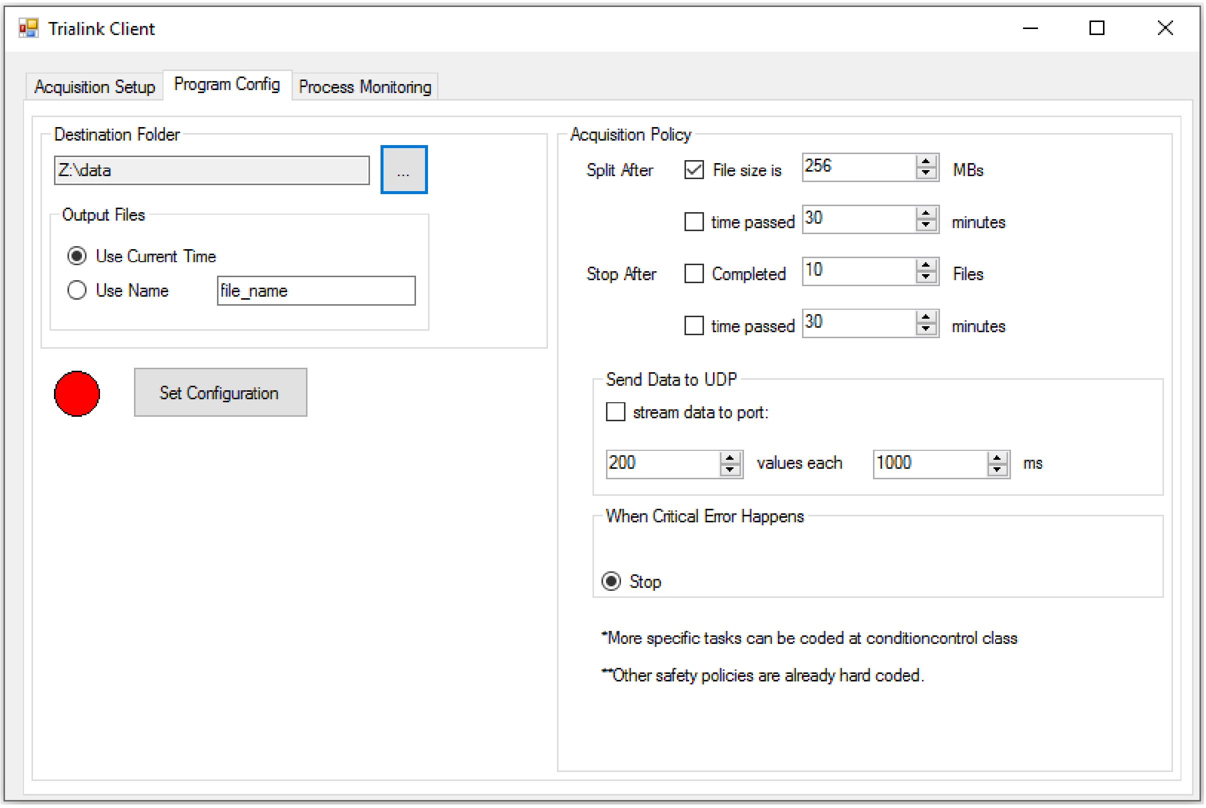


Default_Acquisition

Acquisition_1

Sampling (ticks): 1000

Figure 84 – Program Configuration tab of Tama Client Software.



Destination Folder: Z:\data

Output Files
- [✓ Use Current Time]
- Use Name: file_name

Acquisition Policy

Split After
- [✓] File size is 256 MBs
- [ ] time passed 30 minutes

Stop After
- [ ] Completed 10 Files
- [ ] time passed 30 minutes

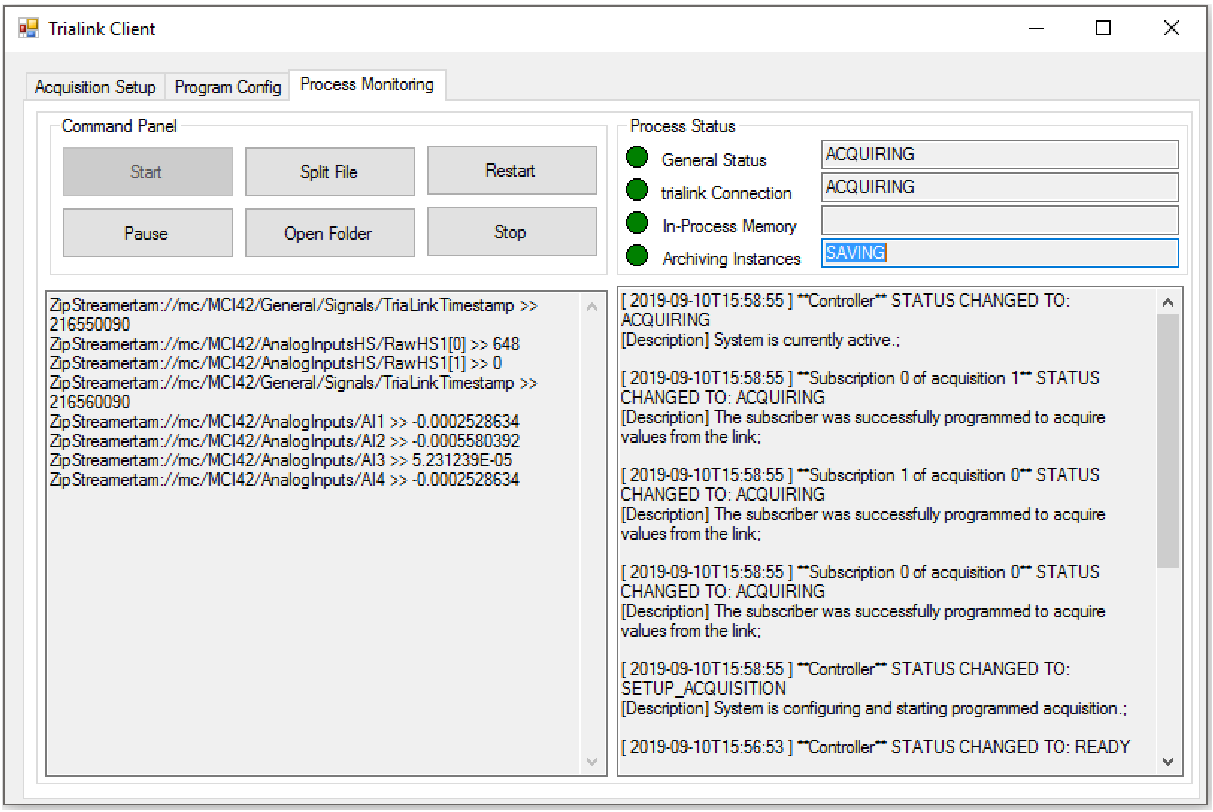
Send Data to UDP
- [ ] stream data to port:
- 200 values each 1000 ms

When Critical Error Happens
- [✓ Stop]

*More specific tasks can be coded at conditioncontrol class

**Other safety policies are already hard coded.

Figure 85 – Tama Client Process Monitoring tab with ongoing acquisition.



Process Status

● General Status	ACQUIRING
● trialink Connection	ACQUIRING
● In-Process Memory	
● Archiving Instances	SAVING

```
ZipStreamertam://mc/MCI42/General/Signals/TriaLinkTimestamp >>
216550090
ZipStreamertam://mc/MCI42/AnalogInputsHS/RawHS1[0] >> 648
ZipStreamertam://mc/MCI42/AnalogInputsHS/RawHS1[1] >> 0
ZipStreamertam://mc/MCI42/General/Signals/TriaLinkTimestamp >>
216560090
ZipStreamertam://mc/MCI42/AnalogInputs/AI1 >> -0.0002528634
ZipStreamertam://mc/MCI42/AnalogInputs/AI2 >> -0.0005580392
ZipStreamertam://mc/MCI42/AnalogInputs/AI3 >> 5.231239E-05
ZipStreamertam://mc/MCI42/AnalogInputs/AI4 >> -0.0002528634
```

```
[ 2019-09-10T15:58:55 ] **Controller** STATUS CHANGED TO:
ACQUIRING
[Description] System is currently active.;

[ 2019-09-10T15:58:55 ] **Subscription 0 of acquisition 1** STATUS
CHANGED TO: ACQUIRING
[Description] The subscriber was successfully programmed to acquire
values from the link;

[ 2019-09-10T15:58:55 ] **Subscription 1 of acquisition 0** STATUS
CHANGED TO: ACQUIRING
[Description] The subscriber was successfully programmed to acquire
values from the link;

[ 2019-09-10T15:58:55 ] **Subscription 0 of acquisition 0** STATUS
CHANGED TO: ACQUIRING
[Description] The subscriber was successfully programmed to acquire
values from the link;

[ 2019-09-10T15:58:55 ] **Controller** STATUS CHANGED TO:
SETUP_ACQUISITION
[Description] System is configuring and starting programmed acquisition.;

[ 2019-09-10T15:56:53 ] **Controller** STATUS CHANGED TO: READY
```

11. APPENDIX C – PREDICTION MODELS: TABLE OF RESULTS

ID	Network	layers	Dataset	input shape
1	ANN Regression	(10,50)-(10,8)-flat*-8-4-2-1	vib_sum	10, 50
2	ANN Regression	(10,150)- (10,8)-flat-8-4-2-1	vib_x3	10, 150
3	ANN Regression	(16,64)- (16,8)-flat-8-4-2-1	ae_workpiece	16, 64
4	ANN Regression	(16,64)- (16,8)-flat-8-4-2-1	ae_spindle	16, 64
5	ANN Regression	(10,50)- (10,8)-flat-8-4-2-1	mic	10, 50
6	ANN Regression	(10,50)- (10,16)-flat-8-8-4-1	vib_sum	10, 50
7	ANN Regression	(10,150)- (10,16)--flat-8-8-4-1	vib_x3	10, 150
8	ANN Regression	(16,64)- (10,16)--flat-8-8-4-1	ae_workpiece	16, 64
9	ANN Regression	(16,64)- (10,16)--flat-8-8-4-1	ae_spindle	16, 64
10	ANN Regression	(10,50)- (10,16)--flat-8-8-4-1	mic	10, 50
11	Auto-Encoder	50-10-50	vib_sum	1, 50
12	Auto-Encoder	150-10-150	vib_x3	1, 150
13	Auto-Encoder	50-10-50	mic	1, 50
14	Auto-Encoder	64-32-16-4-16-32-64	ae_workpiece	1, 64
15	Auto-Encoder	64-32-16-5-16-32-64	ae_spindle	1, 64
16	ANN Regression	(10,50)-(10,16)-flat-drop0.5**-16-8-4-1	auto_vib_sum	10,10
17	ANN Regression	(10,50)-(10,16)-flat-drop0.5-16-8-4-1	auto_vib_x3	10,10
18	ANN Regression	(10,50)-(10,16)-flat-drop0.5-16-8-4-1	auto_mic	10,10
19	ANN Regression	(16,64)-(16,16)-flat-drop0.5-16-8-4-1	auto_workpiece	16, 4
20	ANN Regression	(16,64)-(16,16)-flat-drop0.5-16-8-4-1	auto_spindle	16, 5
21	LSTM Regression	(15,50)-32-drop0.5-32-drop0.5-16-8-1	vib_sum	15, 50
22	LSTM Regression	(15,50)-32-drop0.5-32-drop0.5-16-8-1	vib_x3	15, 50
23	LSTM Regression	(15,50)-32-drop0.5-32-drop0.5-16-8-1	mic	15, 50
24	LSTM Regression	(16,64)-32-drop0.5-32-drop0.5-16-8-1	ae_workpiece	16, 64
25	LSTM Regression	(16,64)-32-drop0.5-32-drop0.5-16-8-1	ae_spindle	16, 64
26	LSTM Classifier	(15,50)-64-drop0.5-64-drop0.5-32-16-6	vib_sum	15, 50
27	LSTM Classifier	(15,50)-64-drop0.5-64-drop0.5-32-16-6	vib_x3	15, 50
28	LSTM Classifier	(15,50)-64-drop0.5-64-drop0.5-32-16-6	mic	15, 50
29	LSTM Classifier	(16,64)-64-drop0.5-64-drop0.5-32-16-6	ae_workpiece	16, 64
30	LSTM Classifier	(16,64)-64-drop0.5-64-drop0.5-32-16-6	ae_spindle	16, 64
31	LSTM Regression	(15,50)-32-drop0.5-32-drop0.5-16-8-1	vib_sum	15, 50
32	LSTM Regression	(15,50)-32-drop0.5-32-drop0.5-16-8-1	vib_sum	15, 50

* : flat represents dimensionality reduction from a 2-D layer to a 1-D layer.

**: dropout portion of the preceding layer

***: left-shifting augmentation

****: auto-encoder architecture

*****: Keras Functions - mse: mean squared error; acc: accuracy estimation or deviation from true value; cce: categorical cross-entropy; cac: categorical accuracy, or portion of prediction max values which matches the true value.

ID	extras	epochs	batch-size	loss-function	accuracy-function
1	aug***	50	1024	mse*****	mse
2	aug	50	1024	mse	mse
3	aug	50	1024	mse	mse
4	aug	50	1024	mse	mse
5	aug	50	1024	mse	mse
6	aug	50	1024	mse	mse
7	aug	50	1024	mse	mse
8	aug	50	1024	mse	mse
9	aug	50	1024	mse	mse
10	aug	50	1024	mse	mse
11	simple****	10	1024	mse	acc
12	simple	10	1024	mse	acc
13	simple	10	1024	mse	acc
14	deep	10	1024	mse	acc
15	deep	10	1024	mse	acc
16	aug	50	128	mse	mse
17	aug	50	128	mse	mse
18	aug	50	128	mse	mse
19	aug	50	128	mse	mse
20	aug	50	128	mse	mse
21	aug	50	512	mse	mse
22	aug	50	256	mse	mse
23	aug	50	256	mse	mse
24	aug	50	512	mse	mse
25	aug	50	512	mse	mse
26	aug	50	256	cce	cac
27	aug	50	256	cce	cac
28	aug	50	256	cce	cac
29	aug	50	256	cce	cac
30	aug	50	256	cce	cac
31	aug+ weights	50	128	mse	mse
32	aug	50	128	mse	mse

ID	acc	loss	val-acc	val-loss	best val-acc
1	267.8612	267.8612	538.7658	538.7658	505.6175
2	225.3065	225.3065	475.1323	475.1323	471.5322
3	575.0486	575.0486	466.0967	466.0967	463.9799
4	531.1392	531.1392	316.1533	316.1533	305.8719
5	624.0636	624.0636	853.4193	853.4193	805.3639
6	243.8391	243.8391	483.3451	483.3451	468.42054
7	212.6319	212.6319	488.3767	488.3767	488.3767
8	430.6961	430.6961	303.6426	303.6426	293.165
9	526.6828	526.6828	313.7809	313.7809	313.7809
10	597.2996	597.2996	886.4907	886.4907	822.6655
11	0.9401	0.0673	0.9215	0.1235	0.9215
12	0.9425	0.0172	0.9257	0.028	0.92737
13	0.9317	0.003	0.9323	0.0031	0.9323
14	0.9024	483.5302	0.9183	403.732	0.92032
15	0.9815	3982.7846	0.9783	3975.0151	0.97831
16	313.439	313.439	583.2678	583.2678	509.45907
17	324.1524	324.1524	640.1625	640.1625	560
18	671.3022	671.3022	853.2402	853.2402	818.04535
19	599.0172	599.0172	470.7251	470.7251	461.7848
20	766.4859	766.4859	742.4829	742.4829	742.36844
21	210.4981	210.4981	421.2085	421.2085	284.29948
22	136.7346	136.7346	524.6397	524.6397	176.15272
23	614.9895	614.9895	846.2655	846.2655	771.1928
24	485.4925	485.4925	721.6194	721.6194	420.97011
25	589.6677	589.6677	978.1514	978.1514	441.0024
26	0.8153	0.4458	0.7176	0.6596	0.78038
27	0.8518	0.3654	0.791	0.4726	0.81738
28	0.445	1.3327	0.1152	1.8604	0.13881
29	0.6709	0.8197	0.5924	1.0481	0.65988
30	0.4646	1.26	0.4624	1.2604	0.56453
31	277.7136	145.0843	364.6487	364.6487	260.5499
32	189.8475	189.8475	544.6927	544.6927	192.73205