Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Diogo Luiz Demarchi

Convolutional and Recurrent Neural Networks in Time-series applied to Injection Molding Processes

Blumenau
2019

Diogo Luiz Demarchi

Convolutional and Recurrent Neural Networks in Time-series applied to Injection Molding Processes

Final paper submitted in partial fulfillment of the requirements for the degree of BEng. in Automation and Control Engineering of the Universidade Federal de Santa Catarina.
Advisor: Prof. Dr. Mauri Ferrandin

Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Blumenau
2019

Diogo Luiz Demarchi

Convolutional and Recurrent Neural Networks in Time-series applied to Injection Molding Processes

Final paper submitted in partial fulfillment of the requirements for the degree of BEng. in Automation and Control Engineering of the Universidade Federal de Santa Catarina.

Examination Committee

Prof. Dr. Mauri Ferrandin
Universidade Federal de Santa Catarina
Advisor

Prof. Dr. Maiquel de Brito
Universidade Federal de Santa Catarina

Prof. Dr. Janaina Gonçalves Guimarães
Universidade Federal de Santa Catarina

Blumenau, January 30, 2019

Dedico este trabalho aos meus pais, que lutaram ao meu lado para que esse sonho fosse realizado.

Acknowledgements

I would like to thank above all my parents Elfi Klug Demarchi and Tarcisio Demarchi for always supporting my dreams. Without their help, it would be impossible for me to be doing this work and mostly be doing this abroad. I also want to thank my girlfriend for always pushing me to become a better person and for wanting us to do our internships in Germany.

I am also grateful for the wonderful people I had the opportunity to meet during my internship at Fraunhofer IPA, who besides from colleagues have become great advisors. Without their support, this would not be possible. I would like to specially thank Mr. Felix Müller for giving me the opportunity to work with him.

My research would not have been possible without the knowledge that my professors at Federal University of Santa Catarina gave me, specially Dr. Mauri Ferrandin for being my advisor during my thesis.

Finally, I would like to thank my friends for all the help and support either with input about the thesis itself, for going to travels together or just for being there for me.

"Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less."

(Marie Curie)

Resumo

A Fraunhofer-Gesellschaft é a maior organização de pesquisa aplicada da Europa, fundada em 1949. Seus campos de estudo variam de automação até novos materiais, energia, meio-ambiente, saúde, física e muitos outros campos. O instituto específico deste projeto é o Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, ou Instituto de Engenharia da Manufatura e Automação, fundado em 1959. O Fraunhofer IPA tem cerca de 1200 empregados e é um dos maiores institutos dentro da Fraunhofer-Gesellschaft. Dentro do instituto, existem vários laboraórios onde diversos experimentos são realizados. Em um destes laboratórios, há uma máquina injetora que não possui métodos para o fornecimento de informação sobre o estado atual da máquina e suas ferramentas. Em vista da criação de uma solução para este problema, o projeto consiste em analisar o processo desta máquina injetora a partir de câmeras e sensores de vibração. Para isto, técnicas de machine learning são utilizadas para obter a classificação do estado de funcionamento atual da máquina a partir destes tipos de dados. Comparações e análises entre as duas abordagens são realizadas ao decorrer do projeto, bem como explicação de resultados positivos e negativos. As câmeras e seus modelos de redes neurais apresentam, em sua maioria, resultados de precisão excelentes na classificação desta máquina enquanto os resultados dos sensores de vibração são apenas satisfatórios, sendo este esperado previamente pela natureza dos dados. Apesar disto, a informação de vibração ainda é útil e pode ser usada para classificar o estado da máquina.

Palavras-Chave: 1. Máquina Injetora. 2. Redes Neurais Recorrentes. 3. Redes Neurais Convolucionais. 4. Câmera. 5. Acelerômetro.

Abstract

The Fraunhofer-Gesellschaft is the biggest application-oriented research organization in Europe, founded in 1949. Its fields of study vary from automation to new materials, energy, environment, health, physics and many other fields. The specific institute of this project is the Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, or Institute for Manufacturing Engineering and Automation, founded in 1959. The Fraunhofer IPA has around 1200 employees and it's one of the largest institutes inside Fraunhofer-Gesellschaft. Within the institute, there are several laboratories where many experiments are performed. In one of these laboratories, there is one injection molding machine that do not have methods to provide information about the current state of the machine and its tools. In order to create a solution to this problem, the project consists of analyzing the process of this injection molding machine with cameras and vibration sensors. For this, machine learning techniques are used to obtain the classification of the current operating state of the machine based on these data sources. Comparisons and analyzes between the two approaches are carried out throughout the project, as well as explanation of both positive and negative results. The cameras and their neural network models present, for the most part, excellent precision results in the classification of this machine while the results of the vibration sensors are only satisfactory, which is previously expected by the nature of the data. Despite this, the vibration information is still useful and can be used to predict the state of the machine.

Keywords: 1. Injection Molding Machine. 2. Recurrent Neural Networks. 3. Convolutional Neural Networks. 4. Camera. 5. Accelerometer.

Zusammenfassung

Die Fraunhofer-Gesellschaft ist die größte anwendungsorientierte Forschungsorganisation in Europa, die 1949 gegründet wurde. Die Studienbereiche variieren von der Automatisierung bis hin zu neuen Materialien, Energie, Umwelt, Gesundheit, Physik und vielen anderen Bereichen. Das spezifische Institut für dieses Projekt ist das 1959 gegründete Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. Das Fraunhofer IPA beschäftigt rund 1200 Mitarbeiter und ist eines der größten Institute der Fraunhofer-Gesellschaft. Innerhalb des Instituts gibt es mehrere Laboratorien, in denen viele Experimente durchgeführt werden. In einem dieser Laboratorien gibt es eine Spritzgießmaschine, für die kein Mechanismus existiert, um Informationen über den aktuellen Zustand der Maschine und ihrer Werkzeuge bereitzustellen. Um dieses Problem zu lösen, besteht das Projekt darin, den Prozess diese Spritzgießmaschine mit Kameras und Schwingungssensoren zu analysieren. Dazu werden methoden des maschinellen Lernens eingesetzt, um anhand dieser Datenquellen die Klassifizierung des aktuellen Betriebszustands der Maschine zu erhalten. Während des gesamten Projekts werden Vergleiche und Analysen zwischen den beiden Ansätzen durchgeführt und sowohl positive als auch negative Ergebnisse erläutert. Die Kameras und ihre neuronalen Netzwerkmodelle weisen größtenteils hervorragende Ergebnisse bei der Klassifizierung dieser Maschine auf, während die Ergebnisse der Schwingungssensoren nur zufriedenstellend sind, was bisher von der Art der Daten erwartet wurde. Trotzdem ist die Vibrationsinformation immer noch nützlich und kann verwendet werden, um den Zustand der Maschine vorherzusagen.

Schlüsselwörter: 1. Spritzgießmaschine. 2. Rekurrentes Neuronales Netz. 3. Faltendes Neuronales Netzwerk. 4. Kamera. 5. Beschleunigungssensor.

List of figures

List of tables

Acronyms

UFSC	*Universidade Federal de Santa Catarina*
UART	*Universal Asynchronous Receiver Transmitter*
CNN	*Convolutional Neural Network*
RNN	*Recurrent Neural Network*
IM	*Injection Molding*
GUI	*Graphical User Interface*
USB	*Universal Serial Bus*
IoT	*Internet Of Things*
RGB	*Red Green and Blue*
GPU	*Graphics Processing Unit*
CPU	*Central Processing Unit*
kNN	*k-Nearest Neighbors*
LSTM	*Long Short-Term Memory*
NaN	*Not a Number*
FPS	*Frames Per Second*
MNIST	*Modified National Institute of Standards and Technology*
CIFAR	*Canadian Institute For Advanced Research*
RAM	*Random Access Memory*
MB	*Megabyte*
GB	*Gigabyte*
PoE	*Power over Ethernet*
IP	*Ingress Protection*
DC	*Direct Current*
VGA	*Video Graphics Array*
SXGA	*Super Extended Graphics Adapter*
IPA	*Institut für Produktionstechnik und Automatisierung*

Table of contents

1 Introduction

A problem in the industry is the inefficiency of some tools, machines or even whole plants. Problems from this mean can come from many ways, like bad positioning of tools or bad synchronization of machines, for example. This kind of problem can cause, aside from the inefficiency, harm to the hardware and reduce its useful life due to friction, heating or vibration.

In order to try to minimize or solve this kind of problem, machine learning can be a powerful tool to detect anomalies or even classify the state of any system purely by taking some relevant information of the process while in operation and feeding to an algorithm that can study the behavior of the system based on these parameters and their respective output. To do this accurately, this algorithm must first be very optimized and the data should also have good quality. This allows to recognize the actual state of the machine and with this knowledge, make adjustments on parameters to increase its efficiency.

In machine learning, there are two main concepts of how to train the model in order to minimize the cost function of a given training dataset. The first and simpler way is to do the offline learning, that is taking a big batch of the whole dataset and train the model once. Despite being a very fast process, it needs a powerful hardware to store all of the data in memory. In some cases, the data is so big that it wouldn't be possible to do this kind of training. For the sake of this problem comes the online learning in action, which trains a small model and constantly keeps adding new weights as it as data flows in. This second method is commonly applied on the spam filtering of emails and applications such as social medias. To add new data to offline training, it would be necessary to train the whole model all over again or at least retrain a model already trained.

Besides from pure inefficiency, there are also major problems, such as failure in some motor, pump or other kind of hardware that is essential to certain productive process. If this hardware doesn't have some kind of detector, the problem could easily be noticed long after the failure happened. To give a generic solution to almost any kind of process in the industry, cameras and vibration sensors as data sources sound like a good solution. They just need a good positioning and then can acquire data for almost every process. The exceptions are when the process is not visible or inaccessible.

1.1 Injection Molding Process

It is fundamental to understand the injection molding process in order to follow with this work. The basic principle of operation of this kind of machine is to take some raw material and change it into a desired shape. To do so, an injection molding machine have

some essential parts, and the most important are explained following. The schematic of a simple injection molding machine is displayed in Figure 1.

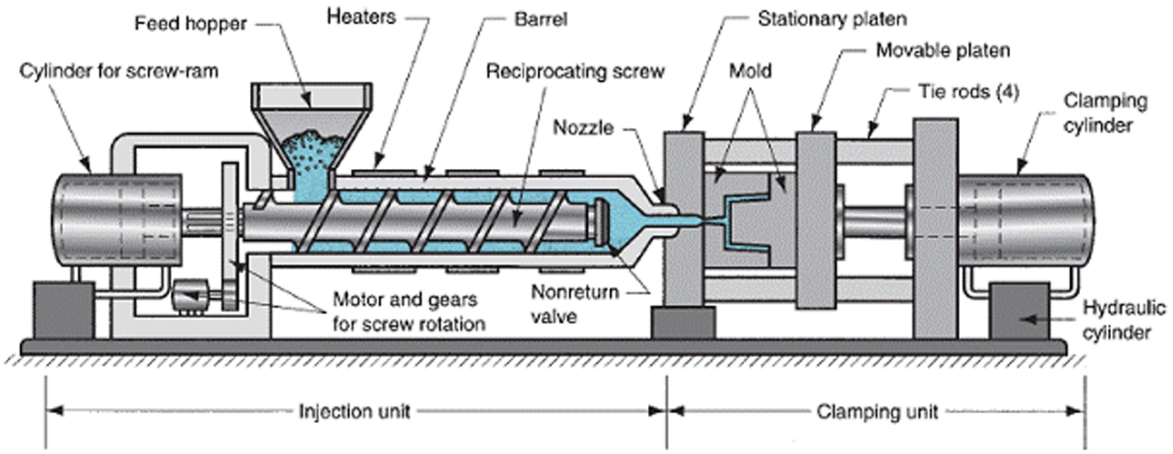


Figure 1 – Drawing of a simple Injection Molding Machine

One important part is the feed hopper, where the raw material is disposed and then transported to a barrel with a reciprocating screw, which transports the material while being heated by the heaters. Then, the melted material passes through a nozzle to get to the mold, in order to get the desired shape. When the melted material already has the form of the mold, it needs to cool to solidify, and then finally the mold can be released to finish the new product. This product is then removed from the mold usually by gravity, falling into a conveyor system or by a pick and place robot if the piece is too big or difficult to remove.

1.1.1 Problems

Some problems that can occur with injection molding machines are simple and easy to solve. One common problem is with burned parts in the output, which can be solved simply by lowering the temperature of the heaters or speeding the process. In fact, most of the solutions for the problems with the output of injection molding machines stay around changing the cycle time or the temperature. It is not important to focus on these kind of problems for this work, but on problems surrounding the machine itself.

One relevant problem is overpressure on the mold. This makes the machine waste a lot of energy and produce a lot of noise and vibration. This problem is specially relevant to this study for being easily reproducible and measured with vibration loggers.

Some injection molding machines do not have the information about their actual state of operation. Sometimes for reasons such as security, optimization or even synchronization with other simultaneous processes, it could be useful to have the knowledge about the mold state and its exact time of opening and closure.

1.2 Objective

The objective of this work is to take an injection molding machine process, acquire data and analyse this data through machine learning algorithms. To do so, two approaches are used in order to collect data.

It is proposed the use of vibration sensors to acquire information about the vibration of the machine. The main objective with this approach is try to map the behavior of the machine while in a certain state of operation with the vibration data acquired, in order to be able to classify the actual machine state based purely on its vibration.

The other approach is to use cameras to identify what task the machine is doing in the moment. The objective is test the positioning of the camera together with different algorithms and compare the results both from other camera settings and vibration predictions.

As the final output desired, both data sources should provide reliable information to one machine learning model each, in order for them to get the same result for the machine state while it is operating.

2 Contextualization

This chapter tries to explain the terms and concepts used in this document and how they affect the project. It is presented from the initial approaches to machine learning all the way to methods that are being used nowadays. It also gives a high-level understanding of complex concepts related to the topic. The objective is to give a basic overview of the theory behind the practice, not going deep into details.

2.1 Machine Learning

The basic concept of machine learning is trying somehow to teach a device, given an unknown data, how to identify that data as belonging to some class or having a given value. The way to do this is by giving a relatively large dataset with already classified information, and then applying some algorithm to identify in which class the new data belongs. It's not possible to write a program that helps a camera to distinguish between a lake or a sea, but it's possible to feed some data in order to make the computer see the similarities and then infer if the given image is indeed a lake or a sea.

Machine learning algorithms let us, human beings, be able to solve problems that we could not solve with simple static algorithms. The main point of interest of machine learning, speaking from a more philosophical and scientific point of view, is that it is not just programming a simple algorithm, but it is in fact, trying to understand the basic principles of intelligence.

2.1.1 Supervised Learning

In machine learning, the term supervised learning is the method or approach to use pairs of inputs and labeled outputs in order to train the algorithm for predicting a new unknown input, based on previously mapped pairs.

Generally, supervised learning problems consist basically in Classification (2.1.3.1) and Regression (2.1.3.3) situations. In both cases, it is necessary to have a set of training values consisting on an input with an output value to it.

2.1.2 Unsupervised Learning

Unsupervised learning allows to learn with little or no information about the expected output. This approach is less used in practice, but it is still an important and powerful tool in some cases. It could be used to detect different group of individuals with similar

patterns, detect tendencies or even just to lower the number of dimensions of a specific problem, focusing only on a more general overview of the attributes in a certain dataset.

2.1.3 Types of problems

There are many tasks that we might want to automate. If we want a machine to talk, then talking is the task. There are two approaches in order to make a machine able to talk: the first is simply program the machine to say some defined phrases, and the other one is making the machine learn how to talk and interact with a human being.

These tasks are what we want the machine to be able to learn, or how they should output a result based on previously known values, given new data. If we give a set of features of a certain process to a machine, we have to specify what we want as output. There are several possible kinds of tasks or approaches to this matter, and some of them are described below.

2.1.3.1 Classification

As the name states, the main purpose of this algorithm is to classify something as belonging to some category. To solve this problem, the algorithm makes a model or function $f : \mathbb{R}^n \rightarrow \{1, ..., k\}$ that represents the k possible classes of the problem. Making $y = f(x)$, when an input vector x is passed through the model f, it returns a category y, a numeric value binded to one class.

Another way to solve classification problems is instead of giving a pure class as the output, make it give the probability of being each class. This is useful to see how well-tuned the model is and how certain it is of the given prediction.

A simple example of a classification problem is the classic MNIST dataset, which consists of handwritten numbers between 0 and 9, and the task of the model is, given just an image of the number, identify which number it is. Another common example is sentiment analysis, which small phrases are taken from user review of some product or service, and the objective of the model is tell whether or not the reaction of the consumer was positive.

2.1.3.2 Classification with missing inputs

It is a very similar problem to the simple classification, with the exception that it may miss one or more value of the input vector x. In order to solve this problem, the algorithm must take into account every possibility of input numbers, being necessary to learn a set of functions, one for each input vector length.

This is a rather common problem in medical diagnosis, where the patient not always have all of the symptoms of a certain disease, so in order to save money with expensive and unnecessary exams for certain people, it is better just to leave the value missing.

2.1.3.3 Regression

This kind of problem does not generate a classification output, but generates a numeric value as output. So, in order to solve this particular task, the function takes the form $f : \mathbb{R} \rightarrow \mathbb{R}$. Despite the output being completely different from the classification algorithm, the input is the same, so it is in fact similar to the classification problems.

A simple example of regression problem is, given certain characteristics of an 3D printer, like temperature, position of the piece in the chamber, and type of filament, determine the exact final length or weight of a piece.

2.1.3.4 Transcription

As the name suggests, in this kind of problem the output is a transcription of an unstructured kind of data. For example, reading houses' numbers just with a picture in front of the house. Similarly, another common kind of transcription is made with audio as input for speech recognition, where the input is the speaking voice of someone and the output is a transcription of what the person said.

2.1.3.5 Machine Translation

Machine translation task is the translation of some kind of language to another. The input is a sequence of symbols in some known language and the output is another sequence of symbols in another language. This is used in online translators, such as Google Translator, in order to translate Portuguese to German, for example.

2.1.3.6 Anomaly Detection

There are times when it is not necessary to know what kind of problem or failure occurred in a given process, but it is crucial to know if an anomaly has indeed happened. The output of this kind of algorithm is simply a boolean flag, but there are cases where this kind of detection is very important. A clear example is a manufacturing line operating uninterruptedly a well-known process. If there is something outside the pattern of operation, a flag would be triggered automatically, removing the necessity of having a human to monitor the process all day. Another common example is to detect credit card fraud. If the credit card companies make a model of each person's style of shopping, it is not hard to identify if a very strange purchase is made. This way, it is not necessary to map all of the possible failures that can occur, but simply teach the model very well how the process should be operating and anything outside that pattern, is in fact an anomaly.

2.1.4 Types of Algorithms

These algorithms for machine learning can be the most simple or the most complex ones, depending on the variety of the dataset and how accurate they need to be. Although there are several machine learning algorithms, for the sake of this text it will be presented just a few relevant algorithms are presented in this text.

2.1.4.1 Linear and Logistic Regression

One of the simplest and well-known algorithms is the Linear Regression, which draws a line in a 2D graph and values above that line belong to class X and values below belong to class Y. The goal with this algorithm is drawing a line in which all values from one class are on one side and values from the other class are on the other side, being the inclination and the offset of the line the only two coefficients, also called weights, to adjust. Alternatively, it's possible to use a non-linear function in order to get better accuracy, with the increase of weights to be adjusted. The difference can be seen in the Figure 2.

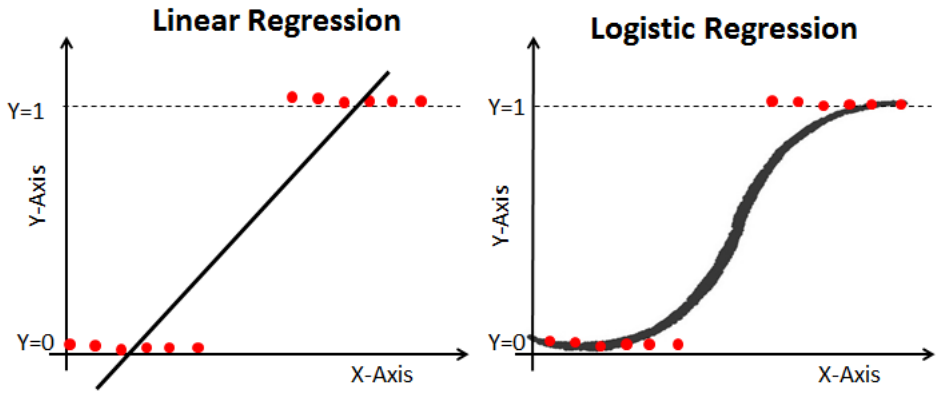


Figure 2 – Linear and Logistic Regression [1]

To more complex problems there is also polynomial regression, where the curve can have any polynomial shape. [7]

2.1.4.2 Decision Tree

Another important algorithm of machine learning is the decision tree. Its basic operation principle is "asking questions" to itself about the data, and then move to another question, until it reaches a result. It's a completely iterative process, so the best is to have the smallest possible decision tree without losing accuracy in order to save computational power [8]. Figure 3 shows an example of a complete decision tree, where it predicts the grade of a student before an exam.

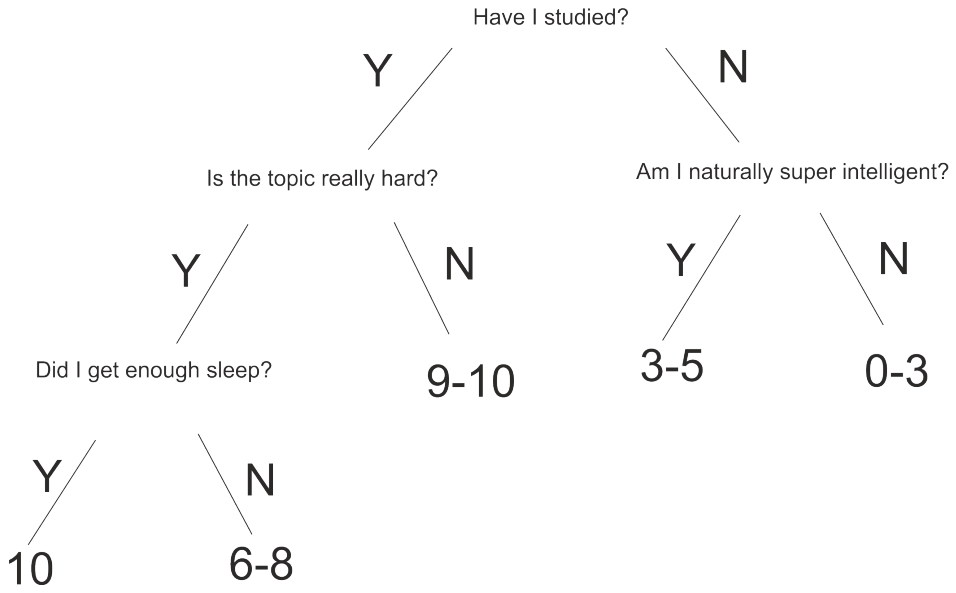


Figure 3 – Example of a decision tree

2.1.4.3 k-Nearest Neighbors

The k-Nearest neighbors (also known as kNN) algorithm can be used both for regression (2.1.3.3) and classification (2.1.3.1) problems, although it is more used for classification. The basic concept is to, given a known labeled dataset, for each new case classify and store the data based on the values that the k nearest neighbors have.

For example, if k is 5, the algorithm will look for the 5 closest values of the new entry. If the majority of the neighbors are from class X, then this new entry will also be classified as being from the class X. If the value of k is equal 1, then this new value will be classified as the same as the closest value on the graph.

Note that the distance can be calculated in many ways, also being possible to go outside of the usual two dimensional space by calculating the spatial distance between values.

However, besides being an easy to understand algorithm, it takes a lot of computational power because of the iterations necessary to find the closest neighbors. Also, there is a lot of pre-processing needed in order to get a good kNN algorithm, like normalizing the data and removing undesirable values.

2.2 Deep Learning

Deep learning is a subset of machine learning. It consists basically in multiple layers of nonlinear processing in order to extract valuable features at each layer. The output from a layer is always the input for the next layer.

This branch of machine learning is specially good to analyze bigger and more complex datasets, such as speech recognition, computer vision, language processing, time-series

data and many other kinds of data. Its success has shown improvement over the years and it is considered state of the art to many artificial intelligence tasks [9]. This evolution is shown on Figure 4, where the best result for classifying a specific public dataset that consist of 100 classes of images (CIFAR100) so far was obtained on May 2018, with an accuracy of 89.33%.

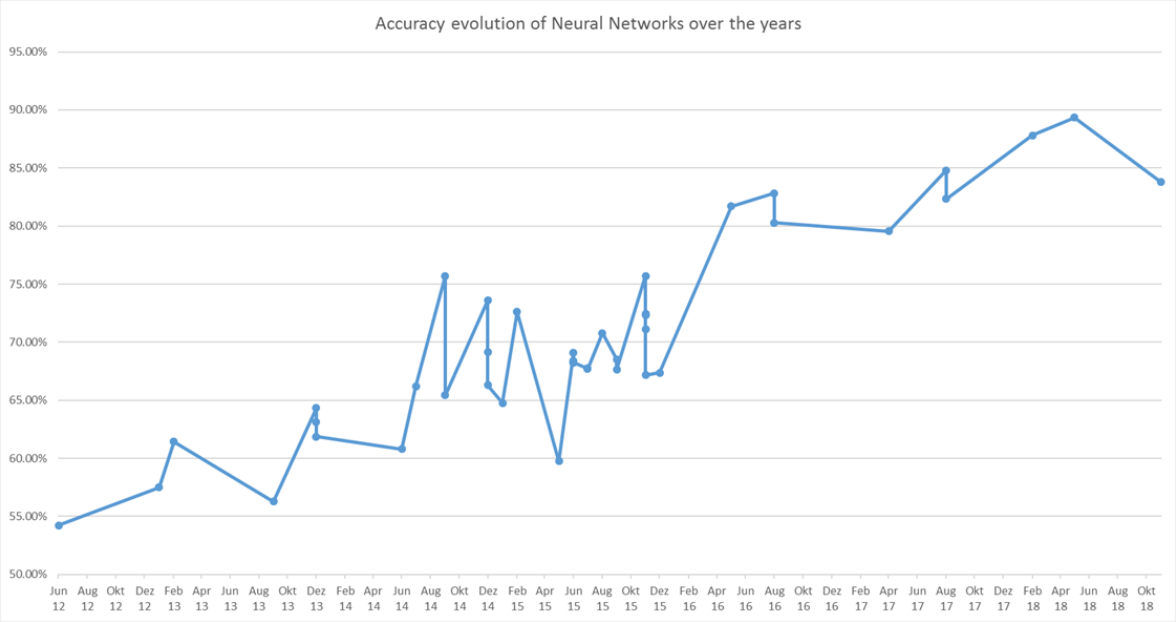


Figure 4 – Neural networks accuracy evolution of published papers for the CIFAR100 dataset [2]

The evolution of algorithms in just 6 years shows that the accuracy went from 54.23% to 89.33%, or an absolute increase of 35.1%, thanks to advances obtained in the deep learning field.

A downside of deep learning algorithms is the fact that it requires much higher computational power as the complexity of the algorithms and the number of layers increase. In order to provide this computational power to algorithms, it is possible to use Graphics Processing Unit (GPU) instead of ordinary Central Processing Unit (CPU). This makes the learning rate much faster. According to benchmarks [3], the use of GPUs instead of CPUs means an increase up to 16 times in the efficiency considering performance per watt. The results can be seen in Figure 5

The main reason that GPUs are in general more efficient than CPUs in deep learning is that GPUs have many simple cores instead of just a few very complex cores as seen in CPUs. This allows GPUs to make many operations at the same time resulting in a significant speed increase even if the single processing itself is slower. Also, the CPU is optimized to do a variety of operations such as managing peripherals, that is, these are tasks that a GPU does not have to do, so it is not optimized for that. Instead, GPUs are optimized exclusively for data computations, and this is what makes GPUs much more

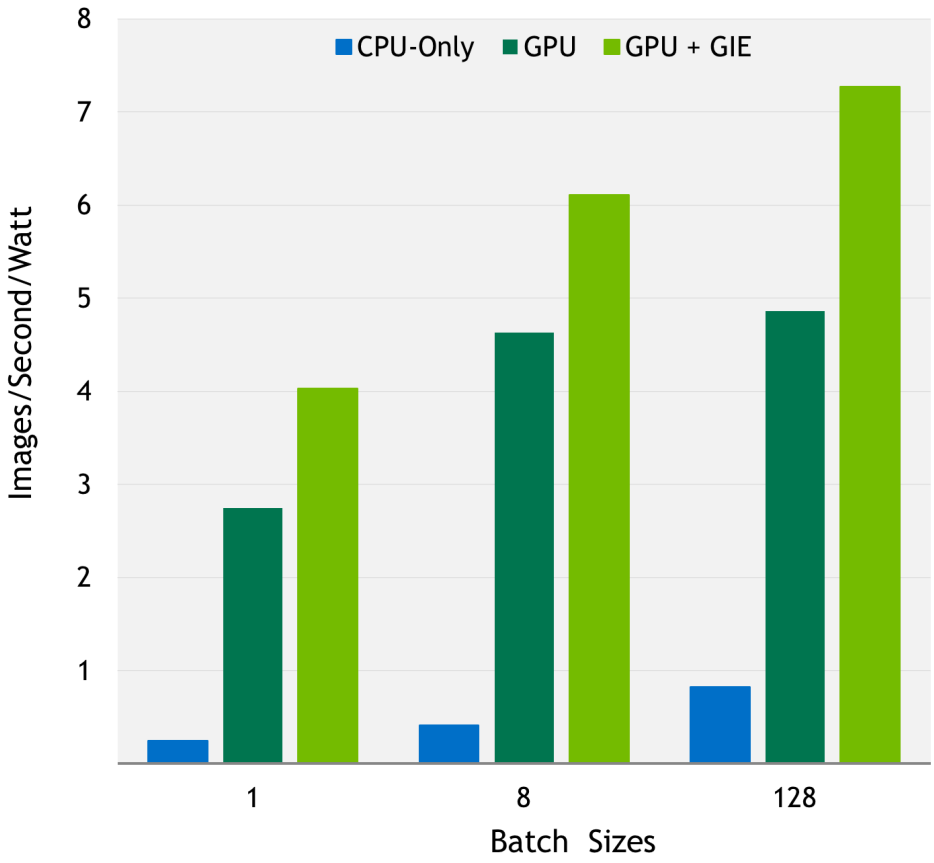


Figure 5 – Comparison of performance per watt of deep learning in CPU and GPU [3]

efficient in deep learning, where the amount of data computations required to train a model is huge.

2.2.1 Digital Image

The human vision has evolved deeply during the history, allowing to differentiate many objects, recognize a person almost instantly and we tend to rely in it, although the world is at constant change. If we stare long enough at something, it will most certainly change at some point, due to shadow, sunlight or any other factor. Despite that, humans know that it is still the same scene in a different setting, when computers do not know that if we do not tell them. To solve this problem there is the tool of image processing, in which it is possible to change the appearance of an image, like rotating, darkening, filtering the edges and many more.

But differently than humans that are predominantly visual creatures, computers see all data as numbers and with images is no different. In order to understand Convolutional Neural Networks (2.2.2) applied to visual data, it is necessary at first know how exactly this works.

A computer see each pixel of an image as a number, or a sequence of numbers. In

the case of an image being grayscale, the computer only sees one value varying most commonly from 0 to 255 or 8 bits. If the image is colored, the most common way of a computer processing an image is the RGB color model. In this color model there are three layers of numbers also from 0 to 255 for each layer for each pixel, being one for red, one for green and the last for blue. There are other models used mainly in the printing process that will not be discussed in this text.

To illustrate this, it is possible to decompose the UFSC logo into matrices of values, and see exactly how the computer processes the image. In Figure 6 it is possible to see how the human eye sees it, and in Figure 7 how the computer sees it.

Figure 6 – Black and white UFSC Logo

```
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,241,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,70 ,142,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,119,0  ,145,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,244,31 ,34 ,0  ,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0  ,59 ,91 ,29 ,173,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,24 ,0  ,0  ,64 ,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,204,238,244,242,234,208,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,204,247,244,244,247,207,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,208,225,229,208,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,81 ,128,255,206,0  ,31 ,6  ,0  ,4  ,7  ,6  ,11 ,18 ,17 ,11 ,7  ,6  ,6  ,6  ,6  ,6  ,0  ,216,255,238,160,255,255,255,255,255,255,255
255,255,255,255,255,255,205,0  ,153,255,203,0  ,255,37 ,0  ,255,34 ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,217,255,149,0  ,255,255,255,255,255,255,255
255,255,255,255,248,255,175,0  ,255,255,209,0  ,32 ,0  ,0  ,70 ,23 ,0  ,6  ,4  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,217,255,255,0  ,255,255,255,255,255,255,255
255,255,255,255,91 ,0  ,211,206,196,111,212,0  ,0  ,0  ,75 ,38 ,88 ,255,140,90 ,11 ,18 ,18 ,18 ,18 ,20 ,20 ,21 ,0  ,223,177,235,232,255,125,229,255,255,255,255
255,255,255,255,244,94 ,14 ,145,0  ,14 ,221,102,188,0  ,119,81 ,67 ,255,220,199,228,235,224,235,229,209,212,198,243,224,58 ,0  ,156,66 ,0  ,255,255,255,255,255
255,255,255,229,61 ,175,175,169,102,255,207,6  ,55 ,0  ,0  ,0  ,47 ,7  ,213,192,215,203,254,211,218,193,210,241,209,209,251,163,153,204,225,228,255,255,255,255
255,255,255,255,0  ,0  ,123,217,22 ,48 ,230,0  ,0  ,0  ,0  ,145,139,0  ,13 ,19 ,17 ,17 ,12 ,16 ,16 ,19 ,17 ,13 ,0  ,227,128,0  ,164,202,0  ,75 ,255,255,255,255
255,255,255,255,255,68 ,116,22 ,68 ,240,213,0  ,0  ,0  ,0  ,47 ,102,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,219,255,53 ,88 ,107,83 ,255,255,255,255,255
255,255,255,154,59 ,239,211,250,166,207,211,181,188,194,195,181,174,188,187,187,187,187,187,188,192,198,188,188,181,201,231,237,255,207,216,71 ,255,255,255,255
255,255,255,222,1  ,0  ,224,31 ,0  ,152,211,255,255,196,200,255,255,255,255,255,255,255,255,255,246,146,255,255,255,190,255,0  ,0  ,181,0  ,72 ,255,255,255,255
255,255,255,185,214,109,71 ,76 ,184,255,198,255,255,255,88 ,14 ,153,255,255,255,255,255,212,72 ,12 ,219,255,255,255,188,255,219,178,158,202,240,239,255,255,255
255,255,255,11 ,0  ,157,226,187,19 ,128,208,255,255,255,255,192,0  ,37 ,255,244,255,168,0  ,82 ,255,255,255,255,255,193,239,0  ,133,229,118,0  ,210,255,255,255
255,255,255,253,92 ,8  ,114,0  ,80 ,240,203,255,255,255,255,255,160,0  ,12 ,12 ,22 ,0  ,0  ,198,238,255,255,255,255,189,255,120,14 ,162,16 ,110,255,255,255,255
255,255,251,57 ,91 ,224,170,188,80 ,179,207,255,255,255,130,24 ,0  ,104,50 ,106,0  ,171,174,1  ,0  ,155,255,255,255,198,189,119,255,189,242,78 ,171,255,255,255
255,255,255,41 ,3  ,31 ,169,0  ,9  ,225,201,255,255,231,0  ,110,34 ,31 ,111,24 ,70 ,228,82 ,23 ,31 ,0  ,39 ,241,255,196,255,0  ,5  ,189,0  ,0  ,255,255,255,255
255,255,255,236,122,139,92 ,134,216,243,199,255,224,19 ,43 ,255,248,124,0  ,23 ,9  ,0  ,62 ,224,255,255,0  ,255,255,200,213,186,236,131,211,224,253,255,255,255
255,255,255,34 ,0  ,146,199,57 ,0  ,221,202,255,116,0  ,44 ,55 ,0  ,21 ,69 ,40 ,62 ,116,0  ,30 ,176,201,0  ,155,255,207,136,0  ,145,135,41 ,3  ,255,255,255,255
255,255,255,216,99 ,4  ,77 ,46 ,229,255,193,255,255,41 ,0  ,8  ,196,255,183,255,255,203,242,109,0  ,0  ,16 ,152,255,188,255,188,185,90 ,45 ,164,255,255,255,255
255,255,255,255,95 ,73 ,145,255,73 ,6  ,255,177,255,249,0  ,0  ,67 ,182,255,255,255,155,255,63 ,0  ,0  ,237,255,176,255,0  ,119,153,249,131,193,255,255,255,255
255,255,255,255,81 ,0  ,10 ,69 ,1  ,235,255,232,183,255,135,30 ,152,179,255,255,255,249,243,82 ,69 ,95 ,255,176,230,235,149,41 ,96 ,7  ,0  ,232,255,255,255,255
255,255,255,255,255,200,130,89 ,255,48 ,36 ,255,244,173,214,252,248,255,255,255,255,255,255,255,255,217,160,244,255,0  ,76 ,255,157,223,255,255,255,255,255,255
255,255,255,255,214,72 ,53 ,98 ,86 ,19 ,224,255,255,255,232,193,174,209,255,255,255,255,214,187,205,232,255,255,255,186,69 ,90 ,87 ,0  ,53 ,255,255,255,255,255
255,255,255,255,247,10 ,52 ,22 ,63 ,255,255,255,255,255,255,255,254,220,161,237,243,186,231,255,255,255,255,255,255,255,207,163,191,57 ,236,255,255,255,255,255
255,255,255,255,255,255,255,242,251,89 ,255,255,255,255,255,255,255,255,255,179,184,255,255,255,255,255,255,255,255,239,86 ,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,109,71 ,94 ,42 ,57 ,62 ,67 ,73 ,15 ,219,186,187,224,3  ,68 ,34 ,44 ,29 ,54 ,113,10 ,208,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,182,84 ,57 ,75 ,130,110,79 ,86 ,97 ,90 ,108,105,0  ,0  ,0  ,0  ,0  ,162,10 ,125,121,80 ,41 ,110,57 ,36 ,80 ,190,255,255,255,255,255,255
255,255,255,255,186,18 ,109,124,105,42 ,0  ,55 ,73 ,86 ,121,106,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,110,111,76 ,77 ,40 ,1  ,49 ,119,162,54 ,0  ,198,255,255,255,255
255,255,255,255,100,65 ,42 ,0  ,19 ,189,242,255,255,255,255,255,129,17 ,210,228,222,239,13 ,104,255,255,255,255,255,242,179,14 ,4  ,70 ,0  ,113,255,255,255,255
255,255,167,59 ,6  ,0  ,0  ,0  ,148,255,255,255,255,255,255,255,79 ,175,255,235,232,255,255,26 ,255,255,255,255,255,255,255,142,0  ,0  ,0  ,5  ,53 ,173,255,255
255,255,255,255,0  ,0  ,111,250,255,255,255,255,255,255,255,240,0  ,255,255,228,228,255,255,74 ,162,255,255,255,255,255,255,255,245,107,0  ,0  ,255,255,255,255
255,255,255,114,131,255,255,255,255,255,255,255,255,255,255,173,99 ,255,255,221,220,255,255,171,122,255,255,255,255,255,255,255,255,255,121,113,255,255,255,255
255,255,255,250,255,255,255,255,255,255,255,255,255,255,255,255,254,255,255,223,225,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,249,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,255,255,134,0  ,0  ,0  ,0  ,0  ,0  ,255,255,255,112,0  ,0  ,0  ,0  ,0  ,237,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,255,132,0  ,0  ,0  ,0  ,0  ,0  ,0  ,255,255,13 ,0  ,0  ,0  ,0  ,0  ,0  ,235,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,255,255,255,255,255,255,255,0  ,0  ,201,255,255,255,255,255,255,112,0  ,21 ,242,255,255,255,255,255,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,255,255,255,255,255,255,255,0  ,0  ,243,255,255,255,255,255,255,15 ,0  ,215,255,255,255,255,255,255,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,0  ,0  ,0  ,0  ,255,255,255,97 ,0  ,0  ,0  ,0  ,0  ,88 ,255,255,0  ,0  ,248,255,255,255,255,255,255,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,0  ,0  ,0  ,0  ,255,255,255,255,82 ,0  ,0  ,0  ,0  ,0  ,64 ,255,0  ,0  ,248,255,255,255,255,255,255,255
255,255,0  ,0  ,255,255,255,255,0  ,0  ,255,255,0  ,0  ,255,255,255,255,255,255,255,255,255,255,255,255,255,0  ,0  ,255,10 ,0  ,228,255,255,255,255,255,255,255
255,255,36 ,0  ,58 ,255,255,46 ,0  ,57 ,255,255,0  ,0  ,255,255,255,255,255,255,255,255,255,255,255,255,255,0  ,0  ,255,97 ,0  ,36 ,255,255,255,255,255,255,255
255,255,230,0  ,0  ,0  ,0  ,0  ,0  ,241,255,255,0  ,0  ,255,255,255,255,255,255,255,0  ,0  ,0  ,0  ,0  ,0  ,0  ,49 ,255,255,0  ,0  ,0  ,0  ,0  ,0  ,0  ,235,255
255,255,255,239,83 ,0  ,0  ,91 ,246,255,255,255,0  ,0  ,255,255,255,255,255,255,255,0  ,0  ,0  ,0  ,0  ,0  ,96 ,255,255,255,255,104,7  ,0  ,0  ,0  ,0  ,243,255
```

Figure 7 – Matrix of pixel values for the black and white UFSC Logo

Note that the value 255 from the matrix is the whitest pixel and 0 is the darkest pixel, comparing with the original image. Figure 6 was resized to only 40x54 pixels, in order

to be easier to visualize the Figure 7 matrix, which has exactly one value for each pixel, meaning it has 40 columns and 54 rows..

If we take a look now at a colored image, this single matrix becomes three, one for each color of the model (RGB). To observe this, take as example the Figure 8. Note that it has many colors, but the image can be decomposed to just three single-color images, exemplified by the Figure 9. If we sum the three new pictures to one another, the result would be exactly the original image.

Figure 8 – Colored UFSC logo.

With the Figure 9 it is more clear that there is actually three channels composing a single colored image. Now, it is possible to extract how the computer sees each channel, with the Figure 10 for the red portion, Figure 11 for the green portion and Figure 12 for the blue portion. In order to recover the original file, the only thing that the computer needs to do is concatenate each single pixel with the other two in the correct order. The order may depend on the software used, since there are softwares that use RGB and others that use BGR.

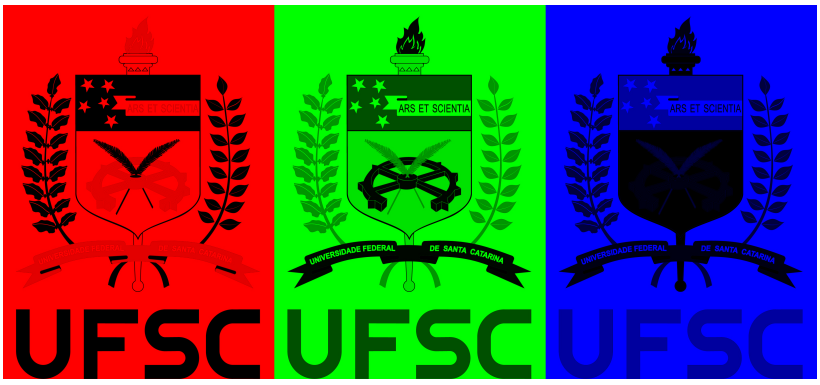


Figure 9 – Decomposition of the colored UFSC logo.

The representation of these single color images is not a single 1-dimensional matrix as the figures 10, 11 and 12 show. The actual representation is a matrix with the same size filled full with zeroes in the excluded channels. If the computer takes these matrices as they are, it would interpreter as a grayscale image, as seen in the Figure 7. In order to a

computer interpret an image as a colored in the RGB color model, it must have the three channels. Despite that, the only relevant channel to be considered for each single color image is the respective to that color.

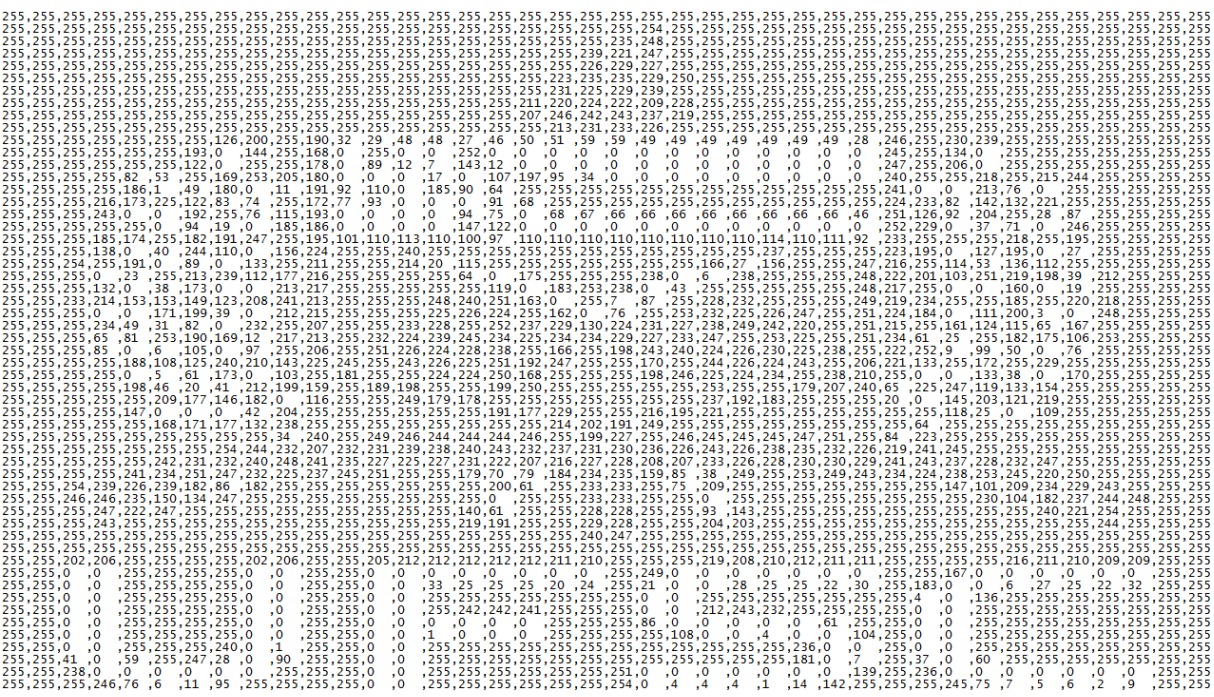


Figure 10 – Matrix of pixel values for the red portion of the image.

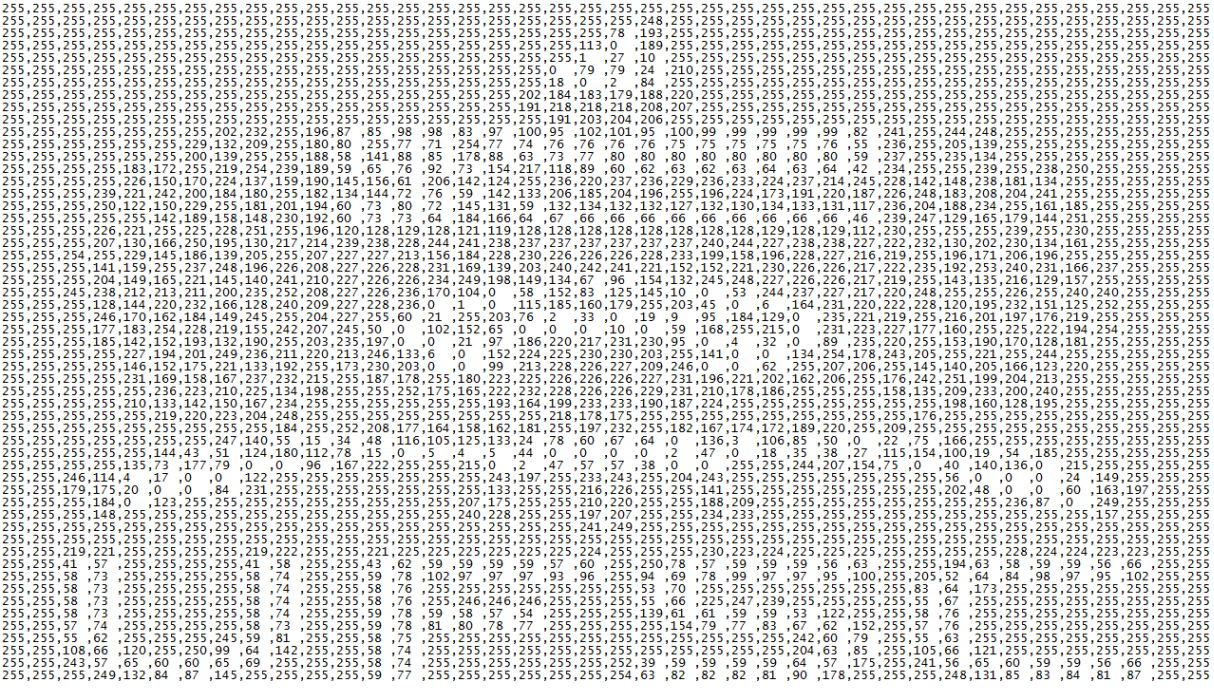


Figure 11 – Matrix of pixel values for the green portion of the image.

Note that where a color is more present in a given area of the Figure 8, the respective decomposed image and its matrix for that color will have higher values in the respective

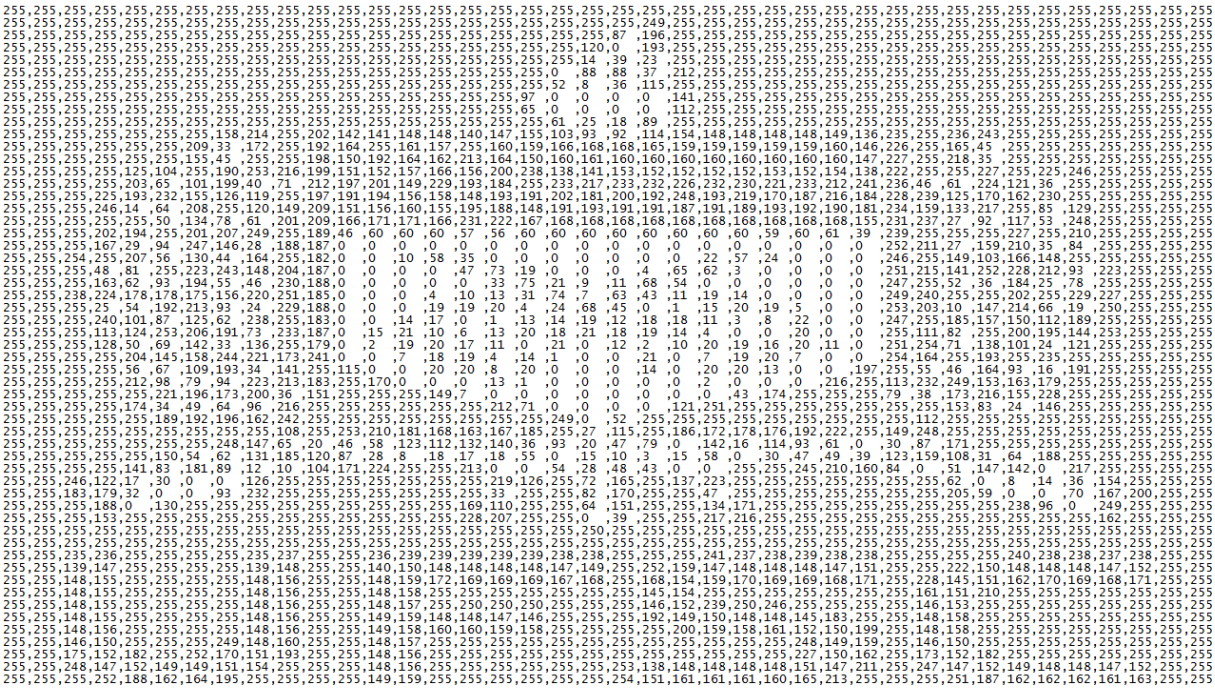


Figure 12 – Matrix of pixel values for the blue portion of the image.

area. Areas where a color is not present have value zero, that is the color black. Areas where a color is very present, there will be high values (close to 255), that is full color.

It is also important to take notice of the borders. For all of the three decomposed images, the borders have values of 255 in most of its extension. This is due to the fact that the junction of the three colors result in white, which is mostly the color of the original border, except the bottom. Speaking in terms of how a computer interprets it, an array with value $[255, 255, 255]$ will be a white pixel, $[255, 0, 0]$ will be a red pixel, $[0, 255, 0]$ will be a green pixel, $[0, 0, 255]$ will be a blue pixel and finally $[0, 0, 0]$ will be a black pixel.

2.2.2 Convolutional Neural Networks

Convolutional neural networks are just a special kind of a normal neural network. They have layers and neurons with trainable weights and biases, just as any neural network, but they are specially designed to be able to detect patterns, being able to identify resembles between neurons. This kind of neural network is neurobiologically motivated, with the inspirations by the works of Hubel and Wiesel[10][11] (1962, 1977) in which they studied the visual cortex of cats and monkeys.

According to Haykin[5], a convolutional neural network is a multilayer perception designed specifically to recognize two-dimensional shapes with a high degree of invariance to translation, scaling, skewing, and other forms of distortion. These aspects of the neural network make it very suitable to use with visual data in the real world, like a process in a manufacturing line.

LeCun, Bengio and Hinton[12] stated that a convolutional neural network has a structure with three main forms of constraints:

1. **Feature extraction:** each neuron takes an input signal from a previous field in the last layer, thus making possible the extraction of local features. This makes less important the exact position of each feature (or each pixel in an image), as long as its position relative to the near features is indeed preserved.

2. **Feature mapping:** each computational layer of the network have many feature maps, which are regions where neurons share the same synaptic weights. It has the benefit of shift invariance that is forced to the operation of a feature map by using a kernel with small size, and also the benefit of reduction in the number of free parameters that is achieved by using shared weights, reducing considerably the number of parameters to be optimized, saving computational power.

3. **Sub-sampling:** After each convolutional layer, it is applied a computational layer in order to evaluate a little sample of each feature map. This evaluation can be by using the average, min or max pooling or sum of the analysis region. Basically, this makes a NxN map become a single value, decreasing the size of the layer. More details in section 2.2.2.2.

2.2.2.1 Convolutional Layer

The convolutional layer in Convolutional Neural Networks is the core concept to be understood. Basically, a filter (or kernel) is created and convoluted to each set of values with the same size as the kernel size in the previous layer's matrix, in order to produce a convoluted layer. The matrix convolution formula is given by (2.1).

$$V = \left| \frac{\sum_{i=1}^{q} \left(\sum_{j=1}^{q} f_{ij} d_{ij} \right)}{F} \right| \tag{2.1}$$

Where:

$f_{ij} =$ coefficient of a convolution kernel at position i,j in the kernel

$d_{ij} =$ data value of the pixel at position i,j in the image

$q =$ dimension of the kernel, assuming it is square (q x q)

$F =$ sum of the coefficients of the kernel (1 if the sum is 0)

$V =$ output pixel value

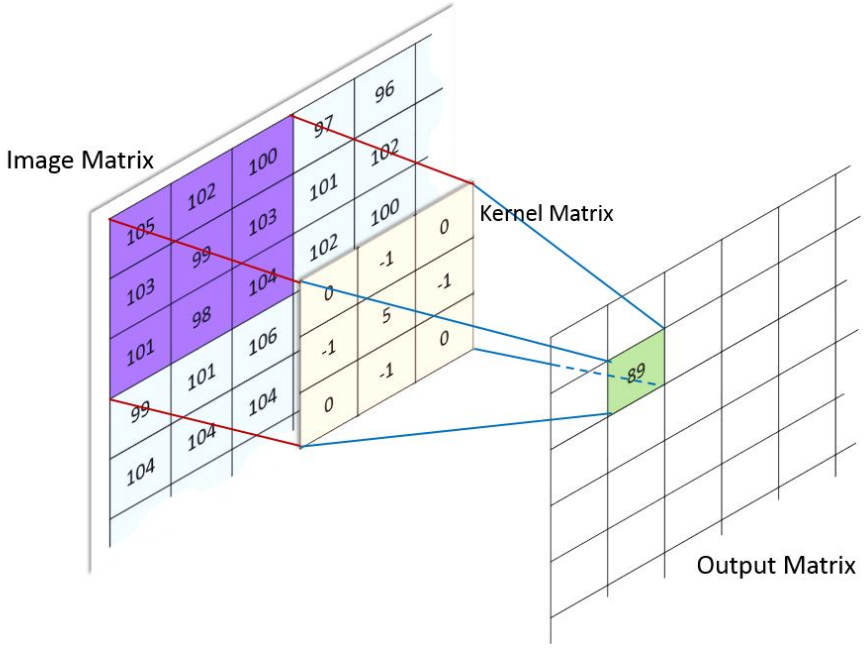


Figure 13 – Example of image convolution [4]

The kernel slides through the whole image matrix, in order to produce a new matrix. Usually, the stride used is 1, that is at every step, the kernel matrix moves one pixel to the side in order to perform a new convolution.

Note that the kernel size in the Figure 13 is 3 x 3, and when the convolution of the first 3 x 3 elements of the image matrix happens, there is a blank row and a blank column in the output matrix. This is due to the fact that the filter (or kernel) of size 3 x 3 slides through the whole N x N matrix and performs the convolution, and can not place its matrix center at the edge of the matrix. For a kernel of 3 x 3 the number of blank columns and rows at the edge is always 1 at each side of the output matrix. For a kernel with size 5 x 5 these blank spaces would be increased to 2. In a more general view, the number of deleted pixel rows and columns is going to be $(q-1)/2$ at each side, being q the size of the q x q filter.

There are ways of letting the size of the matrix stay the same by filling the borders of the original matrix with zeroes, but this approach will not be discussed in this text.

Alongside with the convolutional layer, it is common the usage of activation functions, like the rectified linear unit. The main purpose of this is to increase the non-linearity of the images, in an effort to make it easier for the computer to recognize certain patterns. There are already much non-linearity in normal images (e.g. borders, difference of colors, etc.), but for the computer would be better if those non-linearities were more expressive after a convolutional operation, that breaks non-linearity.

2.2.2.2 Sub-sampling Layer

Sub-sampling is the method used after a convolutional layer (or filtering). It has a window of a given N x N size (usually 2 x 2) that slides through the entire image matrix, taking just the N x N correspondent values for the current step in the image matrix and making some kind of operation based on these values. The operations can be:

- Max Pooling: Compares all the values in the window and saves just the maximum value for the current step.
- Min Pooling: Compares all the values in the window and saves just the minimum value for the current step.
- Sum Pooling: Sums all the values in the window and saves it.
- Average Pooling: Takes the average of the values in the window and saves it.

Alongside with the window size, another common parameter is the stride. The stride is nothing more than the step that the window takes each time. For example, if a pooling layer has a window size of 2 x 2, with a stride of 2, it will jump 2 pixels and therefore not repeat any numbers already evaluated. If the stride was 1, it would repeat the last column or row at each step. The Figure 14 shows the max pooling sub-sampling process, in which for each 2 x 2 matrix that there is in the original 8 x 8 matrix, it only takes the higher value and saves in the equivalent position in another matrix [13].

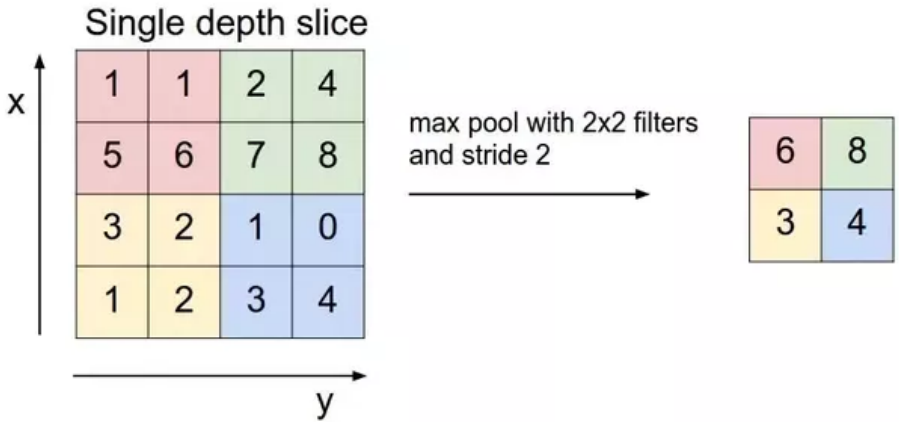


Figure 14 – Example of Max Pooling

The main purposes of this method are to reduce variance and decrease the size of the layer. Each step that a sub-sampling process with a window size of 2 x 2 and stride of 2 occurs, the data reduces by 75%.

2.2.2.3 Layout of basic CNN

The layout of a Convolutional Neural Network model can vary as much as the designer wants, but at the essence it basically consists of an input layer, at least one hidden layer

and an output layer. Figure 15 shows the architecture of a simple convolutional neural network made with an input layer, four hidden layers, and an output layer. This network was specially designed to perform image processing.

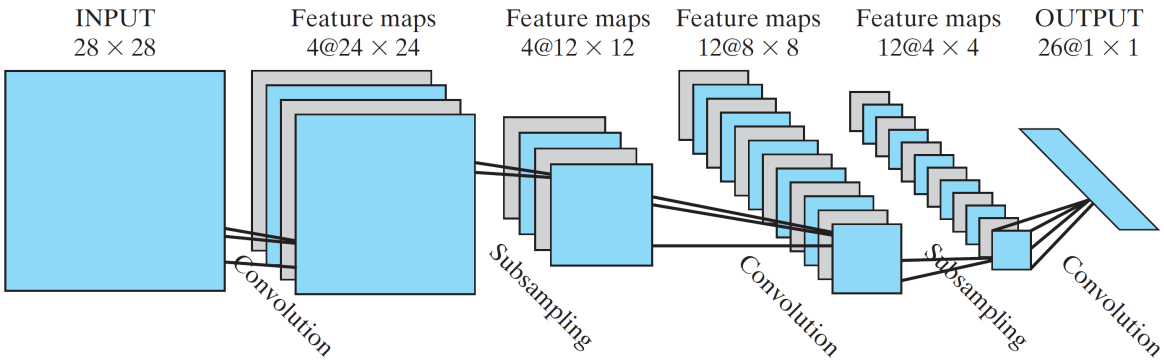


Figure 15 – Example of convolutional neural network [5]

This simple example of convolutional network takes as input images of size 28x28 pixels, containing 26 different centered and normalized handwritten characters. These pixels are read by the computer as simple numbers, therefore producing a numerical matrix, explained with details in 2.2.1.

Basically, the networks consists in alternate layers of convolution and sub-sampling:

1. The first hidden layer performs the convolution. The output is four feature maps of size 24x24 produced by a filter of 5x5 size. The decrease of the input size is explained in the section 2.2.2.1.

2. The second hidden layer performs sub-sampling, taking the local maximum value of a window of size 2, reducing by half the rows and the columns. The overall concept is explained in the section 2.2.2.2

3. The third hidden layer performs again a convolution, decreasing again the size of the feature map.

4. The fourth hidden layer is again a sub-sampling of a 2x2 filter and stride 2.

5. The fifth and last hidden layer is again a convolution, resulting in the output layer. This layer consists of exactly 26 neurons assigned to each possible character, giving the final value for each.

As the processing occurs there is an interesting phenomenon, where the system starts taking the shape of a dual pyramid. That is, after each layer, the feature map size decreases while the the number of feature maps increases, compared with the previous layer.

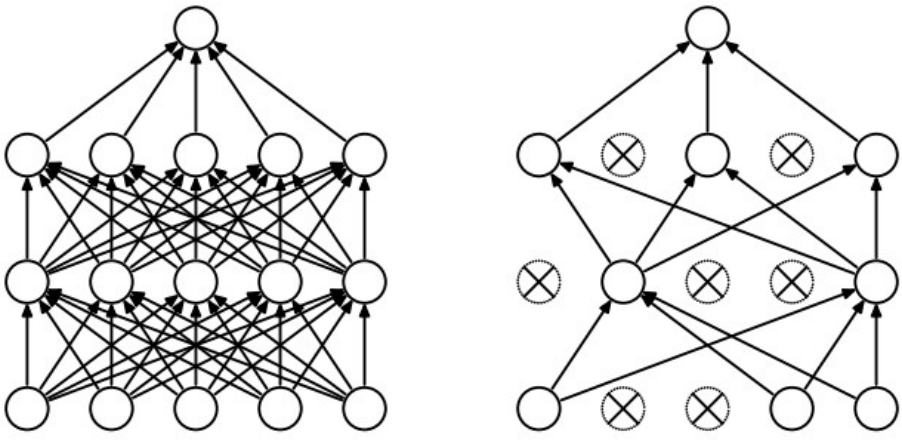
However, CNNs in the practical use are much more complex than the one presented. They will have more convolution and sub-sampling layers, along with other kinds of layers, like dropout and dense layers.

2.2.2.4 Dense Layer

A dense layer is a special kind of a fully-connected layer. In a fully-connected layer, as the name suggests, every neuron from the output of a dense layer is connected to every neuron on the previous layer, with each connection having a specific weight. This kind of layer comes after all of the convolution and pooling layers. The purpose of this layer is to do the final classification of the data. The last layer will be a fully-connected layer with the number of neurons correspondent to the number of classes the network is supposed to distinguish. Usually, the last dense layer is connected to a probabilistic distribution activation function, like the Softmax [14] function. This is a translation invariant normalization across all neurons in the last layer.

2.2.2.5 Dropout Layer

The dropout layer acts as an improvement in order to not over-fit the training of the neural network. It works by deactivating a given portion of the neurons on a particular layer, improving generalization. The Figure 16 shows this in a visual and easy way to understand. The circles (neurons) with an X are deactivated just in part of the training and then activated again, giving space to other neurons be deactivated. This is efficient because it forces the network to learn different paths of reaching the same result, making it much more robust.



(a) Standard Neural Net (b) After applying dropout.

Figure 16 – Standard neural network on the left and after dropout on the right [6]

This way, the error of the algorithm decreases as the network is being trained. Without it, the phenomenon of overfitting comes in to place, where the network focuses on

insignificant aspects of data in the training dataset, making it harder to predict correctly unseen data. This phenomenon makes the network worse as it is being trained after a certain point, thus there is a limit for each network to be trained. Even with dropout, the network can be overfitted, but it is less likely. Figure 16 shows in a real example the difference of using it properly in a network.

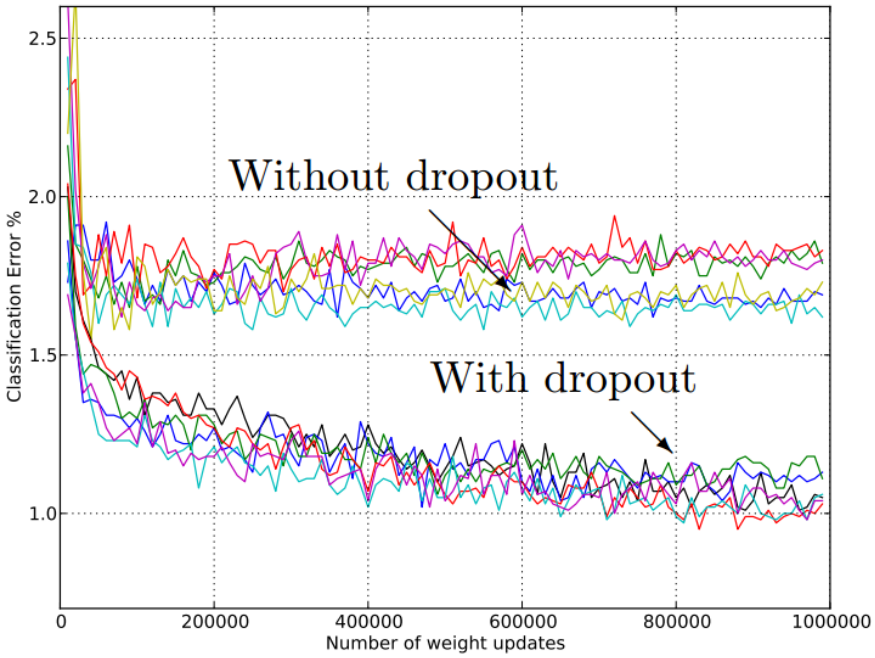


Figure 17 – Difference in the usage of a dropout layer [6]

2.2.3 Digital Audio

Similarly as seen in 2.2.1, the computer does not process audio the same way as humans do. They treat this kind of data also as arrays of numbers. Each audio file is read by the computer as one big array of numbers representing each sample of a waveform.

In order to understand this, it is necessary the previous knowledge of how the sound is propagated through waves. These waves, when captured by a recording device, are taken in samples with a given frequency. This is called the sampling rate. The bigger the sample rate, the lower is the loss of data from the original audio file, and the better is the sound quality. But this comes at the price of higher file size and thus require more computational effort in order to make further analysis, like machine learning. Figure 18 shows the different cases that can occur when sampling an audio file.

Nyquist-Shannon sampling theorem states that in order for a signal to be rebuilt, the sampling frequency f_N must be at least double of the maximum frequency f_m of the original signal [15]. This can be seen by equation (2.2).

$$f_N >= 2 \times f_m \tag{2.2}$$

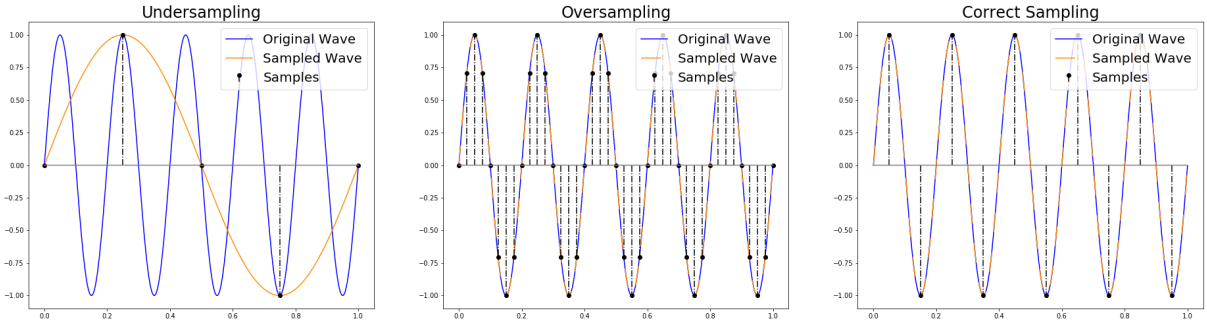


Figure 18 – Different sampling rates for wave signals

2.2.4 Vibration Data Acquisition

Vibration is closely correlated to audio because one of the side-effects from the vibration process is exactly audio. In fact, there are also vibration speakers available on the market, which are plugged into any surface and, as the gadget vibrates in a specific frequency, it produces sound through the material it is plugged to.

So, if it is possible to precisely measure the vibration of an element, it is possible to make assumptions about the sound that it will make and vice versa. Knowing this fact, it is possible to use vibration sensors instead of proper microphones to indirectly measure noise emitted by a machine. One big advantage of using vibration sensors instead of microphones is that they are immune to noise coming from undesirable places, such as other machines and even the environment as long as these outside vibrations are not extremely strong to the point it can interfere the desired measurements.

There are several ways of measuring the vibration of a machine, and the most common types are listed below [16][17][18].

- **Accelerometer**

  An accelerometer is a device that measures the acceleration usually in 3 axis, but can also measure in less. It is the most popular solution for vibration measuring, with low cost and acceptable accuracy for most applications. On the downside it needs to be placed directly on a flat and preferably metallic surface of the device in order to get accurate vibration data.

- **Strain Gauge**

  A strain gauge is simply a device that measures the strain in a material. It consists in a foil with an electrically conductive grid and, as the material moves, the electricity inside the device chances and therefore produces a value that is proportional to the displacement of the material. It is a very accurate low-cost device that can be used in almost any surface because it has flexible properties. The downside is that it requires additional equipment in order to amplify the signal.

- **Microphone**

  There is no mystery in a microphone for industrial vibration measuring. They are the same as common microphones, except that they might have better performance in industrial environments. Microphones are great in the aspect of generalization, being able to be set up in almost any environment, but with the difficulty of capturing sounds that might not be interesting to measure. Microphones might not be good if the interest information is vibration itself, but they do a great job if the information needed is the change of sound behavior with time.

- **Laser Displacement**

  A laser displacement sensor is a device that emits a laser beam to the object of interest to be reflected towards a light sensor inside the device. As the position of the reflection changes over the light sensor, it interprets as a change of displacement of the object. This kind of sensor is highly expensive and also needs to meet a lot of requirements, such as a fixed mount point and distance between the sensor and the object, making it difficult to practically implement.

- **Vibration Meter**

  Vibration meters are devices that already have batteries and an interface. They are used to measure data instantaneously by just pressing it against the object of interest. Although they are easy to use and to start acquiring data, they are not a permanent solution. They often include a traditional accelerometer inside and require a technician to periodically check the machines with this device.

- **Vibration Data Loggers**

  Vibration data loggers are ready solutions to easily install in machines. They have accelerometers, batteries and storage with them. They are also quick to install and do not need much knowledge to start logging data, unlike traditional accelerometers that need some work to get proper data. Besides that, they are hard to synchronize and normally have worse accuracy.

2.2.5 Recurrent Neural Networks

Some systems or problems can not be transformed in simple and fixed inputs with simple and fixed outputs, like the prediction of a single image. In general, problems involving time-series data, such as speech recognition, need to be stored and accessed in order to get the full context for the data. To do so, Recurrent Neural Networks (RNNs) take as input the output of previous layers. So in fact, when a RNN is being trained, it is still using information from the past to make assumptions to the present. [19]

There are several examples in which RNNs can be found. Besides the already mentioned speech recognition problem, there is also sentiment analysis, prediction of stock market, caption of images and many other applications. In the case of sentiment analysis, the input is a classification sentence with multiple words, and the output is a classification for whether the current sentence is positive or negative.

There is a lot of speculation around RNNs. Some authors suggest that RNNs are not as efficient as thought for many years and it might be good to change to other types of neural networks [20][21], while many other authors are still using recurrent neural networks and its features to produce good results [22]. It is not the objective of this work to compare the effectiveness of RNNs with other structures for the same datasets, but just to analyze its results.

But this kind of neural network has a special problem when used by itself. They are designed to use values from the past to predict values from the present, but when the gap between past and present grows this kind of network can not connect the information precisely anymore. This is because of the vanishing gradient problem, discovered by Hochreiter[23] in 1991. The problem is due to the fact that in order to get past values or weights, the network uses a gradient to multiply the actual weight and the lower the gradient is, it becomes harder for it to update its values and also takes longer to reach a solution. As a result, the network is able to train properly just for the part very close to present and ignore past values, which is the main objective of using recurrent neural networks.

Many alternatives were proposed to solve the problem of the vanishing gradient. The most important of them is Long Short-Term Memory (LSTM) units, which is explained in section 2.2.5.1.

2.2.5.1 Long Short-Term Memory

Long short-term memory is a technique for improvement of neural networks that has shown great results. It has the implementation of a memory cell that is able to keep information for longer time without causing the vanishing gradient problem. This is done by the addition of new units and a re-designed workflow in each layer. [24]

This re-designed workflow is just the addition of a set of gates where the information passes through the memory cell. These gates determine which information to keep and which information to delete, based on the importance given for the current data. This allows the network to learn which information is important over time.

LSTM networks were first published by Hochreiter and Schmidhuber[25] in 1997 with great results for some applications. In the beginning of the 21st century, LSTM based networks made a revolution in technology with the popularization of voice assistants.

2.3 Machine Learning Frameworks

Machine learning (or deep learning) frameworks provide the capacity of programming, training and testing neural networks with high level interface. The most used frameworks for this purpose are TensorFlow [26], PyTorch [27], Keras [28] and many others widely available. These frameworks provide different ways of programming with the goal of giving a better level of abstraction together with the simplification of challenging programming problems.

2.3.1 TensorFlow

TensorFlow is an open-source framework that supports Python, C++, Java and Go, with the first being the best supported language. It is developed by Google Brain and its first stable release was in 2017, despite the initial release being dated from 2015. This is the most used deep learning framework for a series of reasons.

This framework is capable of computing many types of machine learning such as classifications, regressions until the most complex neural networks. It also has support to GPU computation, which helps a lot on the processing time of the algorithms. For better visualization, TensorFlow has the visualization feature TensorBoard which is as a powerful tool to display live graphs and plots for some given metrics. Another great feature of TensorFlow is that it also has the capability of running on mobile, so the possibilities are many while using this framework.

For the extension of this work, this will be the main framework used because of its great performance and flexibility. With it, there is the possibility of running the models in a completely different system without many problems, making it easier to run the same algorithms in multiple devices.

2.3.2 PyTorch

This is considered one of the biggest competitors of TensorFlow, achieving impressive numbers of adoption by developers. It is also an open-source library but it is exclusively developed for Python. This framework is developed by the artificial intelligence resource group from Facebook with the initial release in 2016 and the first stable release in late 2018.

Comparing to TensorFlow, it is a more straight forward framework for creating deep learning algorithms, although it does not provide a good scalability or really complex and optimized solutions. This tool might be better in some cases, for example when there is need to change some parameters during training, but in overall the performance is worse than TensorFlow, without taking into account the fact that it does not provide natively

a solution for visualization like TensorBoard. Despite its qualities, PyTorch is not used in this work.

2.3.3 Keras

Keras is an extremely minimalist and high-level solution for neural networks. It is not a full solution like TensorFlow or PyTorch, but more like an extension of a framework, being able to run on top of TensorFlow, CNTK or Theano. This integration enables fast experimentation, being easy to change parameters and try new and different neural network settings without much difficulty.

It also have four main principles, which are user friendliness, modularity, easy extensibility and working with Python. These four principles are the core for the development of Keras, where the main points are developing simple and understandable code for human beings and being able to change a set of parameters or adding new modules without changing the whole code.

It was developed by ONEIROS and the first released was in 2015, with the first stable release in 2016, long before the first stable release of TensorFlow. Being also open-source, it connects very well with TensorFlow and all of its features work fine. The main features of Keras are blocks of layers, activation functions, loss functions, optimizers and a full toolbox for data processing. For this work, Keras is used with integration to TensorFlow.

2.4 Filtering

Most of the times when acquiring data from deployed sensors, the signal does not come with a perfect shape. If no filter is applied, there is always noise and this interference must be erased in order to get a good signal. [29]

To solve this problem, lots of filters are available for use. The most common digital filters are low and high pass filters, which cut any signal that have frequencies under or above a given cutoff value.

There are many ways to implement a high and low pass filters. The one used in this work is the Butterworth[30] filter, which is designed to have the most flat and stable response in the passband. There are many other methods to implement these filters producing different outcomes. The most common filters are the Chebyshev[31] with a good approximation of the ideal response, the Elliptic[32] which can have a faster change between the gain, suitable for operations where the cutoff frequency specified must be the most exact possible, and the Bessel filter that is able to keep a linear phase delay.

The objective of this section is just to give a brief overview of filtering. The mathematical approach will not be further discussed in this document. Sedra and Smith[33] provide more detailed information about the whole topic.

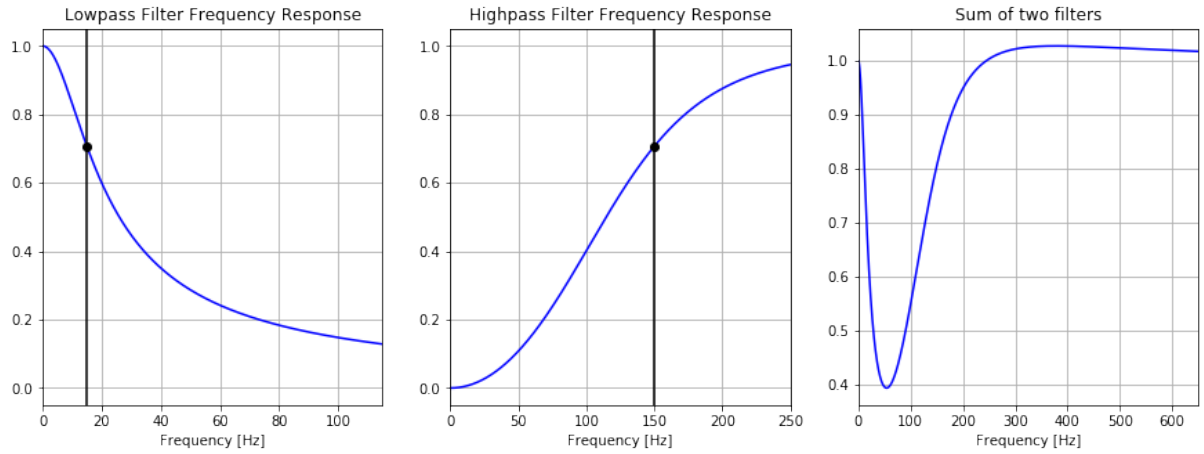


Figure 19 – Frequency response of digital filters

2.4.1 Low-Pass Filter

Low-pass filters are a kind of filter that rejects high frequency data. This kind of filter is very useful when the acquired data has high frequency noise and it must be eliminated. The low-pass filter is very useful for example, when an accelerometer is installed inside a car with the purpose to measure its acceleration, but there is also noise and vibration inside a car environment, which is undesirable. Low-pass filters act eliminating this noise and high frequency data that is not wanted for a specific system. Figure 20 displays a signal without any filter and the output when a low-pass filter is deployed to the system.

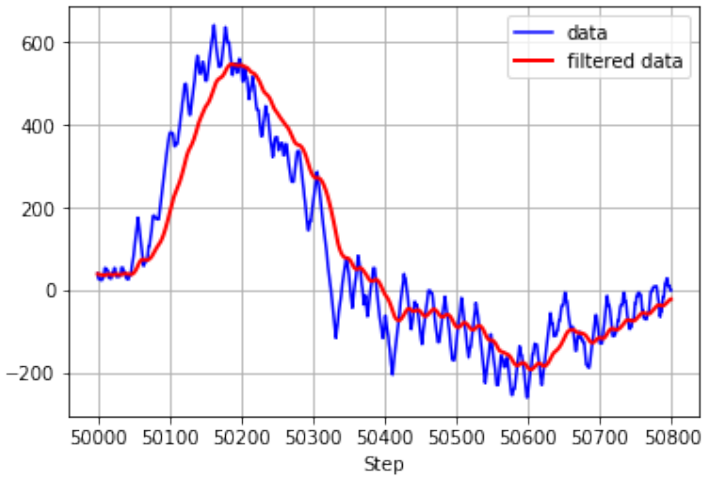


Figure 20 – Output of a low-pass filter

It is possible to see that the high-frequency data, which can be interpreted as noise in this case, is almost vanished from the system resulting in a smooth wave form. This may allow a better visualization and understanding of the system, depending on its properties.

2.4.2 High-Pass Filter

Opposite to low-pass filters, high-pass filters reject low frequency data. This filter is usually used when there are undesired changes in the overall data value. This way, the output of the system is just high frequency data without any offset. For example, it is useful when there is an accelerometer inside of a car that is installed for measuring the vibration of the car while it moves, but the acceleration of the car brings undesirable data for the dataset. The output of a high-pass filter in this case would be just the high frequency vibration data without the offset from the acceleration of the car. Figure 21 exemplifies the action of this kind of filter by displaying the original response and its output when a high-pass filter is applied.

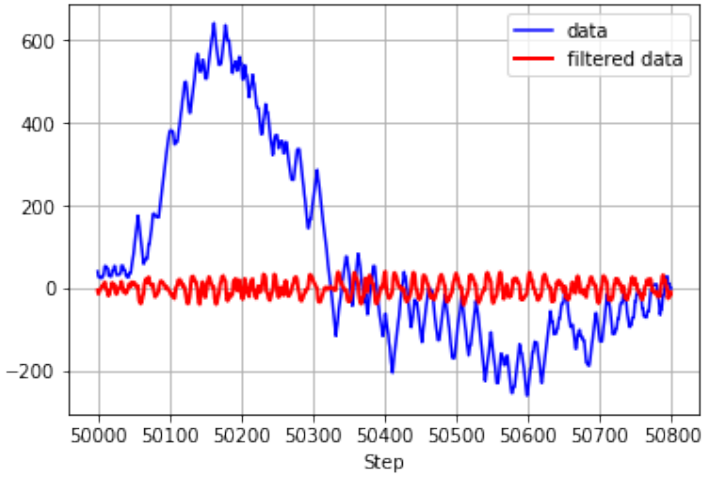


Figure 21 – Output of a high-pass filter

It is clear that the low frequency wave is excluded from the output of the filter, remaining just high frequency data with no change in the offset. When doing the sum of the outputs from a low-pass filter and a high-pass filter that have the same cutoff frequency, ideally the result should be the same to the original wave before the addition of any filter.

2.4.3 Other Frequency Filters

The main filters are the low and high-pass filters, but there are many filters that can be designed from these two. Common filter with this property are pass-band and reject-band.

The pass-band filter is when just data which is between a specified range of frequency is kept to the output. It uses one high-pass filter to specify the start of the passing band and one low-pass filter to specify the end of the passing band. This way, there is the exclusion of all values with frequency outside of the passing band.

The reject-band filter is very similar to the pass-band but the objective is the opposite. Instead of selecting a passing band, the filter excludes just a frequency range specified. To accomplish this, the reject-band uses one low-pass to specify the frequency it starts rejecting the values and uses a high-pass filter to start passing the values once again when the interval of the reject-band is satisfied.

3 Methodology

This chapter is about the methodology used during the construction of the project. From the most generic and high-level aspects until the deepest details necessary for the completion of the work.

The work is developed with an extensive research, followed by prototyping and testing. To accomplish good results, it follows a system of knowledge acquisition and development, consisting of literature research and study followed by replication of published results and finally development to practical implementation. By following this methodically, it is easy to make proposals for improvement or approach changes to already published works, even if those proposals are proven not to act as expected.

This methodology allows to create projects that are less likely to failure because of the background study. To develop new ideas from scratch after acquiring the base knowledge, it is necessary first to have a hypothesis that have to be done, then tested and finally improved. The documentation of everything in the whole process is very essential in order to give credibility and solidify the results.

3.1 Data Acquisition

Before entering the topic of data acquisition, it is important first to take a look into the process itself and how the acquisition could be done. The process consists basically in a horizontal injection molding machine and the goal is to observe if the mold is open or closed in real-time with the help of a single camera, two accelerometers and machine learning algorithms. Figure 22 shows the model of the injection molding machine used in all of the experiments.

The Arburg 220 S is a fully hydraulic horizontal injection molding machine with 250kN of clamping force weighting 2100kg, being considered very small for the standards. It is possible to find injection molding machines going up to 5000kN of clamping force. The machine was manufactured in 2001 but makes no more part of the company's catalogue. It is manufactured in Germany [34].

This machine is also fully covered with metal, being suitable for vibration acquisition. It also possesses a mobile chamber made of metal with a transparent window, making possible the visualization of the mold. This also makes the image acquisition from outside the machine possible. On its interior, it is possible to wire cables under the machine since it has an open space for wires, making the acquisition from inside the chamber also possible. The actual output of this machine is small plastic pen holders, being produced manually for educational purposes.

Figure 22 – Arburg 220S Injection Molding Machine

3.1.1 Camera

While talking about cameras, there is an infinity of possibilities available for choosing according to requirements. But there were two special possibilities of cameras for this work available for use at the Fraunhofer IPA institute. The narrowing for these two cameras was made because of availability and the fact that they are two smart cameras with capability of embedded scene recognition.

The two cameras are from different vendors but are used for the same purpose. They are industrial smart cameras designed specially for automobile manufacturing, but suited for many applications. They have industrial connections for power supply, communication, outputs and also an input for triggering. The comparison from both cameras can be seen in Table 1.

It is important to notice that these values are taken from the datasheets [35][36] from the cameras, but they do not necessarily match the real results. For most testing, the optimal framerate could not be reached in both cameras. The Sensopart V20-OB-A2-C had a special problem when trying to acquire fast data which made the output data to be asynchronous because of the lack of processing power. The Cognex Insight Micro 1403C was also not able to reach 7 FPS at full resolution, even with really simple pattern

Table 1 – Comparison of Smart Cameras

Characteristic	**Sensopart V20-OB-A2-C**	**Cognex Insight Micro 1403C**
Resolution	1280x1024 (resizable to 640x480)	1600x1200
Frame rate	40 FPS (SXGA), 80 FPS (VGA)	7 FPS
RAM	128MB	256MB
Flash Memory	128MB	128MB
Operation Temperature	0°C to 50°C	0°C to 45°C
Protection	IP65	IP51
Power Supply	24V DC	PoE
Color	No	Yes

recognition. For better results in the Cognex camera, it is possible to crop the image, lowering the resolution. The Sensopart camera has the function of lowering the acquisition resolution without losing the field or view or proportion.

Finally, the camera chosen for this project is the smart camera Cognex Insight Micro 1403C, which has built-in algorithms to detect patterns and save colored images with timestamps. For the purpose of this work, the built-in pattern detection of the camera is used only as a pre-processing of the images taken, in order to make it easier to separate the frames for each class (open or closed). More details are provided in section 3.1.1.3. Figure 23 shows the camera chosen for the project.

Figure 23 – Cognex Insight Micro 1403 C

The camera is very small and light, with dimensions of 30mm x 30mm x 60mm and 146g of weight. This allows the camera to fit pretty much anywhere without any trouble. It has also protection IP51, which provides protection against dust and drops of water.

3.1.1.1 Positioning

It is not so simple to find a good positioning for a camera to properly capture the movement of the mold of an injection molding machine while working. It has to take some aspects of the process into consideration like temperature, visibility and vibration.

According to the Cognex Micro 1403 C datasheet, the maximum operation temperature supported by this camera is only 45°C and the operation temperature of the injection molding machine inside the chamber is around 60°C, so it is not possible to position this camera inside the chamber for a permanent solution in an industrial environment. There is also the problem that the camera has to be fixed in a way that it vibrates the minimum possible, in order to maintain its current position and keep a certain fixed pattern along the frames. Another possibility is to place the camera outside of the injection molding machine, losing visibility of the process but with a much lower temperature and vibration.

For this work, the camera was positioned in both situations, inside and outside of the chamber. The reason why it was possible to fix this specific camera inside the chamber of the injection molding machine is because it is a controlled environment, without real production. The main objective of testing the camera inside the chamber was to verify if it has some improvements compared to placing the camera outside of the machine. Case positive, it is possible to exchange the camera for a more robust similar camera without losing fidelity to the results.

The positioning inside of the chamber is made by screwing a magnet mounting plate under the camera making the setup cleaner, embedding all cables inside the machine. The position outside of the machine had to have its own cable management, making it a little inconvenient and taking more space.

3.1.1.2 Acquisition

The connection to this camera is done by using the software In-Sight Explorer 5.7.0 and connecting to the camera via Ethernet connection. After that, it is possible to choose many options of resolution, shutter speed, triggering method, pattern selection and also the option to save classified images in a specified folder with timestamps on them.

The resolution of this camera is 1600 x 1200 pixels with a shutter speed in range of 16μs to 1000ms. The lower the shutter speed, the faster the image acquisition gets and the framerate increase, but with a low shutter speed, there might be some darkness in the picture. But as long as the shutter speed is not too low, it should not be a problem if the process has some ambient light, which is in fact the case in this work.

The camera also acquires images with 24 bit color (three layers of 8 bits each), with a maximum capacity of 7 frames per second at full resolution. Better results of framerate can be achieved if the resolution is lowered. The framerate can also change depending on the complexity of the patterns recognition tasks or the shutter speed.

3.1.1.3 Labeling

Labeling is a fundamental process while creating any machine learning dataset. It consists of assigning one or more status or outputs for some measured variables. This way,

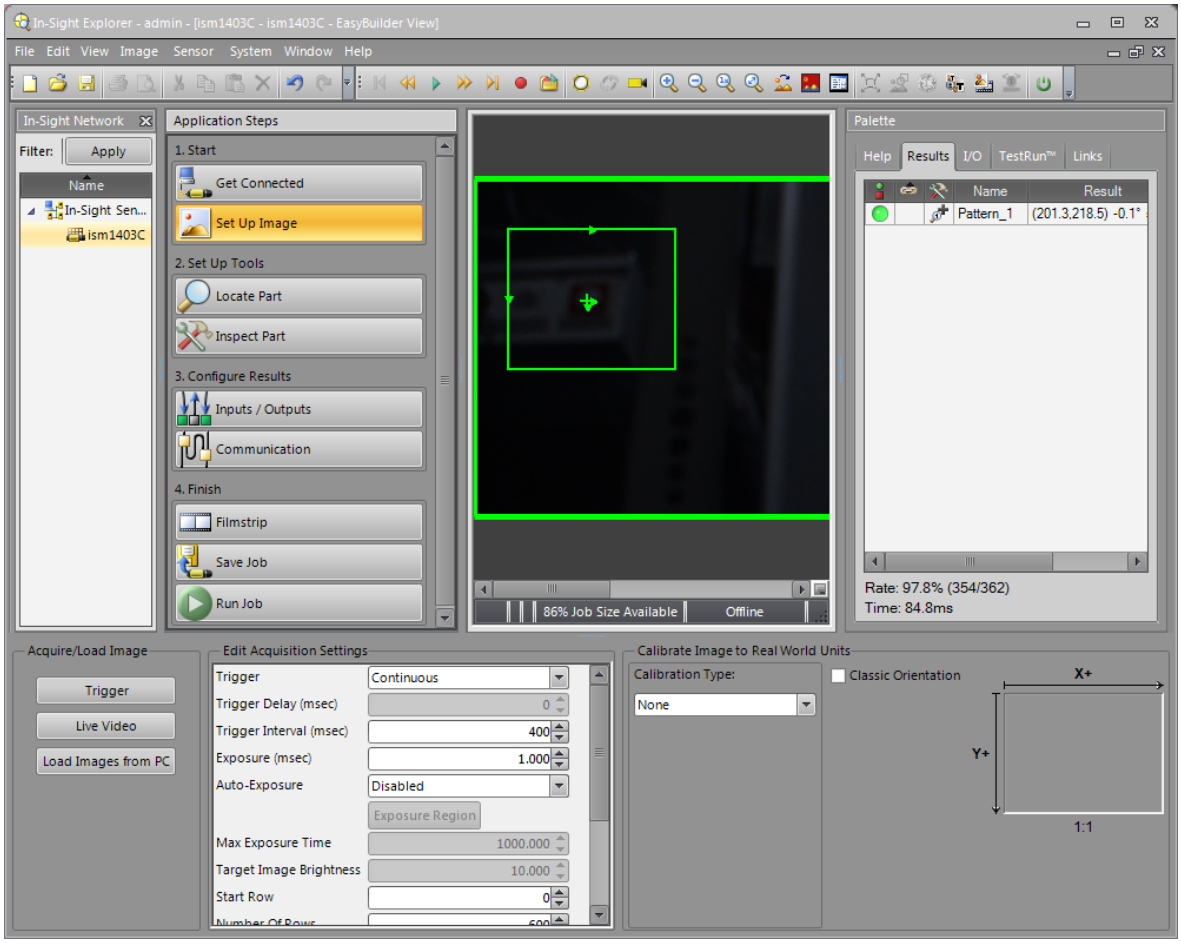


Figure 24 – Software In-Sight Explorer 5.7.0

there is a link between the inputs taken in a given time with the real status or output of the process at the same time. It is also important that these values are accurate, otherwise the algorithm would have wrong data to train, therefore being subject to make bad classifications. Without labeling, it is completely impossible to train a model unless it uses a Unsupervised Learning [2.1.2] method. For this case, as the objective is to create a Supervised Learning [2.1.1], it is necessary to label all of the data acquired.

As mentioned, the chosen camera has the built-in feature of image classification. This helps a lot the labeling work, but does not provide a reliable solution. The simple method the camera uses is not enough for a perfect image classification, therefore some post-processing is necessary.

In order to make the dataset reliable, after the pre-classification of the camera images, there is a manual correction of wrong values. In this case, the injection molding machine has no communication and does not provide any information about its status, otherwise it would be possible to automate the labeling process. The manual labeling is only possible because the dataset is small, otherwise manual checking would be impossible and other solutions would have to be taken. The result is a dataset of classified images between classes, which each measurement have its respective timestamp. In this case, the images

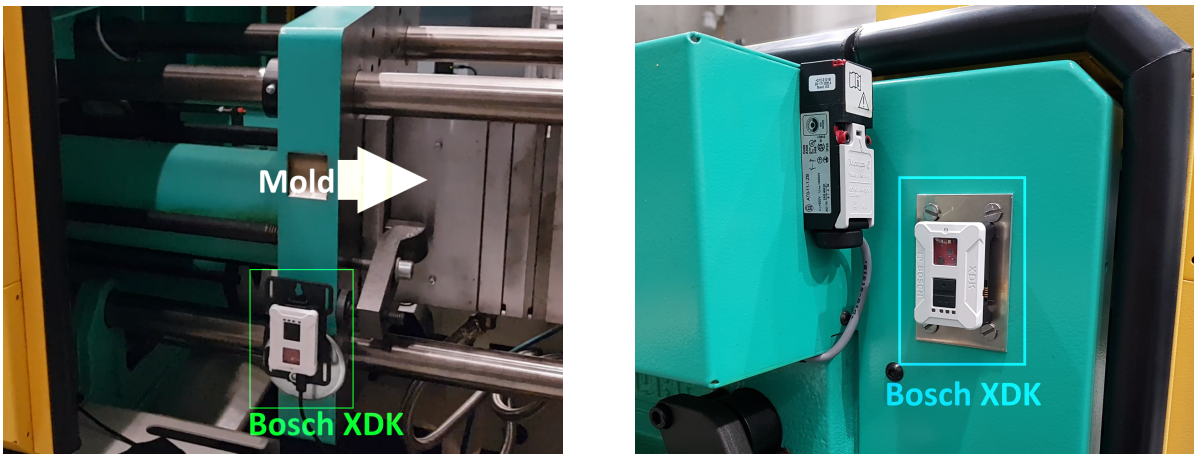
are labeled by the mold status (open or closed) and the separation is done by placing the images in different folders.

3.1.2 Accelerometer

After considering many different vibration sensors, the best approach seemed to be to use an accelerometer and the chosen for this role was the Bosch XDK, which is a development kit produced by Bosch. It comes with many different kinds of IoT (Internet of Things) related sensors, such as accelerometer, thermometer, gyroscope and others. The only required sensor for this task is the accelerometer. It could be considered as a waste of resource, but this sensor was already available for use and it has some good features, such as access to Wi-Fi, connection to NTP Servers for time source and most important the ability to communicate over serial port.

3.1.2.1 Positioning

It was used two Bosch XDKs to fully read the status of the machine. One of them was positioned right on the mould, so it was not effectively measuring just vibration, but also the acceleration of the mould. It gives much more readable data, as there is a good change in acceleration when the mold is opening or closing and is also fixed to the actual part of interest in the machine. To fix this device, it was used the default plastic mounting plate of the gadget in addition to a strong magnet. This way, there is no need of a permanent installation on the mold, as the magnet is easily removable. The setup for this XDK can be seen in Figure 25.a.



(a) XDK positioning on the moving mold

(b) XDK positioning outside the machine

Figure 25 – Positioning of the two vibration sensors

The other XDK, on the other hand, is used purely to measure the vibration of the machine, so it has to be static in order to get accurate data. Considering this, it was placed outside of the machine with a custom made metal mounting plate with dimensions

that match pre-existing holes in the machine's exterior. This way it is possible to fix the device on the mounting plate and then screw the mounting plate on the machine. Doing this, everything is connected by metal, transferring the maximum possible amount of vibration from the machine to the accelerometer. The setup for the XDK placed outside of the machine with the custom mounting plate can be seen in the Figure 25.b.

3.1.2.2 Acquisition

The programming of the Bosch XDK is done via a proprietary software called XDK Workbench, which uses C as programming language. The company does not provide ready solutions for data logging at the maximum possible frequency, and as this is one of the project's requirement, it was necessary to build a data logging program from scratch. After frustrated tries of making the gadget communicate wirelessly, it was finally possible to make it work sending data over USB at an acceptable rate. The output frequency of the logger for just the accelerometer in three axis (x, y, z) is around 2.5kHz, more than enough for good vibration measure [37].

On the other hand, sending data at this rate over USB brings some kinds of problems, such as wrong values due to bad connection and also it is an UART (Universal Asynchronous Receiver Transmitter) communication, so the data comes in small batches being more susceptible to errors. To solve this problem, the solution was to use a post-processing software to filter the data, so values that are outliers are not passed to the final output or at least are corrected. This is done by an algorithm that compares the actual value with the last measurements. If the new value is more than double of the previous value, the new value is divided by a factor and tested once again recurrently until the value is in an acceptable range.

It is also proposed to delete some values that are not very important from the training datasets in order to get a more even dataset and enhance the dataset quality. Originally, the datasets have more values from one status than the other. This makes the algorithm be more susceptible to predict values from the class with the majority of values in training if the data is not evened beforehand. This could help if the data is well-known for having more one value than the other, but most of the times it leads to wrong predictions. Therefore, it is important to select proper data that has few or no importance to the dataset. This means if at the beginning or at the end of the dataset it belongs to one class or another for a long time without any change and that class is in fact part of the majority class, then it is suitable to strip this data out of the dataset. This process makes the dataset smaller, but more even, which could be beneficial for the accuracy.

Another pre-processing method used is the Butterworth filter for low and high pass. More details about filtering can be seen in section 2.4. The results for using this filter can be found in chapter 4, as well as comparisons between different types of filters.

3.1.2.3 Labeling

The labeling of the vibration data is a little more difficult than labeling the images from the camera. The main reason is the frequency difference. While the camera records video at a maximum capacity of 10 frames per second, the accelerometer has a frequency of 2500Hz, so for every frame of video there are at least around 250 vibration measures.

As already mentioned in 3.1.1.3, it is difficult to label data without a reliable source of information about the real machine status. Because of this, some considerations were made for labeling the vibration data. The main consideration is that the real machine status changes at the exact same time as the camera data is labeled, so it is possible to have a synchronous time between the vibration and camera datasets.

As done for the camera images, the vibration data is also labeled with the same two status. But differently than the images, where they are placed in different folders, the vibration data is just a table with a timestamp and read values. The assignment for the real status of the machine is done by adding a new column to the table for each measurement.

3.2 Feature Selection

One crucial step for a good machine learning algorithm is a good feature selection. For example, if the actual problem is to find if there is a car or not in a specific parking spot and the only source of data is a camera of the whole parking lot. It makes no sense to analyze the full frame of the camera, because the point of interest is just a specific parking spot.

This helps not only in better prediction, but also in faster training process and less hardware requirements. But the same way that it can help predicting the correct value, it also could hurt the accuracy of the algorithm if the feature selection is not done properly. The Figure 26 shows two features in a single frame with the same purpose: find if the mold of the injection molding machine is open or closed.

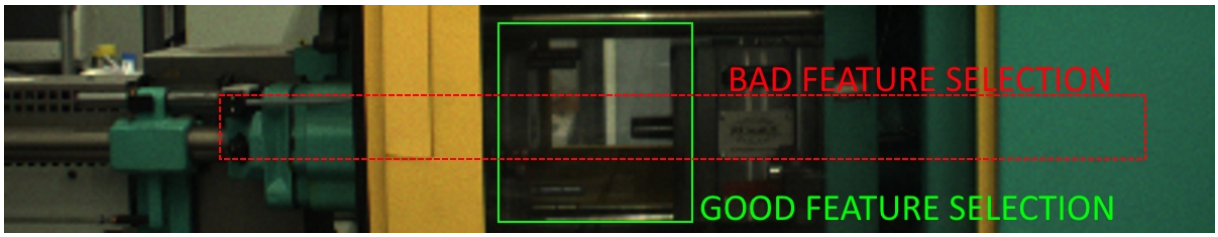


Figure 26 – Feature selection for a frame

Note in Figure 26 that it shows the clamping unit, or the part where the mold is. It is a frame with the mold of the machine open taken from a video that recorded the movement of the mold of the injection molding machine used for this work.

Considering that the task is tell if the mold is open or closed, the red box does not provide good data about the process, while the green box does. This is because there is a lot of useless data in the red box, which should not be considered by the algorithm to make a decision about the machine status. In the meantime, the green box gives just the essential information about the mold, or only the mobile parts of the machine, without any interference from static parts. This way, the algorithm has a much more clear view of the process and can not make mistakes due to insignificant data.

An important thing to mention is that all of the frames must have the exact same feature taken from the full frame in order to make a good model. In order to facilitate the feature selection from the images, the author also developed a software with a GUI (Graphical User Interface) to help making the selection of multiple features in a series of frames. Figure 27 shows the main window from the software.

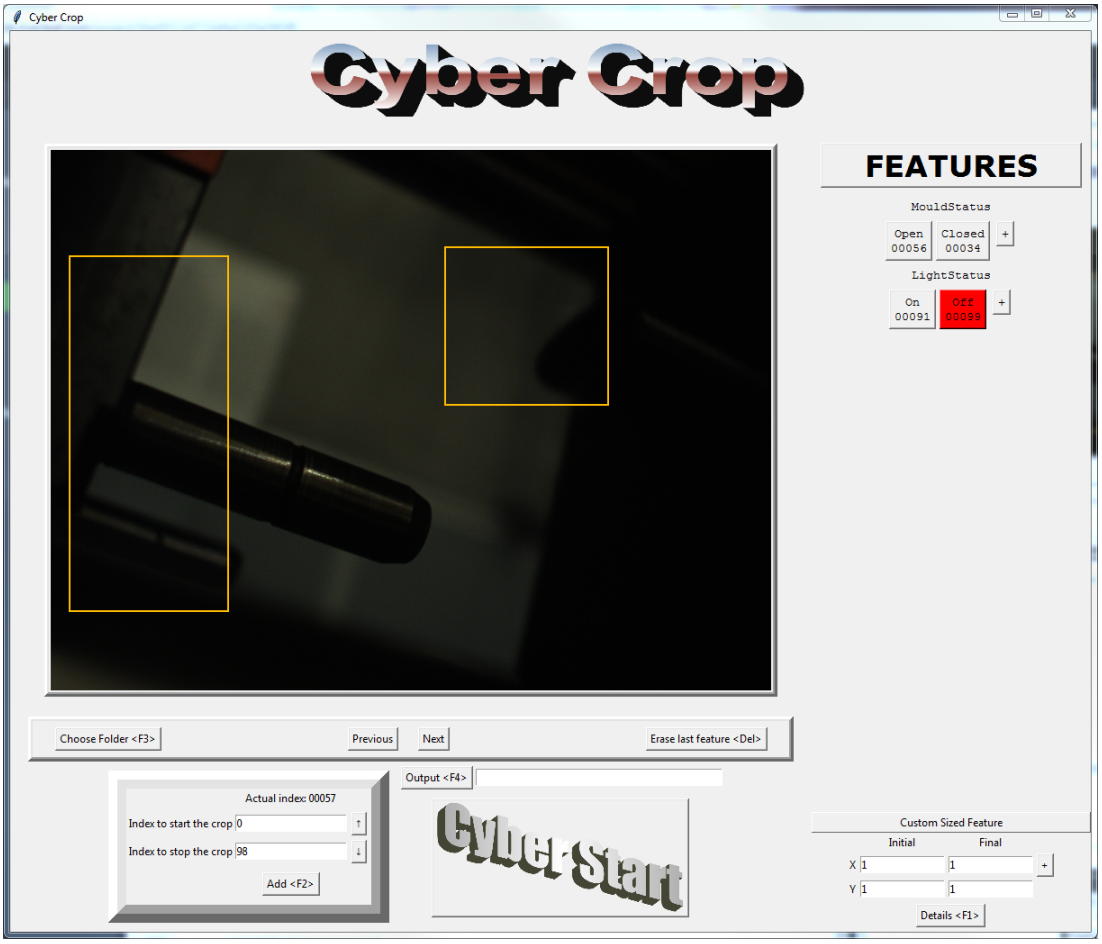


Figure 27 – Software to select feature and crop images

The software's functionality is really simple. The first step is to select a folder with a series of images that need to be labeled according to features and classes. When the folder is selected, the first image of the folder appears on the main window and the user has to draw a box for the first feature of interest in the image. From that point on, it is

just a matter of naming the features and classes needed and assigning the frames to their respective class.

There is also a log of the operations, helping the user to find the exact location and size of the drawn features and the actual indexes of images inside the folder that were assigned to each class or feature. The information about the size and location of the features is very important later when building the machine learning model or trying to predict. A sample log for a session can be seen in the Figure 28.

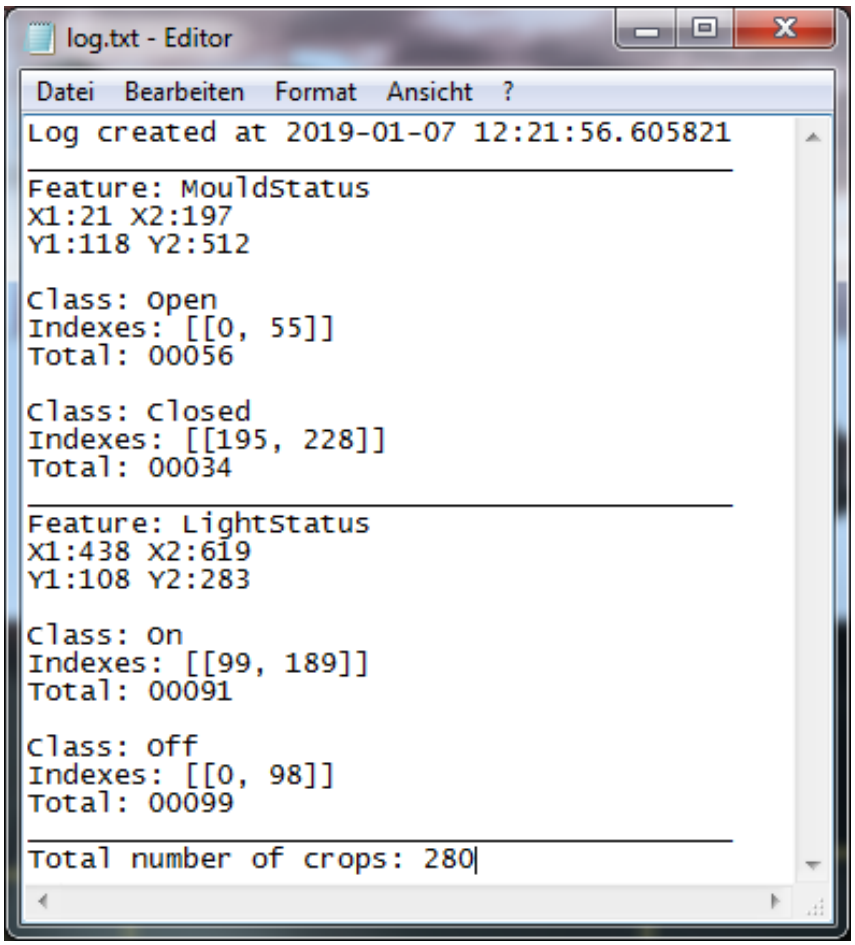


Figure 28 – Log of the feature selection

The output of the software itself is really simple. The images are cropped according to the feature and saved in different folders for each feature and each class inside each feature. The log is also saved in the main directory selected.

No feature selection is made for the vibration data, as there is only one value by axis by measurement. The processing done before feeding the data to the algorithm is filtering the values during acquisition and more details are provided in section 3.1.2.2.

3.3 Data Augmentation

In an industrial environment there are some parameters that change during the day, week, month or even season. These parameters can be the day light, temperature, humidity and many others. In factories, there is also usually a lot of vibration that could result in a change of the precise positioning of a camera for an instant or even permanently.

And it is not possible to map all of this possible change that can happen in an industrial environment with real data. It would take months of data acquisition and also probably would not be enough.

Fortunately, there are methods to compensate these environmental changes to give more robustness to the models. It is possible to generate new data with some changed parameters such as brightness, rotation, offset, blur, cropping. This way it is possible to transform a small without much variation dataset into a much larger dataset composed by different settings of light and positioning.

Data augmentation is one fundamental step before feeding the acquired images to a neural network. It will make the original dataset much more reliable and robust. The parameters used to produce this data augmentation really depend on the process itself. If it is much variable, then the robustness should be higher and therefore the parameters for the data augmentation should be in a way that gives images that are not so similar from each other.

The tool used in this work for data augmentation is ImageDataGenerator provided by Keras [38]. This class is written and is also available for general use in Python. The arguments for this function are essentially the parameters for how the shape of the generated images should be. The parameters used for this work can be seen in the Table 2.

The important parameters to notice are the rotation, shifting, shear, zooming and channel shifting. Those are the main parameters used to give a more robust and reliable dataset, considering the environment that an injection molding machine is usually placed. This gives robustness against small displacements of the camera in the setup and also against changes in daylight or weather.

3.4 Model Structure

It is not always an easy task to determine which model structure will work best for a given dataset. There are algorithms that are known for working best with some specific kinds of data, but it will not always be the best.

Through mathematical methods it is possible to have a better understanding of which model structure works best for each kind of data, but this is not the focus of this work. The methods for defining the best model structure for this project are based on recent researches and iterative methods.

Table 2 – Parameters for data augmentation

PARAMETER	DESCRIPTION
featurewise_center=False	Boolean. Set input mean to 0 over the dataset, feature-wise.
samplewise_center=False	Boolean. Set each sample mean to 0.
featurewise_std_normalization=False	Boolean. Divide inputs by std of the dataset, feature-wise.
samplewise_std_normalization=False	Boolean. Divide each input by its std.
zca_whitening=False	Boolean. Apply ZCA whitening.
zca_epsilon=0	Epsilon for ZCA whitening. Default is 1e-6.
rotation_range=1.	Int. Degree range for random rotations.
width_shift_range=0.02	Float (fraction of total width). Range for random horizontal shifts.
height_shift_range=0.02	Float (fraction of total height). Range for random vertical shifts.
shear_range=0.01	Float. Shear Intensity (Shear angle in counter-clockwise direction as radians)
zoom_range=0.02	Float or [lower, upper]. Range for random zoom. If a float, [lower, upper] = [1-zoom_range, 1+zoom_range]
channel_shift_range=10	Float. Range for random channel shifts.
fill_mode='nearest'	One of {"constant", "nearest", "reflect" or "wrap"}. Points outside the boundaries of the input are filled according to the given mode
cval=0.	Float or Int. When fill_mode = "constant": Value used for points outside the boundaries.
horizontal_flip=False	Boolean. Randomly flip inputs horizontally.
vertical_flip=False	Boolean. Randomly flip inputs vertically.
rescale=1./255	Rescaling factor. Defaults to None. If None or 0, no rescaling is applied, otherwise data is multiplied by the value provided (before applying any other transformation).
preprocessing_function=None	Function that will be implied on each input. The function will run before any other modification on it. The function should take one argument: one image (Numpy tensor with rank 3), and should output a Numpy tensor with the same shape.

3.4.1 Structure for the image classification

The structure used to create a model for trying to predict the correct result from the data of the camera is a CNN. But there are infinite ways of building a CNN, from the most basic to the most complex structures. For this work, it is proposed to use not just one, but several structures for convolutional neural networks, and compare their results and performances.

But first, it is important to take a close look at the steps before feeding the data to the network. As previously discussed, the first step is Data Acquisition [3.1], then Feature Selection [3.2] and finally Data Augmentation [3.3]. But still after these steps, it is important to do some final processing on the data before really start training the model.

The first step is to read the images inside a script. They must be read as arrays instead of images, as shown in 2.2.1. To do so, there are a lot of possibilities. For this work, it was used the Python programming language with the OpenCV [39] library. It is a library with many tools for image processing with great performance.

With this library, it is possible to change the shape of the dataset to reduce the complexity of the algorithm. The first and most important thing for reducing the complexity of the computations is reading the images in grayscale. This helps reduce the number of channels from three to just one, as explained in section 2.2.1. It is only helpful because the images do not rely on color for good differentiation in this dataset.

After reading the images in grayscale, another important step is to normalize them to a square image with low resolution. This helps the algorithm to focus just on main aspects of the images and waste no time observing singularities of individual images, also helping the model to be more robust. For the human eye it may not be the best approach to increase the accuracy on the image classification, but for a computer it is easier to make assumptions and relations if the complexity of the data is low.

Another step for making the data more suitable for a computer to read is normalizing the array of data to 1. In this case, the data is an array of a given dimension with values from 0 to 255. For computational reasons, it is best to set these values in a range from 0 to 1. Then it is possible to finally shuffle the data, if needed. In the case of this work, the image data is shuffled before being fed to the network. This way, the algorithm does not consider periodicity for the prediction. Sometimes it is better not to shuffle the data, but in this case it is shuffled. After this step, the data is ready for the model to train on.

It is also important to state that by doing this, it is extremely important that the images for actual classification also go through this process afterwards, otherwise they will have different shapes and the model will most likely make bad classifications. By doing all those steps, the algorithm is trained to classify normalized grayscale images with a specific square size and will make bad decisions if other type of data is fed to it. There is also the possibility to train the algorithm with pure and raw images, but the accuracy might change.

To have a better reliability and a wider variety of results, the camera was positioned both inside and outside of the chamber, as mentioned. But not only that, different resolutions and regions of interest were also used in order to improve acquisition quality.

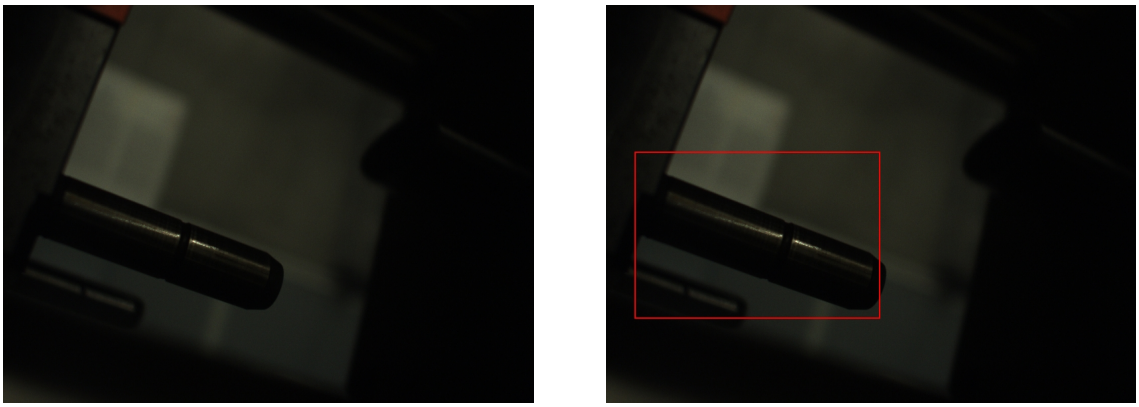
3.4.1.1 Frame with full resolution inside the IMM chamber

This position used for acquiring images is inside the machine's chamber, ideal scenario for best image quality. Positioning the camera this way leaves less susceptible to exterior interference like shadows or something blocking the direct view of the machine. Another important aspect of positioning the camera is the better management of cables, being possible to integrate everything inside the machine.

It is important to notice that suitable lens should be used in this case, otherwise it will be very difficult to focus and get a good image due to the proximity of the camera to the mold. Unfortunately in an industry scenario this setting would not be possible because of the temperature specifications of the camera (See Table 1). For research purposes, there is no problem in installing this camera inside the chamber as long as the temperature does not exceed the maximum specified by the vendor.

For this camera setting it is used full resolution (1600x1200) to capture and analysis of the images inside the software. Despite this fact, the images are saved to the hard disk with half of the maximum resolution, that is 800x600, making the process faster and therefore gaining a more reliable framerate. The final framerate obtained for this camera setting is around 2.45 FPS, value very low but with very high quality for being a smart camera.

After analysis on the images, it is possible to narrow even more the size of the region of interest by doing the feature selection [Section 3.2]. Figure 29 shows the original frame and the selected area actually used for classifying the images marked by a red box.



(a) Camera inside chamber with full resolution

(b) Feature selection for camera inside chamber with full resolution

Figure 29 – Position and feature selection for the inside camera with high resolution

It is possible to see that the selected area inside the frame is very relevant to the process because it shows exactly the position where the moving part of the mold fits the static part of the mold. This makes a frame with the mold closed totally different from a frame with the mold open and therefore easier for the algorithm to classify.

3.4.1.2 Frame with low resolution inside the IMM chamber

Similarly to the frame with full resolution inside the IMM chamber, this frame is located at that exact same position with the only difference being its size. It uses a lower resolution by excluding rows from bottom and top of the frame, resulting in a capture resolution of 1600x600 with 600 rows excluded in total. The images are saved afterwards with a a quarter of that resolution (400x150), providing a much better framerate of 9.07 FPS.

Despite the camera has the option to lower the resolution, it does not provide binning or any method to reduce the image resolution with the same field of view, being possible just to delete only a specified number of rows, without the possibility of ignoring also a given number of columns. This way the field of view becomes limited, but for this project it is not a problem since the region of interest is very small, thus can be selected from a small area on the image.

(a) Camera inside chamber with lower resolution

(b) Feature selection for camera inside chamber with lower resolution

Figure 30 – Position and feature selection for the inside camera with low resolution

Note that if the camera supported the exclusion of columns of pixels, it would be possible to make the region of interest for image acquisition even smaller, being possible to achieve a better framerate. Note also that the region of interest got smaller from Figure 29.b to Figure 30.b. This is intentional, showing that it is possible to choose a smaller area and get the same or better results with less computational power.

For these two camera settings, there are vibration data available for later comparison. The vibration data covers almost the whole period of time as the cameras are recording and is linked by a timestamp of the same source.

3.4.1.3 Frame outside the IMM chamber

If this camera were to be installed in an actual factory environment, it would have to be placed outside of the machine's chamber. To try to replicate such environment, the camera was placed right outside the machine. It was obtained a value of 9.65 FPS with a final capture resolution of 400x75. The Figure 31 shows the actual positioning of the camera while operating outside of the machine.

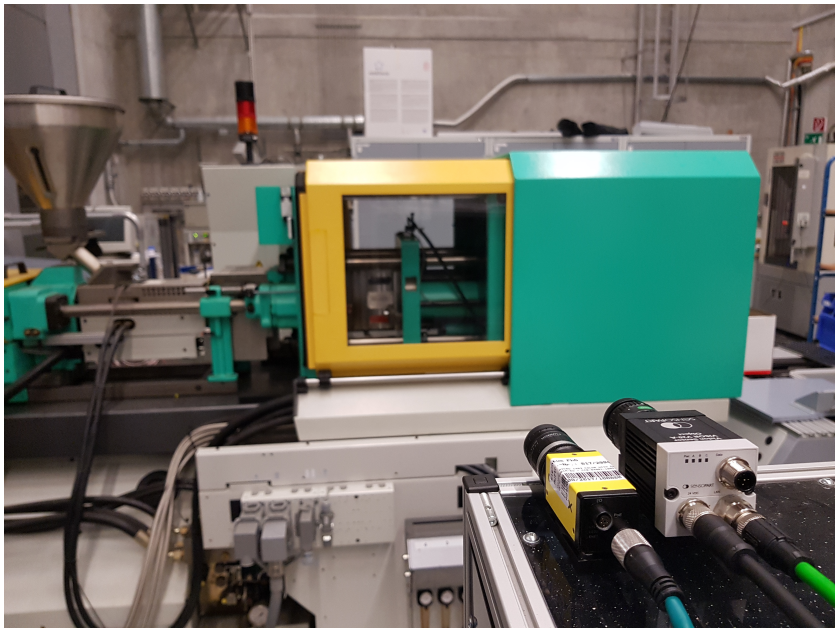


Figure 31 – Positioning of the camera outside the IMM chamber

It is possible to see the Cognex and the Sensopart installed on the setup. The Sensopart camera has proven to be worse for this specific task for some reasons that are not relevant for the topic of this work and was discarded.

The field of view of the Cognex camera in this position can be seen in Figure 32.

Figure 32 – Field of view from the camera outside the machine

The image is cropped so that the mold is barely inside the frame. This is not a problem as long as all of the moving parts stay inside the frame. In this case, this condition is satisfied, so it is possible to define a good feature. The feature selected for this frame can be seen in Figure 33.

The final resolution used in this camera setting is very slower if compared to

Figure 33 – Feature selection for the camera outside the machine

This runs into the same problem as mentioned in section 3.4.1.2, where there is no possibility of excluding columns of the frame, resulting in a great part of the image being filled with useless data.

For this camera setting, the vibration data logger was not put in use. The main objective of this setting is to check if it is possible to detect the status of the machine with the camera being positioned outside of the machine.

3.4.2 Structure for the vibration data classification

Before explaining the structure of the model, it is necessary to understand the dataset itself. After the data is acquired, it must be stored in a dataset. The dataset is really simple and straight forward, with just one table filled with data for each measurement and an output value equivalent to the status of the mold. Table 3 shows part of the dataset and its structure.

Table 3 – Vibration Dataset

real_time	execution_time	x	y	z	Status
19:06:59.019	9398.205332	973	25	-12	1
19:06:59.019	9398.374675	987	36	1	1
19:06:59.019	9398.540719	1002	48	15	1
19:06:59.019	9398.704564	1002	48	15	1
19:06:59.019	9398.881238	1017	55	27	1
19:06:59.019	9399.05278	1028	55	46	1
19:06:59.020	9399.229087	1040	52	35	1
19:06:59.020	9399.394398	1053	52	35	1
19:06:59.020	9399.557876	1053	52	35	1
19:06:59.020	9399.749944	1038	38	18	1
19:06:59.020	9399.990763	1027	24	5	1
19:06:59.020	9400.182099	1007	31	-10	1

As seen in Table 3 there are two columns for timestamps, three columns for actual data and a final column for the status of the machine. The status is defined by 0 if the mold is open or 1 if the mold is closed. With the data in this structure, it is possible to move forward with building the model for classifying the status of the injection molding machine.

The basic structure used for classification of the vibration data is a RNN using LSTM. This type of algorithm is well suited for time-series data, as explained in section 2.2.5 and might be a good approach to be used on this dataset.

But for being a time-series, it is not straight forward to input this data into the network. It must first be separated in blocks, because the data is not evaluated by single measures, but in a context. After defining the size of the block, it must also able to slide through all of the data by a given step. If the step size is bigger or the same as the size

of the block, none of the data of one block will belong to another block and if the step is very small, there will be very few variation and a lot of repeated data between each neighbor block. So, choosing the correct block size and step is a difficult and important decision. Both parameters depend mostly on the frequency and shape of the data and is optimized by trial.

This procedure reduces drastically the number of outputs by a factor of the step size. Now instead of analyzing each row of the table, the algorithm has to analyze the number of rows equivalent to the block size at each iteration. The next iteration has to be the same block size but with an offset equivalent of the step size. Usually the step size should be approximately 10% to 20% of the block size, so it is not too small nor too big. After analysis on the data, the defined block size for this dataset is 100 samples with a step of 15 samples per iteration.

3.5 Model Structure and Training

As already explained in section 2.2, the training process requires a lot of computational power and to make this easier GPUs are often used, reducing the time of processing. For the efforts of this work it is used a Tesla V100-PCIE-16GB located in a server. This is a powerful hardware for machine learning and reduces drastically the time of processing if compared to the use of an ordinary CPU. For some experiments it is also used a NVIDIA Quadro K4100M, a smaller GPU embedded in a laptop that also has compatibility to TensorFlow [26]. All of the hardware is provided by Fraunhofer IPA.

In order to create neural network model, the first step is to define its parameters. The most essential parameters are the number of epochs, the validation split, the batch size, the size or number of nodes for each layer, the type of layers required and finally how many layers of each kind.

The number of epochs is the amount of times for the network to be trained repeatedly, that is repeating the training several times with different starting weights and in some cases with different parameters. Validation split is literally taking part of the data out of the training dataset just to confirm the accuracy of the model after each epoch. This way, it is sometimes possible to detect if the model is overfitting or just remembering the exact values from the training dataset and not being able to identify unseen data. The batch size is just dividing the dataset so there is no need for the network to train the entire dataset at once. This makes it easier for the network to read and improve the accuracy. The type of layers is what makes the type of the algorithm be a convolutional, recurrent, or any other kind of neural network. Finally, the size of each layer and how many layers of each kind implies on the size and complexity of the network. A network with just one fully-connected layer is called a neural network while a network that has more than one layer is called a deep neural network. In this work, only deep neural networks are used.

All of the algorithm was built using Python and its supported libraries with TensorFlow and Keras frameworks for building neural networks [Section 2.3]. The main reasons for using TensorFlow and Keras are for being open-source with a lot of official documentation, well developed frameworks, user friendly and easy to use, integration between the two and also support of GPUs and most importantly, having good results.

3.5.1 Convolutional Neural Network Parameters

After all of the procedure taken into account in section 3.4.1 it is possible to move forward and actually start building the model and training the network.

The output of the algorithm used in this project gives 18 different models to each single dataset. This way, it is possible to compare between different configurations and select the best suited for each dataset. During training, the network sends information to TensorBoard [Section 2.3.1] which displays the current status of the network with real-time graphs and plots. After each model is completely trained, the parameters are changed and the process starts again with TensorBoard logging in a different file. This way, it is possible to compare the different models and their performances and finally export the information to a table.

All of the different parameters used while training the convolutional neural network used for classifying the images are displayed in Table 4.

Table 4 – Parameters used during CNN training

Parameter	**Value**
Epochs	20
Validation Split	50%
Batch Size	8
Layer Size	128, 256, 512
Layer Types	Convolutional, Max Pooling, Dense, Dropout
Dropout	20%
Number of Layers	2, 3, 4, 5, 6

A fixed number of epochs, validation split, batch size and dropout was set prior to training and the variables are the number of convolutional and dense layers with their respective variable sizes.

The number of dense layers varies between zero and two, the number of convolutional layers varies between two and three and the layer sizes can assume the values of 128, 256 and 512. In total, there are three different settings for dense layers, two for convolutional layers and three for the layer sizes. Multiplying all of them, the result is eighteen models for each filmstrip, as mentioned. If taken into account that there are three different filmstrips, the total number of trained models is 54 and all of them are logged in TensorBoard with full statistics.

The validation split is set to 50% to give robustness to the model, since there is no separate testing data to validate the results. That means that the model will train with only half of the images and the other half is used for measuring the quality of the model. This helps to detect overfitting and to avoid it when it happens. The images are also not related with each other, so the model does not care about the result of the previous image to make a prediction.

Cross-validation is a good technique where the dataset is divided by smaller subsets and the model is trained every time with all datasets except one. This gives much more information about the quality of the model trained, but is very computational expensive, so it was not used for measuring the accuracy of the models in this work. Instead, it was used a much higher split of randomized data to give good credibility to the models.

3.5.2 Recurrent Neural Network Parameters

The algorithm used for training the recurrent neural network is similar to the one used for the convolutional neural network. The major changes are how the data is extracted and the shape of the model itself.

Differently than what occurs during CNN training, the shape of the model during the training of the recurrent neural networks is fixed in this project, changing only how the data is fed to the algorithm. The fixed parameters can be seen in Table 5 and these values are obtained by experimentation.

Table 5 – Parameters used during RNN training

Parameter	**Value**
Epochs	10
Validation Split	40%
Batch Size	64
Layer Size	64
Layer Types	LSTM, Dropout, Dense
Dropout	30%
Number of Layers	2

The variation of the models comes from delivering the dataset in different shapes into the network. As explained in section 3.1.2.1, it was used one accelerometer inside of the IMM and one outside of the IMM. Therefore, for each filmstrip there are two vibration measurements with each one having five other variations. Summing all of the datasets, considering there are two vibration sensors and the data was acquired in two different periods of time, there are exactly 24 datasets to be trained and analyzed. To better understand this concept, Table 6 shows how the data is divided between each category.

On each of the original datasets from both sensors, two filters were used, being one low-pass [Section 2.4.1] and one high-pass filter [Section 2.4.2]. The low-pass is meant

Table 6 – Vibration datasets structure

Location	**Filter**	**Dataset**
Inside of the IMM	Filterless	Reduced
		Entire
	High-Pass	Reduced
		Entire
	Low-Pass	Reduced
		Entire
Outside of the IMM	Filterless	Reduced
		Entire
	High-Pass	Reduced
		Entire
	Low-Pass	Reduced
		Entire

mostly to extract good data from the accelerometer from inside the machine, filtering out the high frequency vibration and keeping just the acceleration of the mold. On the other hand, the function of the high-pass filter is to filter out any low frequency acceleration that might happen on the outside and mostly on the inside of the machine, leaving just the high frequency vibration data. Results can be found in the chapter 4.

4 Results

In this chapter all of the results from the experiments held during the paper are displayed and explained. The analysis with TensorBoard is made using four different metrics: accuracy, loss, validation accuracy and validation loss. For each model trained, all four metrics are being measured after each epoch. This results in learning curves of the algorithm through time, making possible the evaluation of the system behavior throughout the training process.

Training accuracy is the accuracy of the model through each batch of data. It updates constantly after each batch by predicting the trained values with the actual model and getting a percentage of correct values. Loss is calculated by a function inside Keras. There are many loss functions and the one used for this work is categorical cross entropy. This loss function penalizes more predictions that are very different from the expected value than values that are close to the expected result. The formula for categorical cross entropy is given by:

$$L_{CC}(\hat{y}, y) = -\sum y_i \log(\hat{y}_i) \tag{4.1}$$

Where:

$L_{CC} =$ categorical cross entropy loss

$\hat{y} =$ classifier output

$y =$ distribution over labels

$i =$ iteration

The validation accuracy is given after every epoch, using the actual state of the model to predict the validation dataset. The percentage of correct classifications results in the validation accuracy. The validation loss is exactly the same thing as the training loss, with the exception that is calculated by validation data.

4.1 Camera Results

The results of training the images with different models are displayed in this section. As already mentioned in 3.5.1, the output of training the data of each single camera setting is 18 different models. To better visualization of the results, they are displayed in three plots of six curves for each metric of each camera setting. The three different plots are sorted by their final value. That means that the first plot will have the six

worst final values, the second will have the six intermediary final values and the third will have the best final values. Note that sometimes the scale of the graphs changes for better visualization.

The analysis of each camera is made separately and major points will be discussed as the results are shown. For better understanding of the results, Table 7 makes a relation between the indexes used on the names of the models and the actual description of each camera.

Table 7 – Reference table for model names

Camera Label	**Camera Description**
cognex_1	Camera placed inside the chamber with high resolution
cognex_2	Camera placed outside the chamber
cognex_3	Camera placed inside the chamber with high resolution

The model names follow a strict pattern. For each one, there is always the following structure:

Camera Label_**NUMCONV***conv* – **NUMNODES***nodes* – **NUMDENSE***dense* – *shuffle***SHUFFLEOPTION** – *val_split***NUMSPLIT** – **TIMESTAMP**

Where:

- **Camera Label** is given accordingly to Table 7
- **NUMCONV** is the number of convolutional layers of the model
- **NUMNODES** is the size of each layer in the network
- **NUMDENSE** is the number of dense layers excluding the last one
- **SHUFFLEOPTION** is a boolean which enables or disables data shuffle
- **NUMSPLIT** is the percentage of the dataset that is used for testing

Therefore, if a model name is:

cognex_1_3conv-256nodes-1dense-shuffleTrue-val_split0.3-1546948531

then it is a model trained with data from the camera placed inside the chamber with high resolution, with three convolutional layers, one dense layer and 256 nodes for each layer, where the data is shuffled before going to the algorithm, with a 30% validation split of the data for testing and the time that the model was created is represented by the Posix [40] timestamp 1546948531 that could be converted to Tuesday, 8. January 2019 11:55:31.

4.1.1 Frame with full resolution inside the IMM chamber

In this setting there is the slowest speed of image acquisition, therefore it has the lowest framerate. This results in a big difference between each image if there is movement, as the time between frames is relatively long. This makes the dataset have few data while moving, meaning it could possibly make it harder for the algorithm to train in transition states if the dataset is not big enough and has not good variation. In some cases it could mean that the accuracy of the algorithm stays the same, just being slower to react.

In Figure 34 it is possible to see the accuracy results with the training data while the model is being trained.

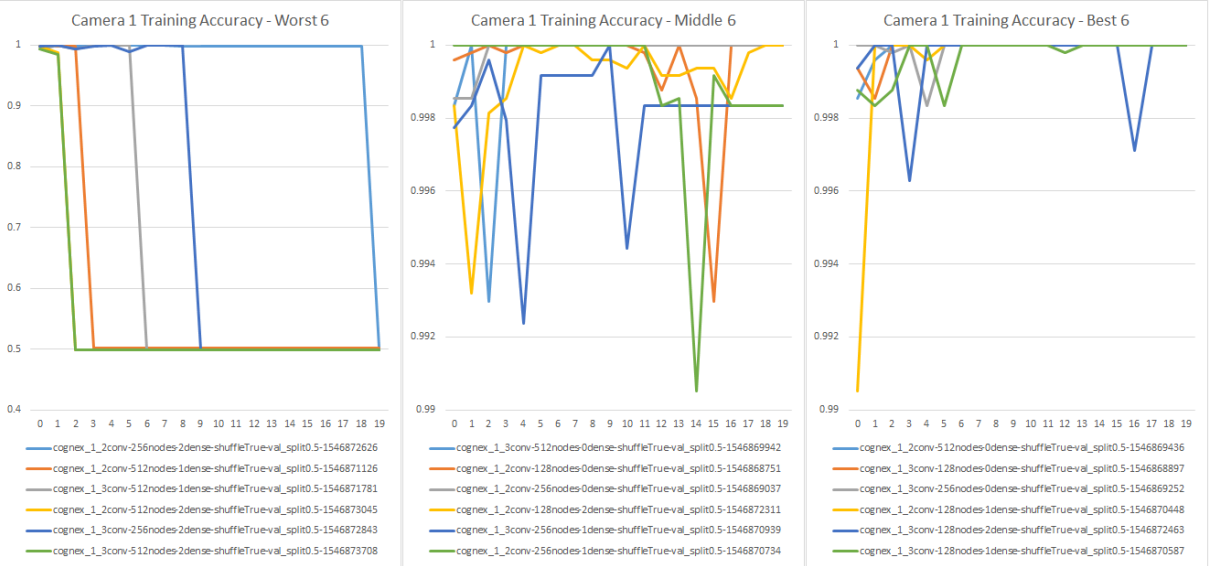


Figure 34 – Training accuracy for the *cognex_1* models

From Figure 34 it is possible to see the performance difference between the worst and best models. The worst six models could not complete the training process with success, descending to around 50% accuracy after some epochs. This is due to too much complexity in model structures which leads to weight values getting too small and therefore resulting in NaN values leading to sudden loss of accuracy. It is possible to see by Figure 34 that the first model to decrease the accuracy to 50% is the most complex one, having 3 convolutional layers, 512 nodes for each layer and 2 dense layers. The other models that descend their accuracy values are also either not trivial or have some complex parameters. This complexity leads the model to compute very small gradient values, which after some time are not able to be accurately computed anymore. One possible solution is to set a minimum value for the gradient in order not to let it be that small. The specific value of 50% happens because there are just two classes and exactly half of the data belongs to each class, thus the algorithm outputs only one class and gets half of the values right. Figure 35 continues to exemplify this phenomenon with loss values.

As for the other two graphs in Figure 34, it is possible to see a good result in general for almost every model. Ten models get 100% training accuracy at the last epoch, which is a great value but could also mean that there was overfitting during training. It is possible to check if this is the case by analysing the graphs on figures 36 and 37. It is also noticeable that all models that used 128 nodes for each layer got 100% accuracy, meaning that for this dataset it would have been enough to have low complexity on the models. This possibly would not be the case with different datasets with more classes to classify, where more power and complexity are required.

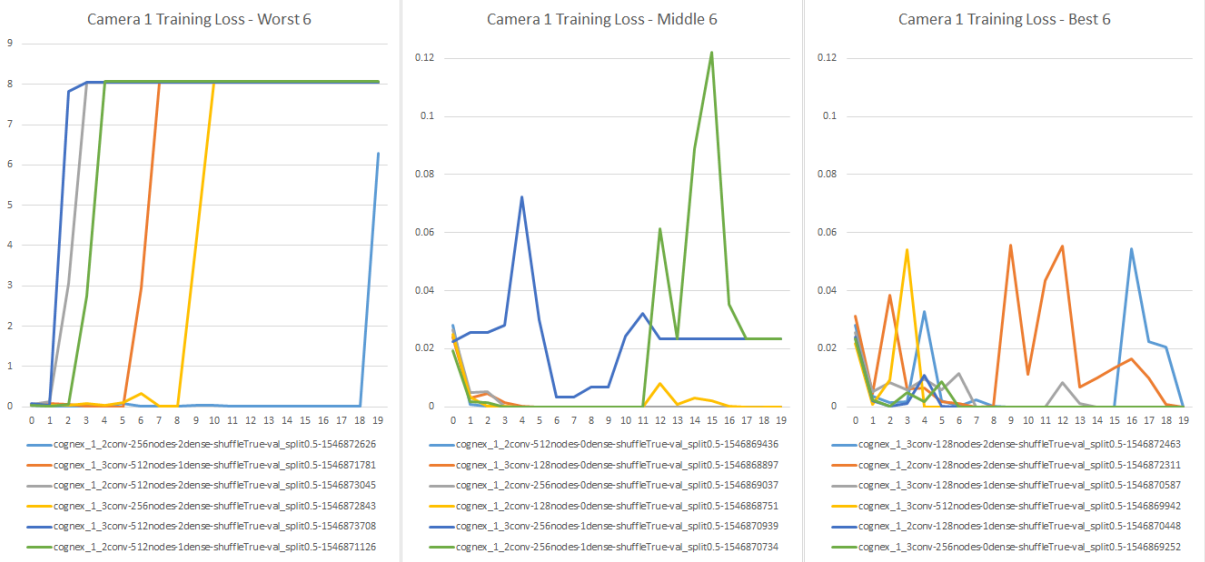


Figure 35 – Training loss for the *cognex_1* models

Similarly to the accuracy results, the loss also tends to follow the same pattern for the same models, but instead of the value falling it grows to some absurd value.

The best results for training loss are displayed in Table 8 with the respective name of the model, following the standard adopted previously.

Table 8 – Best results for *cognex_1* training loss

Model Name	**Result**
cognex_1_3conv-128nodes-2dense-shuffleTrue-val_split0.5-1546872463	1.19E-07
cognex_1_2conv-128nodes-2dense-shuffleTrue-val_split0.5-1546872311	1.19E-07
cognex_1_3conv-128nodes-1dense-shuffleTrue-val_split0.5-1546870587	1.19E-07

From it it is possible to see that the three best model for training loss have the same 128 layer size. This does not mean yet that the best model is indeed one with this layer size because this is just based on previously fed training data. To get a conclusion for which is the best model in this case, it is necessary to analyse how the algorithm behaves with unseen data. For that, Figures 36 and 37 display in the same format the results with testing data for both accuracy and loss metrics.

Figure 36 – Validation accuracy for the *cognex_1* models

As already mentioned, the validation results are obtained by feeding unseen data to the model and watching its behavior. If the training accuracy is very high but the validation accuracy is very low, it is most likely that there was overfitting to the model while training.

In this case, this appears not to be the case but it is important not to jump to conclusions. Even if the accuracy of the training data and the testing data are high, if the dataset does not have enough variety, it could happen that the overfitted algorithm is also able to be accurate with unseen data as long as it is very similar to the training data. For that, it is very important to have a varied dataset with also augmented data [Section 3.3].

With the validation accuracy results from Figure 36 it is possible to observe a very similar behavior to the training accuracy on Figure 34. The same ten algorithms that got perfect score for training accuracy also got perfect score for the testing dataset and also the same six models dropped severely the accuracy, as expected by what happened while training the model. Figure 37 displays the loss results for the same testing dataset.

It is not strange that the loss results follow again the accuracy trends. For better visualization, Table 9 displays the best three results for testing loss.

Table 9 – Best results for *cognex_1* validation loss

Model Name	**Result**
cognex_1_2conv-128nodes-2dense-shuffleTrue-val_split0.5-1546872311	1.19E-07
cognex_1_3conv-128nodes-1dense-shuffleTrue-val_split0.5-1546870587	1.19E-07
cognex_1_3conv-512nodes-0dense-shuffleTrue-val_split0.5-1546869942	1.59E-07

When comparing Tables 8 and 9, two models remain leading the loss but there is the appearance of a different model. It is a model with three convolutional layers with 512

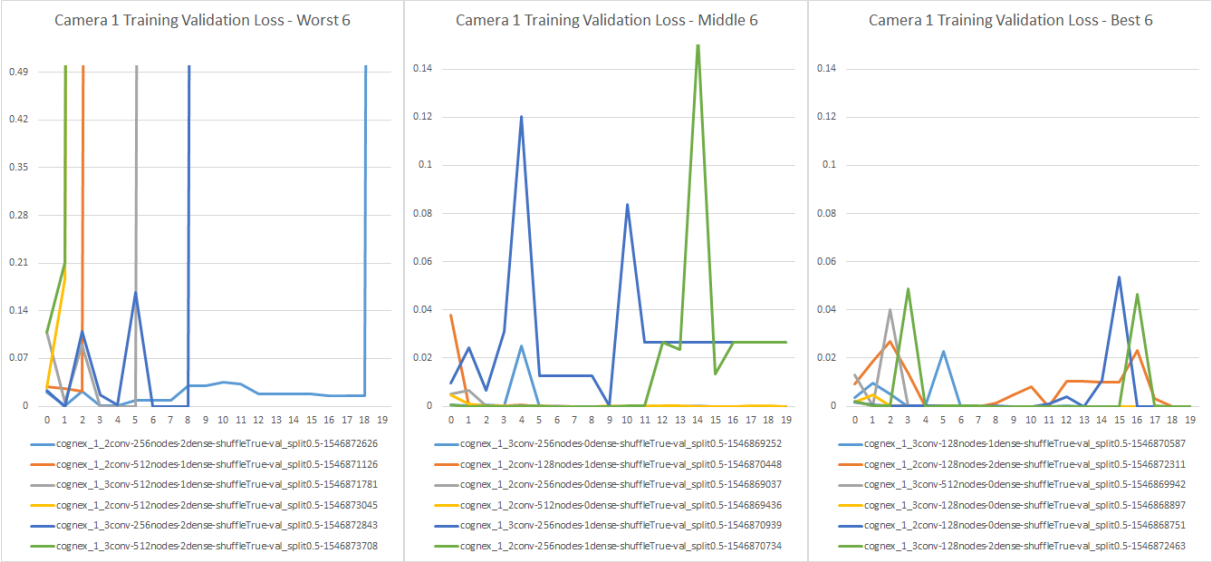


Figure 37 – Validation loss for the *cognex_1* models

nodes for each layer and zero additional dense layers. It is possible to see by Figure 35 that this algorithm also got excellent result during training, so it is not a suspicious result. As for the model that was removed from the best models list, it got worse results in testing (8.10E-07), but also not even close to being a bad result.

For this dataset, it is possible to say that any of the best twelve models achieved very good results, specially the ten best that got a perfect score at the end of training. It is safe to say that any of them would have a great performance if put to a real plant for detecting the movement of an injection molding machine mold.

4.1.2 Frame with low resolution inside the IMM chamber

This is the exact same camera position as the setting in section 4.1.1 with only a lower resolution and a smaller feature selection. The main objective of this setting is to see if there is any difference between the two approaches. This is a better setting in terms of camera performance, with a higher framerate and lower memory allocation. This allows to make faster predictions and this factor can be crucial in some systems.

The analysis for training accuracy of the algorithm for this camera setting is displayed in Figure 38, where it is possible to see the model development with time.

Note that same as in Figure 34, there are exactly six models that run into the weighting problem. Although there is one different model on this plot for each one of the models.

While training the *cognex_1* models, the one with 3 convolutional layers, 0 additional dense layers and layer size of 256 has the third best training accuracy, but then after the second epoch the same model runs into early accuracy problems, even before the most complex model. This happens because despite not being the most complex model, it still is very complex for the task and therefore could run into problems like this depending on

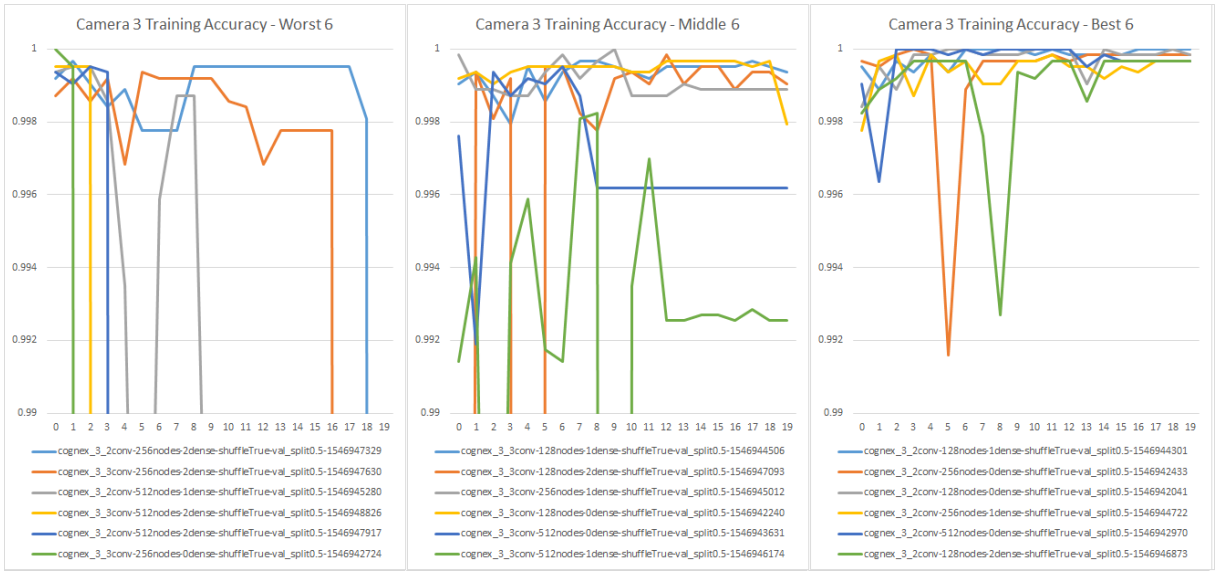


Figure 38 – Training accuracy for the *cognex_3* models

the training data. This was not a problem on the *cognex_1* because differences in dataset did not lead to weights receiving very small values.

At the same time, despite the model with 3 convolutional layers, 1 additional dense layer and layer size of 512 had a stable response while training on the *cognex_3* dataset, it had major problems when the training dataset respective to the *cognex_1* dataset. This is also really depending on the dataset and there is not a solution besides trying to set a limit to the gradient during training, changing the model structure or changing the dataset itself.

Following the same path as before, it is possible to analyse loss by Figure 39.

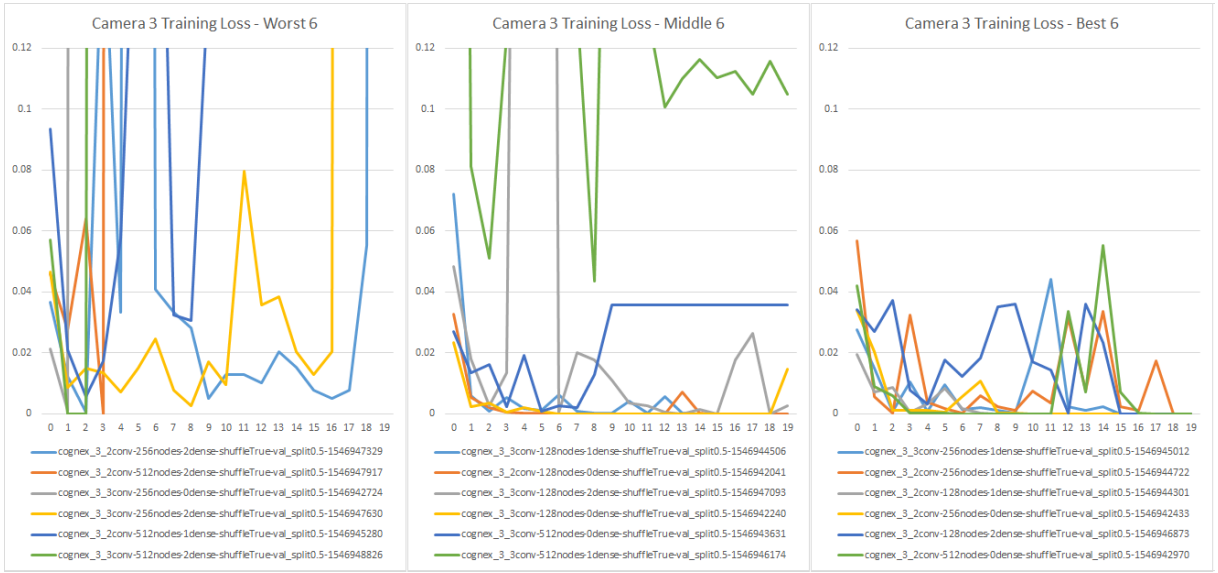


Figure 39 – Training loss for the *cognex_3* models

Once again, the correlation with accuracy training is present on this plot. It is possible

to notice the oscillating response shape of the model that did not complete successfully its training on the *cognex_1*, with a late stabilization around one value. This happens for the same reason that some models can not finish training successfully. Multiplication and division with very low values result in unstable responses, despite this model being able to stabilize after some oscillation. Table 10 quantifies the best three models according to their respective final losses.

Table 10 – Best results for *cognex_3* validation loss

Model Name	**Result**
cognex_3_3conv-256nodes-1dense-shuffleTrue-val_split0.5-1546945012	1.19E-07
cognex_3_2conv-256nodes-1dense-shuffleTrue-val_split0.5-1546944722	1.26E-07
cognex_3_2conv-128nodes-1dense-shuffleTrue-val_split0.5-1546944301	1.73E-07

Despite the model with 2 convolutional layers, 1 additional dense layer and 128 nodes being the best in accuracy, it gets just the third best value for loss. Because of this it is important to always analyse more than just one metric, and see which one or which combination is the best suited for a specific task.

For the really valuable information, Figure 40 and 41 display the results respective to the validation dataset.

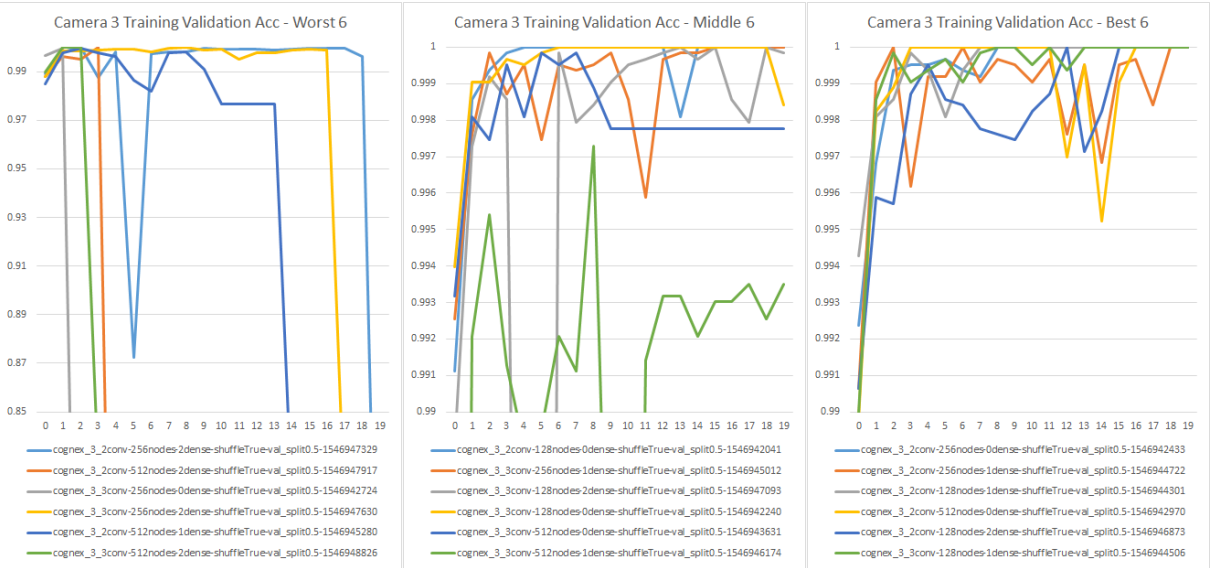


Figure 40 – Validation accuracy for the *cognex_3* models

Surprisingly, the validation results for the best models are better than the training results. This may be caused by differentiated datasets for each operation. For example, the training dataset could have more variation, causing the algorithm to get confused with a specific kind of data that was not present in the validation dataset. For this reason, it it essential to have good variety of data on the training dataset for better robustness and also good data variety for the testing dataset in order to detect overfitting and obtain a final result considering all of the possible data.

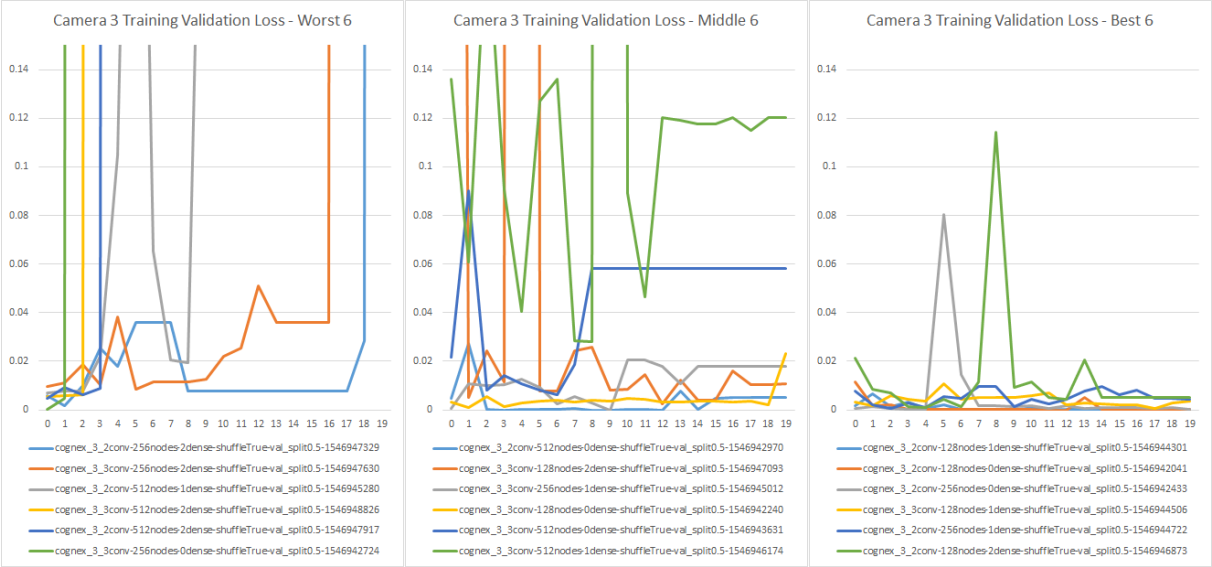


Figure 41 – Validation loss for the *cognex_3* models

On the plot for the validation loss it is possible to see a similar behavior to the one found in the training loss plot with the main difference being the magnitude of both. To enable a better comparison, Table 11 provides the numerical result for the best three models, so it is possible to compare with the results from Table 10.

Table 11 – Best results for *cognex_3* validation loss

Model Name	**Result**
cognex_3_2conv-128nodes-1dense-shuffleTrue-val_split0.5-1546944301	5.58E-05
cognex_3_2conv-128nodes-0dense-shuffleTrue-val_split0.5-1546942041	0.00025241
cognex_3_2conv-256nodes-0dense-shuffleTrue-val_split0.5-1546942433	0.00032122

The only model that is present in both tables is the model with 2 convolutional layers, 1 additional dense layer with the layer size of 128. Again one of the simplest algorithms tested getting great results in comparison to others more complex.

With the results from both the *cognex_1* and *cognex_3* dataset models, it is possible to compare them and analyse that the higher resolution with a bigger feature makes little to no difference while analysing the state of the mold. Both of the models created with their respective datasets perform very well to the imposed conditions and can detect easily if the mold is open or closed. Now the objective is trying to detect the same parameter from outside of the machine [Section 4.1.3] and measure the performance for the vibration data models [Section 4.2].

4.1.3 Frame outside the IMM chamber

Finally, the last camera setting is the shot from outside of the machine, simulating the usage of this specific camera on an industrial environment. There are limitations to this

such as space and variables that are impossible to consider with a relatively small dataset. These limitations, despite being very briefly measured, may reflect on the results.

The same setting from the other two settings for model creation was applied to this dataset. It also consists in 18 models and each model is measured the same way. Figure 42 starts showing the results for training accuracy.

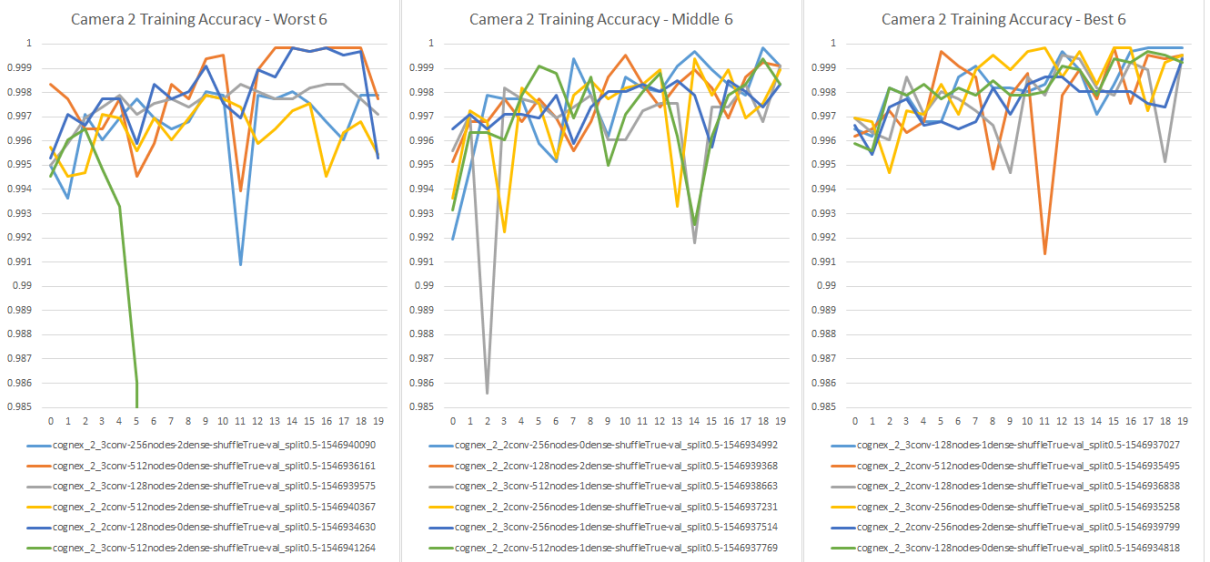


Figure 42 – Training accuracy for the *cognex_2* models

Opposite to what it is seen on the response from the other settings, there is just one model that can not finish the training properly, and is exactly once again the most complex model. With its structure, the model has shown unable to train successfully in all three settings.

As for the rest of the model settings, there was a considerable overall improvement over the other two settings. This is most likely due to wider variation of the dataset, providing more ways for the algorithm to learn. Besides the model that could not complete the training process, all models got over 99.5% of training accuracy. Figure 43 displays all training loss results, that can be observed and compared to models from other cameras datasets.

The loss also maintain the accuracy trend, with the worst final value of 0.033657663 besides the failed model, not bad for being the worst among 17 different settings. Table 12 displays the best three models according to training loss.

Table 12 – Best results for *cognex_2* loss

Model Name	**Result**
cognex_2_2conv-512nodes-0dense-shuffleTrue-val_split0.5-1546935495	3.31E-05
cognex_2_3conv-128nodes-1dense-shuffleTrue-val_split0.5-1546937027	7.61E-05
cognex_2_3conv-128nodes-0dense-shuffleTrue-val_split0.5-1546934818	0.000114837

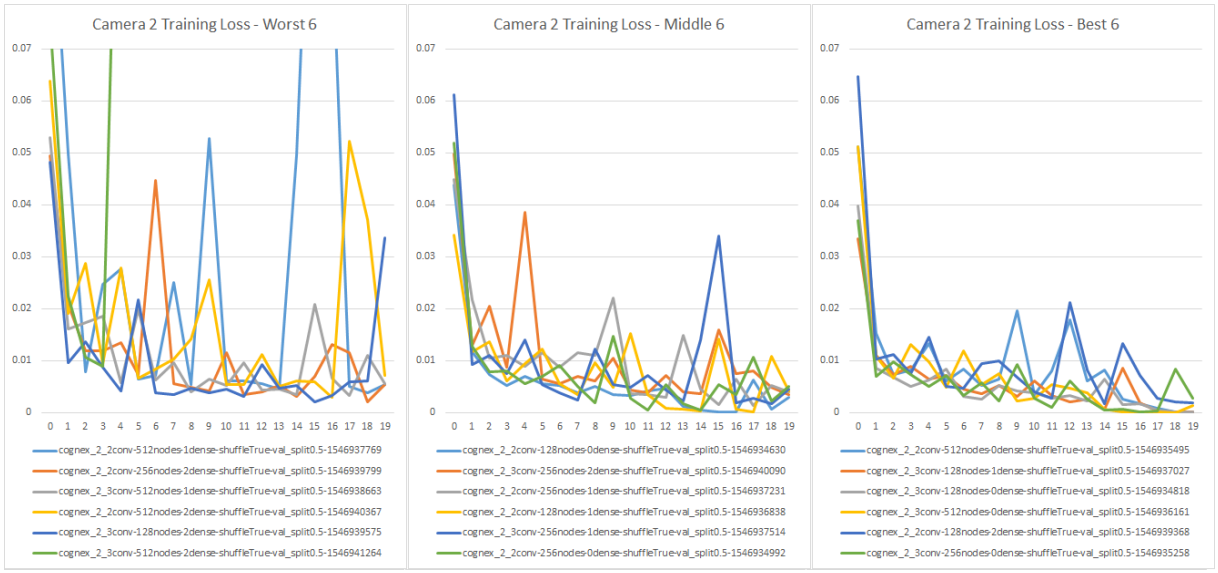


Figure 43 – Training loss for the *cognex_2* models

Despite the fact that just one model got an error during training, the best loss values continue coming from models that have not much complexity. Two of the three best loss values are from models that have only 128 nodes and the other does not have any additional dense layer. This also means that this dataset possibly does not need a very complex model in order to classify accurately the data. But it is important first to analyse the validation values with Figures 44 and 45 before taking any conclusion.

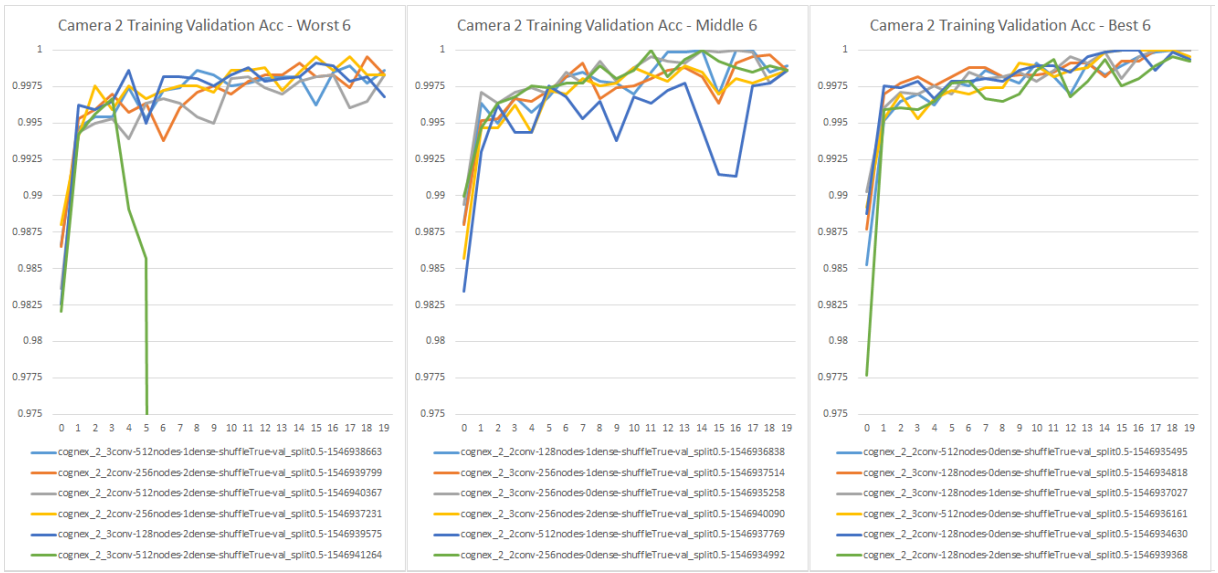


Figure 44 – Validation accuracy for the *cognex_2* models

For the validation accuracy results with this camera dataset, the worst validation accuracy value is around 99.6%, value even better than the ones obtained on training results. This could be because the testing dataset is more homogenic and therefore has only few variation that the model is not able to detect accurately. Normally, the expected

result is to have a worse result in the validation results than the training results, but the opposite can happen in small datasets like this, not meaning that the model is inaccurate, but just that the testing dataset has less variation than the training dataset. Figure 45 brings more information about the loss on the validation dataset.

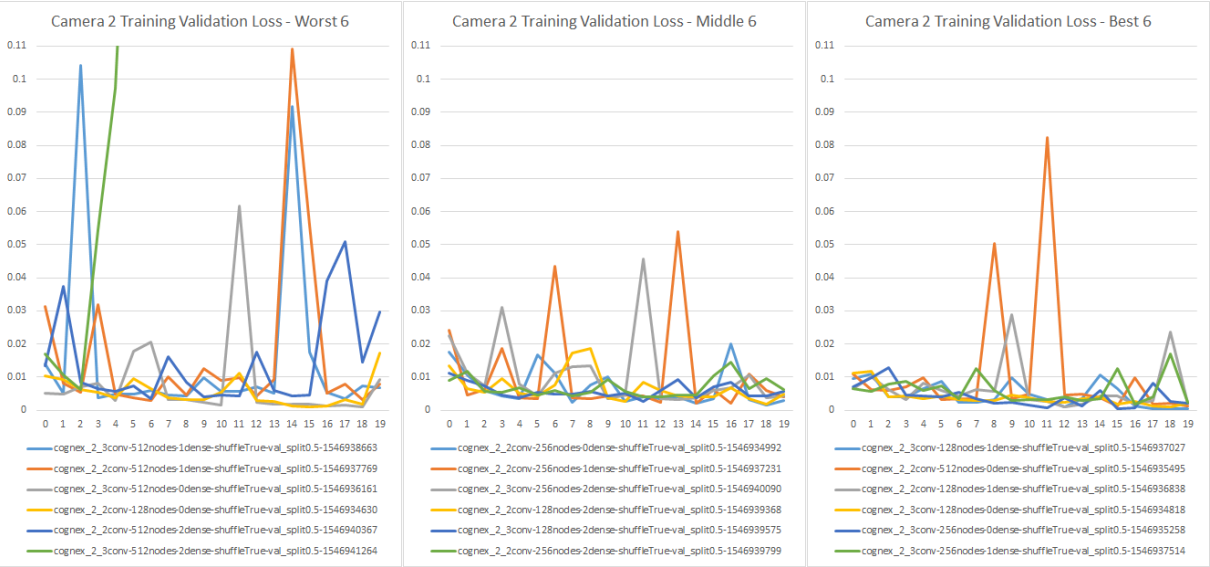


Figure 45 – Validation loss for the *cognex_2* models

Note that some spikes are visible during training. This happens after an epoch because the model gave high weight values to features that might not generalize the whole dataset. It is also noticeable that it corrects right away because of the high loss increase, since the model changes its weights by the loss value. The best results can be found on Table 13.

Table 13 – Best results for *cognex_2* validation loss

Model Name	**Result**
cognex_2_3conv-128nodes-1dense-shuffleTrue-val_split0.5-1546937027	0.00036074
cognex_2_2conv-512nodes-0dense-shuffleTrue-val_split0.5-1546935495	0.001363855
cognex_2_2conv-128nodes-1dense-shuffleTrue-val_split0.5-1546936838	0.001976611

Two of three models appear both on Table 39 and Table 41, meaning that probably one of these two is the best. As expected, the validation loss is a little worse than the results obtained in training, despite being still very small.

The best model in this specific setting is the one with 3 convolutional layers, 1 additional dense layer and the layer size of 128 nodes. It gets 100% accuracy for the testing dataset with the lowest loss, meaning it can handle unseen data with ease. It also has the best training accuracy with a very low training loss. These last two parameters are not as important as the validation results, but if they are low there is certainly something wrong with the model. This is the model used for generating the final output.

4.1.4 Final output

After the model is trained completely, it is possible to use it to predict arbitrary data, in this case frames took from a video. The output of the classification is a number between 0 and 1, that is a value that corresponds the confidence the algorithm has for the data belonging to one class or another. If the number is closer to 1, the final output is then rounded up to 1 and rounded to 0 if the output is closer to 0.

If there is a necessity to predict a whole video, this video should first be separated into images and the data from these images stored in an array, so the algorithm can iterate over the values from the array. After each array of data that is equivalent to one frame, the algorithm sends the output value and this value must be linked to its respective frame. With the linked data it is then possible to annotate the frames according to the belonging class.

To do this, it is used the same OpenCV library for Python that is used for pre-processing the images. The library allows also to draw boxes and write on images. This way it is possible to make a procedure that draws a box in the selected feature position with a given color correspondent to its classification together with predicted class name near the box. The final output becomes then a whole new video with a box drawn on the selected feature that changes its color and the name every time a different prediction is made.

4.2 Vibration Results

The vibration data was collected during the acquisitions from the two camera settings inside the machine's chamber. As mentioned in section 3.4.1.3, the vibration sensor was not active during the shot from the camera outside of the machine.

The model names also follow the pattern

Filter_**Camera Label**_*vibrationdata*_**Location**

for better identification, where:

- **Filter** is given accordingly to Table 14
- **Camera Label** is given accordingly to Table 7
- **Location** refers to the location of the vibration sensor being either inside or outside the machine's chamber

The display of results is similar to Camera Results, with the difference that these results are achieved by changing the shape of the dataset instead of changing the model setting. The model was previously optimized by empirical methods and therefore all datasets will be trained by it.

Table 14 – Reference table for naming models acquired with vibration data

Filter	Description
LABELED	No filter applied
CHOPPEDLABELED	Exclusion of beginning and ending values
HIGHPASS_2order_150cutoff	Butterworth High-Pass filter with 150Hz of cutoff
CHOPPEDHIGHPASS_2order_150cutoff	High-Pass filter with exclusion of beginning and ending values
LOWPASS_1order_15cutoff	Butterworth Low-Pass filter with 15Hz of cutoff
CHOPPEDLOWPASS_1order_15cutoff	Low-Pass filter with exclusion of beginning and ending values

The results in this case are also divided in three different plots with the same approach for dividing them. There is a total of 24 models for all vibration data, so there are eight different curves for each plot. It may be a little difficult to extract the exact values from the plots, but the important thing is to notice the general shape of each curve, as relevant results are going to be further approached.

The testing is made the same way as in the Camera Results section, with the addition of one step to the process. The first part is exactly the same method used in the camera datasets, but the second is made by taking the models trained by one dataset and applying on another dataset. For example, the model resulted from training with the *cognex_1* camera data filtered by a high-pass filter, placed inside the machine will be tested by feeding the *cognex_3* data with the same filtering and location to have a more reliable result.

4.2.1 Training

The first metric measured is the training accuracy. This metric is obtained after each epoch, although being updated after every batch of data, from training the dataset and trying to predict values from the same dataset it has trained on. Figure 46 displays three plots with 8 curves respective to one model each.

There are four major characteristics to observe. The first thing to notice is that there are no models that could not complete the training process. This is because the layer used is LSTM [Section 2.2.5.1], which does not have the problem of the vanishing gradient [Section 2.2.5]. Using this kind of layer, the gradient does not get small to the point where mathematical operations result in NaN values, so the result is smoother.

The second point is the smoothness itself. Note that the accuracy does not have sudden changes on its shape. This is not entirely true because the scale of the graph is very different from the ones found in the results from the models with camera datasets. What happens in that case is that the camera models are already so optimized and close

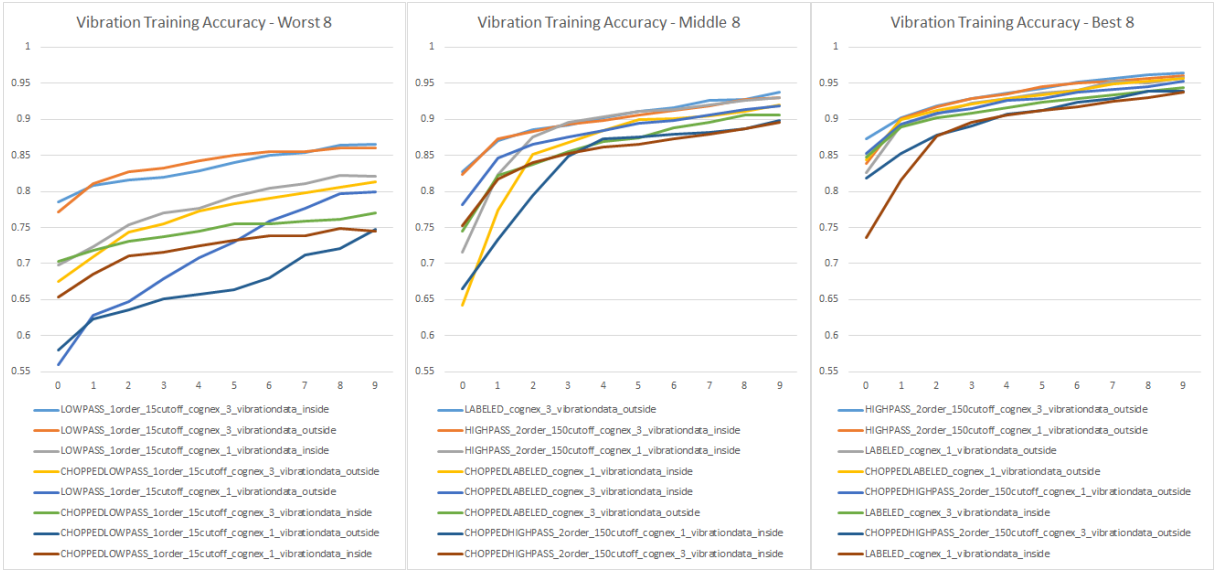


Figure 46 – Training accuracy for the vibration models

to perfection that they can not do much to improve and they keep around an average value. On these models the trend is more constant, with a clear improving movement for most models.

The third important part is the accuracy values in comparison to the camera models. It is noticeable that there is a great gap between the accuracy obtained with the camera datasets and the vibration datasets. While most of the best CNN models for camera datasets have near 100% accuracy, the best accuracy got for the RNN models with vibration data is 96.5% according to Table 15. This is already expected due to the shape of the data and the difficulty of classifying the state of a machine just by its vibration. It would be understandable if the accuracy was the same for an easier problem, as classifying the acceleration of a car to predict the fuel consumption.

Table 15 – Best results for vibration training accuracy

Model Name	**Value**
HIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.96502227
HIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.96078545
LABELED_cognex_1_vibrationdata_outside	0.95857781
CHOPPEDLABELED_cognex_1_vibrationdata_outside	0.95704383
CHOPPEDHIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.95321351
LABELED_cognex_3_vibrationdata_inside	0.94363904
CHOPPEDHIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.93958986
LABELED_cognex_1_vibrationdata_inside	0.93742657

The fourth and final noticeable result from the accuracy is the huge difference between the accuracy of models with data passed through a low-pass filter and the rest. All of the eight models trained with low-pass filtered data are the worst. This is also expected because the shape of the data and the processing done after feeding the data to the model.

As mentioned in section 3.4.2, the block size is 100 and the step size is 15. This means the algorithm analyses 4ms of vibration data per iteration, turning the low pass filter close to a straight line in that period of time. For this reason, it is impossible to have good results using a low-pass filter with this dataset and the block size used. Low-pass filters are also not ideal in this case because the objective is to make predictions based on high frequency vibration data, and low-pass filters basically cut these values. Complete analysis about which filter or position is better for each setup will not be discussed before analysing all the results. To increase the understanding about the results, Figure 47 displays the loss for the datasets used.

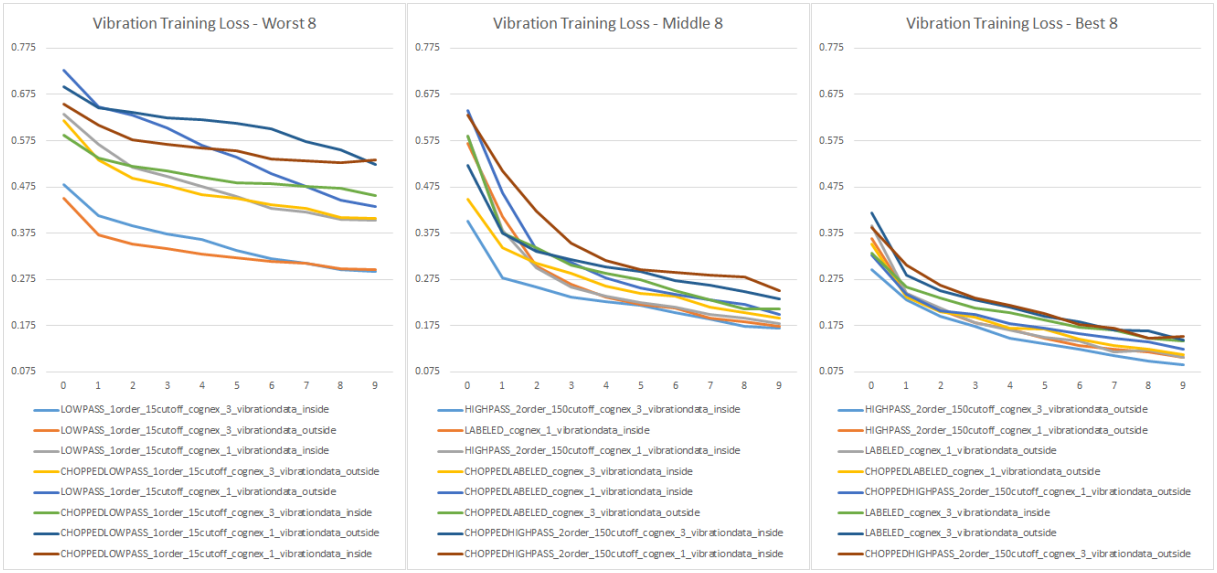


Figure 47 – Training loss for the vibration models

Loss follows the same trend as seen in Figure 46 for the accuracy. It has a smooth curve with very high values for low-pass filtered data. Table 16 quantifies the results of the best 8 results obtained for loss with the training dataset.

Table 16 – Best results for vibration training loss

Model Name	**Value**
HIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.089672
HIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.10506452
LABELED_cognex_1_vibrationdata_outside	0.10637832
CHOPPEDLABELED_cognex_1_vibrationdata_outside	0.11199314
CHOPPEDHIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.12448058
LABELED_cognex_3_vibrationdata_inside	0.1421558
LABELED_cognex_3_vibrationdata_outside	0.14362009
CHOPPEDHIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.15092763

If compared to Table 15, the training data loss table displays not surprisingly almost the same models. The difference comes from the loss function that penalizes more values very off from the expected value, as seen in the beginning of chapter 4.

4.2.2 Validation

Vibration loss is taken by testing the model after each epoch with part of the data that was previously taken aside from the training dataset. This data corresponds to 40% of the whole dataset, according to Table 5. This is just raw data that was not shuffled and has variance between classes. Figure 17 continues with analysis about the system behavior through time by showing validation accuracy for the 24 models trained with different datasets.

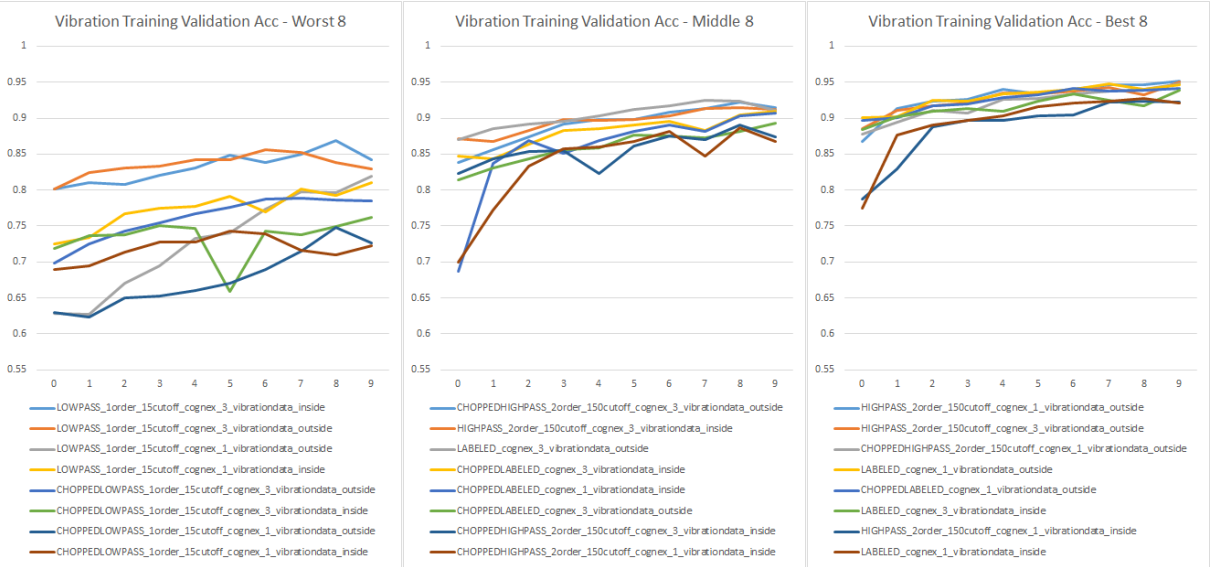


Figure 48 – Validation accuracy for the vibration models

It is visible that all of low-pass filtered data have still the worst results for accuracy. This indicates the low-pass filter is indeed ineffective for this dataset with this model settings. Other approaches could be taken for low-pass filter datasets, such as simple comparison of the actual value with some constant value to classify between the direction of acceleration, not being necessary a machine learning algorithm. Table 17 presents the best eight models according to validation accuracy.

Table 17 – Best results for vibration validation accuracy

Model Name	**Value**
HIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.95207387
HIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.95003396
CHOPPEDHIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.94819945
LABELED_cognex_1_vibrationdata_outside	0.94640988
CHOPPEDLABELED_cognex_1_vibrationdata_outside	0.94123107
LABELED_cognex_3_vibrationdata_inside	0.93918604
HIGHPASS_2order_150cutoff_cognex_1_vibrationdata_inside	0.92185664
LABELED_cognex_1_vibrationdata_inside	0.92141599

Compared to Table 15, the validation accuracy table for the best models displays accuracies a little worse, but nothing besides the expected. Seven of the eight models

appear on both tables, indicating that overfitting is unlikely to have happened and the model is able to predict unseen data. Figure 49 shows the best validation loss for the vibration dataset models, one of the most important metrics to analyse.

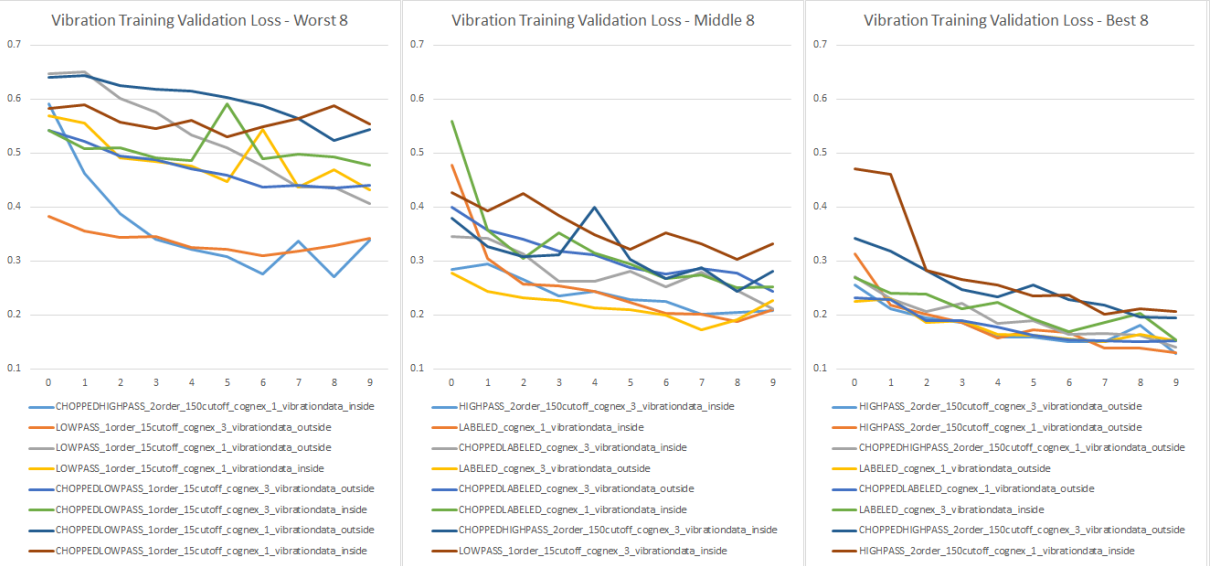


Figure 49 – Validation loss for the vibration models

The same trend seen in Figure 17 is also seen for the validation loss. It is possible to see a clear correlation between accuracy and loss on some epochs that have spikes. On epochs that the accuracy has a drop, the loss also has a value increase with proportional magnitude. To quantify these final results, Table 18 displays the validation loss for the best eight models according to this metric.

Table 18 – Best results for vibration validation loss

Model Name	**Value**
HIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.1280296
HIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.12978308
CHOPPEDHIGHPASS_2order_150cutoff_cognex_1_vibrationdata_outside	0.14018846
LABELED_cognex_1_vibrationdata_outside	0.15132636
CHOPPEDLABELED_cognex_1_vibrationdata_outside	0.1515362
LABELED_cognex_3_vibrationdata_inside	0.15318418
CHOPPEDHIGHPASS_2order_150cutoff_cognex_3_vibrationdata_outside	0.19433904
HIGHPASS_2order_150cutoff_cognex_1_vibrationdata_inside	0.20679566

If compared to Table 16 there are again seven models appearing on both tables. This just corroborates the results acquired from accuracy metrics.

According to training and validation results, the best models come from data acquired outside of the injection molding machine. The reason for this result is the higher stability of the location. On the data acquired with the accelerometer placed directly on the moving mold there is the interference of the actual mold opening and closing, which in this case is harmful for the quality of predictions.

The high-pass filter is also a good addition to optimize some results. The improvement from a model trained with the original dataset to one model trained with the same dataset after a high-pass filter is a consistent increase of around 0.6% for the data taken from outside the machine's chamber. As for the data taken in the mold of the machine, there was no improvement by using filtered data.

Chopping the beginning and ending of dataset also did not improve the results for both training and validation results. This is not necessarily as bad as it might appear. This dataset consists more of one class than the other so if the data is not cut, the model will pend to predict more one class than the other, making the model more accurate for a dataset where the data is divided by the same proportion. But if the data is divided by an inverse proportion, the model will have difficult to correctly predict this new dataset because it knows for training that most of the values belong to one class, that will no more be true for the inverted dataset.

4.2.3 Testing

For better analysis of the system accuracy, the models are tested using a different dataset. For the models trained with data from the period when the camera inside of the injection molding machine chamber with low resolution was recording, the data from when the camera in the same position with high resolution was recording is fed to analyse its output, and the same goes for the opposite. This makes all models predict unseen data from different times, being totally independent one from another, thus having some variation between them. Figure 50 demonstrates the different accuracy values obtained by feeding crossed data to the models.

To read this plot, the naming of datasets follows the pattern:

Camera Label_Data Mode_Filter_Location

Where:

- **Camera Label** is given accordingly to Table 7 [1]
- **Data Mode** is stripped for cropped data or normal for unaltered data
- **Filter** is the filtering used by each model and dataset
- **Location** is the location of the accelerometer, inside or outside the IMM chamber

The name given to each bar in the graph is respective to one model that was trained using data from its respective name. This model was then used to predict unseen data from the dataset when the opposite camera was recording. The results shown are respective to the accuracy got using the model represented by the name of each bar to predict the

[1] *cognex_1* means the same as *cognex1*

opposite dataset, which is marked by the "Dataset" label, filtered according to the model settings. The parameters from the model settings are respective to filtering applied to the training dataset and therefore also during testing, with the exception of chopping that was not used for testing. Models trained with low-pass filters are not displayed due to poor results, being unreasonable doing test result analysis. Results from training process and validation data can be found in sections 4.2.1 and 4.2.2.

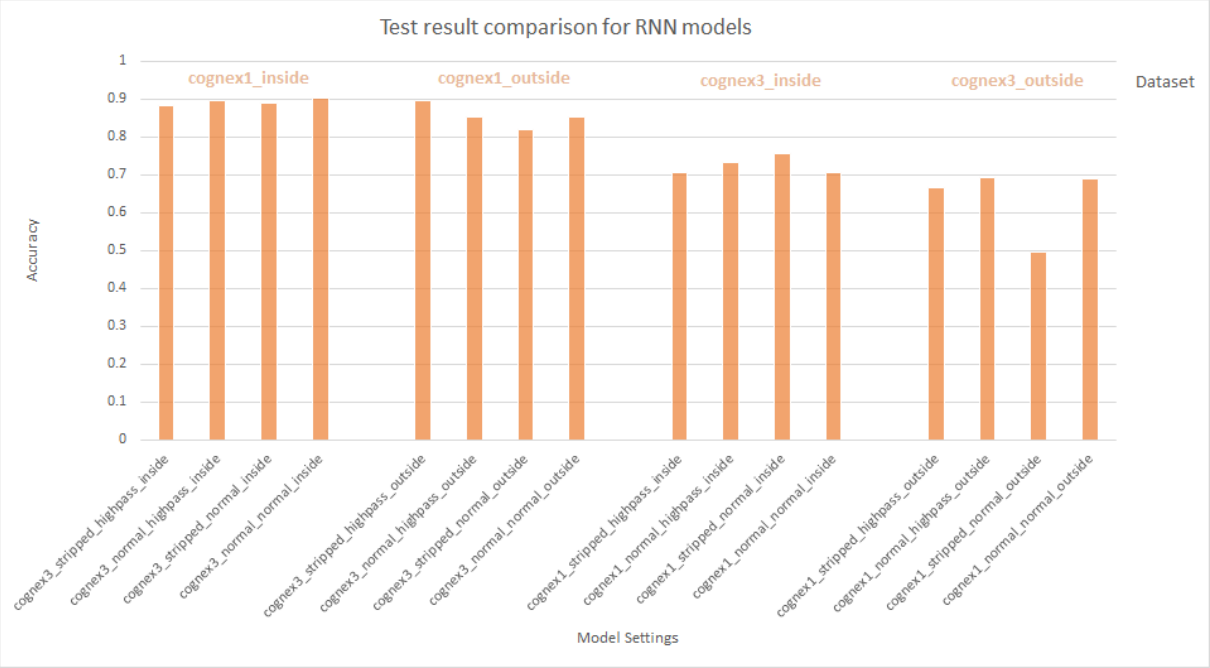


Figure 50 – Test result with unseen data for the vibration data models

It is possible to see a big difference between the two datasets. The models acquired by training with the *cognex3* vibration dataset can predict very accurately vibration data from *cognex1*, but the opposite is not entirely true. This is because variations on the original datasets. The *cognex3* vibration dataset has more variation, covering both datasets behaviors.

With this information, it is now possible to safely use the model trained with *cognex3* vibration data to generally predict vibration data from both inside and outside of the injection molding machine. On the other hand, models trained with *cognex1* vibration data are not reliable enough to predict with precision unseen data.

4.2.4 Output of vibration prediction

The model analyses and makes a prediction every block of 15 values of data taking 100 values into account [Section 4.2.1], so every step of 15 measurements or around 6ms of data, there is a new prediction. This data is then put to a table alongside with the parameters and expected values without any filtering.

Classification of vibration data is separated between two classes that are represented by the numbers 0 and 1. Therefore this data can be compared, analysed and if necessary, filtered afterwards.

4.2.5 Post-filtering

After predicting the whole dataset in blocks of 15, it is possible to match the time of this data with the time from the camera data. The camera acquisition for the frame with high resolution inside the machine has a frame rate of around 2.45 FPS [Section 3.4.1.1], meaning there is a frame every 410ms. To match the vibration with the camera predictions, the most common value of vibration prediction between the space of time between each frame was assigned for the prediction value to that frame. This is done only because the data from vibration have a much higher frequency than the camera, that is only acquired 2.45 times every second.

But this still brings some problems with oscillating values of vibration predictions. To solve this, it is possible to add some delay in order to only allow a change in value if there is a sequence of the same prediction in a row. This is only possible due to the nature of the process and must be used carefully, otherwise it can extend periods of time with wrong prediction. As the process is relatively slow, there is also no problem in delaying the output by one frame.

4.3 Link of camera and vibration result sources

It is possible to join the two result sources and compare them. To make it more visual, a good way of displaying both results is annotating each frame of a video taken from the machine while in operation with the prediction from a model trained with camera data and also annotate with a model trained with vibration data, then creating a video with all the predicted information of the process.

This way, it is possible to visually compare the effectiveness of each approach for solving the same problem. One sample frame of the final result can be seen in Figure 51.

After the acquisition of results, filtering and analysis, it is possible to make a real comparison between the two types of models and their performances. The CNN models obtained from camera data are superior in every aspect. They are robust and have reliable predictions, achieving amazing values if correctly optimized. At the same time, the RNN models created from vibration data are less reliable with the available data. They struggle to make good predictions and may not be the best option for the process in study.

Nevertheless, recurrent neural network models have shown some effectiveness for getting results by analysis of vibration data. To do this accurately, the training dataset needs to be large enough and also have great variation. Without these two requisites

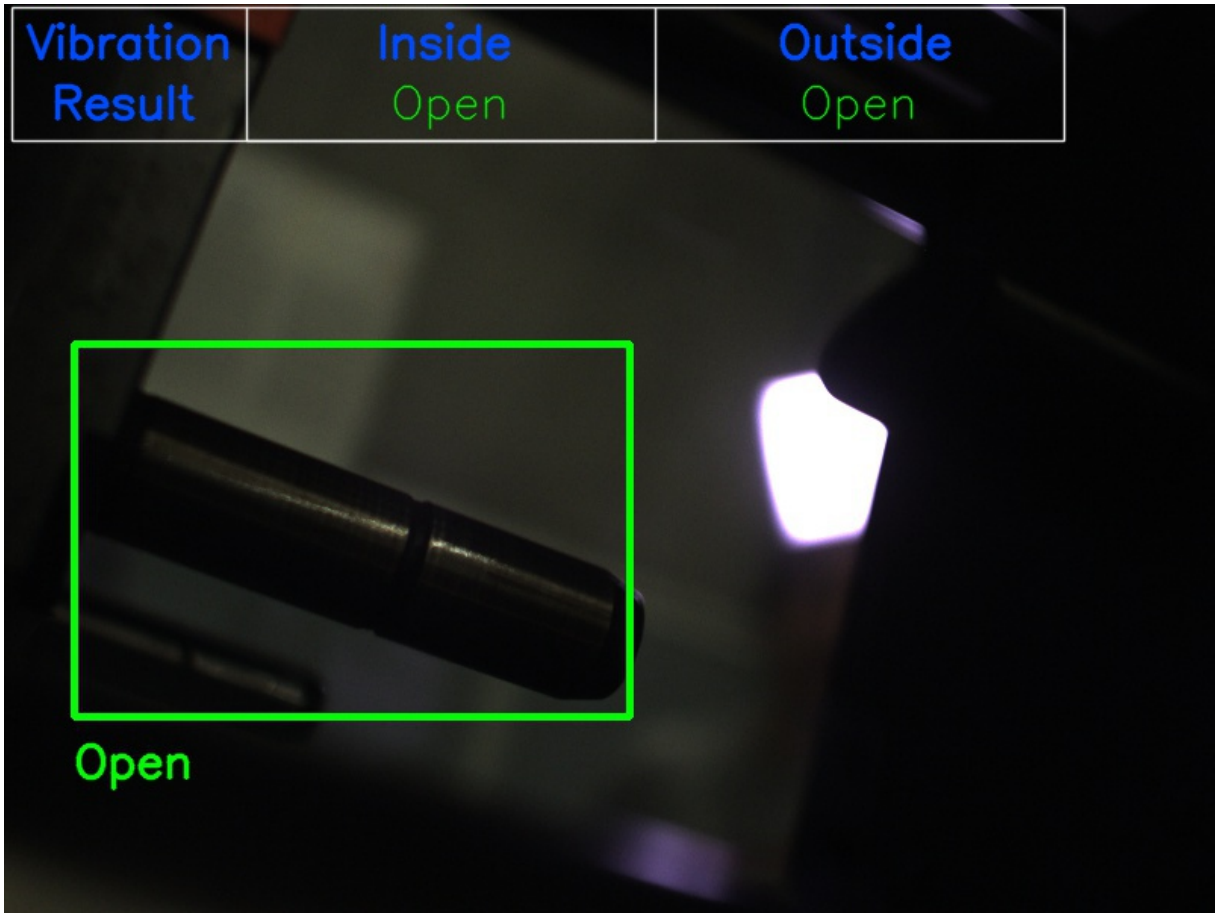


Figure 51 – Camera and vibration prediction in one frame

being satisfied, the model will perform poorly in this process. Another disadvantage of using RNNs in this particular process is the long training time due to the fact that RNNs have to be processed sequentially, while CNNs computations can be done in parallel [41].

In the meantime, convolution neural network models are by design faster to train and therefore can learn to recognize patterns in much less time. Since they are time independent, there is no need of making computations over the same data more than once. Every image from the dataset counts as how the model will be, but there is no direct relation between them. The prediction is also done by just taking one single image and giving a single result without the relation of anything else. This is a good approach for the process in study because the difference between a closed and open mold is very significant.

5 Conclusions

As target of the thesis, it was defined to predict the state of an injection molding machine by live images and vibration measurements. To do so, it was proposed to use convolutional neural networks to train a model based on camera images and recurrent neural networks to train a model based on vibration data. The expected result is creating an algorithm that is able to classify accurately the information about the machine's mold state and output if it is open or closed by visual and vibration data.

From the results obtained and analysis done, it is possible to state that the models worked as intended and satisfied the expectations. The two types of datasets are very different and therefore have to be treated differently. One is straight forward and easily readable by humans, while the other is impossible for a human being read accurately.

The objective from studying the behavior of an injection molding machine by a video of it and creating a complex neural network model is purely educational. Simpler methods could have been applied to get similar results. The main objective of using these complex algorithms is to try to understand the result behavior for different settings of models and check the line where the model gets too complex for making good predictions. This does not mean that models like these could not be applied in real industries. They work very good and are very reliable if the quality and complete accuracy of results are requirements.

Despite being worse than the CNNs with the camera data, the RNNs with the vibration data were proven to work properly under some circumstances. It is not an easy task for an algorithm to tell whether the machine has its mold open or closed just by vibration data because there is not much change in the dataset between each state. There is a clear difference in vibration when the mold is in movement, which was not the objective to predict. In order to make a model learn the vibration behavior for each state and be capable of reading the correct state after some variation or after a reboot of the machine it must be very good design and trained. The training dataset must have variation that it can map all the machine's possibilities of vibration. Just after doing this the model is going to reliably predict the state of the machine.

The RNNs does not have perfect accuracy, but after filtering the data after the prediction with the addition of a small delay and pairing the results camera images to get a slower update frequency, the results become very close to perfection. The best models are able to predict the exact same values of the camera and rarely get confused by anomalies in the dataset. This kind of sensor can be very useful in some situations where direct source of information or cameras are not an option.

Despite having a better result, it does not mean that CNNs are superior to RNNs in any way. They have two different purposes and the best one depends on the system to be

studied. Convolutional neural networks do not have any relation with time while recurrent neural networks' most basic principle is to solve time-series problems. Nonetheless, the two result sources work in their own way and are beneficial for acquiring information about the state of the process.

Overfit is not completely discarded for the CNNs because the best scenario would be to have more machines with different behaviors to test the system. It can be prevented by leaving more data outside of the training dataset, but if there is not good variation between the training dataset and validation or testing dataset, it is not possible to detect if the model is indeed overfitted.

From the results obtained it is also possible to state that the camera does not necessarily needs to be placed inside of the injection molding machine's chamber, despite being more effective. Placing the camera outside of the machine brings problems but also solves the temperature problem, which can be very crucial depending on the camera used. It does not get much affected by the poorer image detail, but there is the requirement of nothing passing in front of it or also accidentally changing the camera positioning.

To improve the results from this work, it is firstly change both sensors. The camera, despite being very good for working on its own, is not very effective for the efforts of this work because it does not provide a good framerate and also does not provide a good solution to lower the video resolution, together with the fact it does not support high temperatures. This could be changed for a camera that does not have algorithms embedded because, despite useful to pre-filter data, it is not essential to the acquisition and labeling.

The vibration sensor must also be changed by a more reliable data source. It does not provide data with excellent quality due to its communication protocol. For being asynchronous, it ends up not getting data any data in a short period of time, but this short period of time when the data frequency is around 2.5kHz becomes a relative long time, losing valuable data information. To enhance the performance of this sensor, the perfect solution would be to use an industrial accelerometer with high frequency synchronous data. Another approach for the vibration sensor and its data would also be an option. Changing the focus from classifying the state of the machine to detecting anomalies while in operation could be a better match for vibration data analysis.

References

1 NAVLANI, A. *Understanding Logistic Regression in Python (article) - DataCamp*. 2019. Available at: <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python>.

2 CIFAR-100 on Benchmarks.AI. 2018. Available at: <https://benchmarks.ai/cifar-100>.

3 PRODUCTION Deep Learning with NVIDIA GPU Inference Engine. 2018. Available at: <https://devblogs.nvidia.com/production-deep-learning-nvidia-gpu-inference-engine/>.

4 2D Convolution – Abdulsamet İLERİ – Medium. 2019. Available at: <https://medium.com/@abdulsamet.ileri/2d-convolution-ced5d339aa5>.

5 HAYKIN, S. S. *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009.

6 SANTOS, L. A. dos. *Dropout Layer · Artificial Inteligence*. 2019. Available at: <https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/dropout_layer.html>.

7 FAN, J.; GIJBELS, I. *Local Polynomial Modelling and Its Applications: Monographs on Statistics and Applied Probability 66*. Taylor & Francis, 1996. (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). ISBN 9780412983214. Available at: <https://books.google.de/books?id=BM1ckQKCXP8C>.

8 QUINLAN, J. R. Induction of decision trees. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 0885-6125. Available at: <http://dx.doi.org/10.1023/A:1022643204877>.

9 JACKSON, A. S. et al. Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. *GigaScience*, v. 6, n. 10, 08 2017. ISSN 2047-217X. Available at: <https://dx.doi.org/10.1093/gigascience/gix083>.

10 HUBEL, D.; WIESEL, T. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology*, v. 160, p. 106–154, 1962.

11 HUBEL, D. H.; WIESEL, T. N.; LEVAY, S. Plasticity of ocular dominance columns in monkey striate cortex. *Philosophical Transactions of the Royal Society of London Series B*, v. 278, p. 377–409, 1977.

12 LECUN, Y.; BENGIO, Y.; HINTON, G. E. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. Available at: <https://doi.org/10.1038/nature14539>.

13 CS231N Convolutional Neural Networks for Visual Recognition. 2019. Available at: <http://cs231n.github.io/convolutional-networks/>.

14 SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

15 , F. ; , P. . Universal minimum-rate sampling and spectrum-blind reconstruction for multiband signals /. p. 99, 1997.

16 RAO, S. S.; FAH, Y. F. *Mechanical vibrations; 5th ed. in SI units*. Singapore: Prentice Hall, 2011. Available at: <https://cds.cern.ch/record/1398617>.

17 SONG, H. X. et al. Design and performance analysis of laser displacement sensor based on position sensitive detector (PSD). *Journal of Physics: Conference Series*, IOP Publishing, v. 48, p. 217–222, oct 2006. Available at: <https://doi.org/10.1088%2F1742-6596%2F48%2F1%2F040>.

18 MIDé TECHNOLOGY CORPORATION. *Slam Stick Shock & Vibration Data Loggers*. [S.l.], 2017. Rev. 11.

19 JAIN, L. C.; MEDSKER, L. R. *Recurrent Neural Networks: Design and Applications*. 1st. ed. Boca Raton, FL, USA: CRC Press, Inc., 1999. ISBN 0849371813.

20 BAI, S.; KOLTER, J. Z.; KOLTUN, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. Available at: <http://arxiv.org/abs/1803.01271>.

21 ELBAYAD, M.; BESACIER, L.; VERBEEK, J. Pervasive attention: 2d convolutional neural networks for sequence-to-sequence prediction. *CoRR*, abs/1808.03867, 2018. Available at: <http://arxiv.org/abs/1808.03867>.

22 ZHANG, X. et al. Drawing and recognizing chinese characters with recurrent neural network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 40, n. 4, p. 849–862, April 2018. ISSN 0162-8828.

23 HOCHREITER, S. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991.

24 CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. Available at: <http://arxiv.org/abs/1412.3555>.

25 HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, v. 9, p. 1735–80, 12 1997.

26 ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Available at: <https://www.tensorflow.org/>.

27 PASZKE, A. et al. Automatic differentiation in pytorch. 2017.

28 CHOLLET, F. et al. *Keras*. 2015. <https://keras.io>.

29 SMITH, J. *Introduction to Digital Filters: with Audio Applications*. [S.l.]: Booksurge Llc, 2006. v. 2.

30 BUTTERWORTH, S. On the Theory of Filter Amplifiers. *Wireless Engineer*, 7, 1930.

31 MATTHAEI, G. L.; YOUNG, L.; JONES, E. M. T. *Microwave Filters, Impedance-matching Networks, and Coupling Structures*. [S.l.]: New York : McGraw-Hill, 1964.

32 CHAVAN, M. S.; AGARWALA, R.; UPLANE, M. D. Digital elliptic filter application for noise reduction in ecg signal. In: *Proceedings of the 4th WSEAS International Conference on Electronics, Control and Signal Processing*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2005. (ICECS'05), p. 58–63. ISBN 960-8457-38-6. Available at: <http://dl.acm.org/citation.cfm?id=1974895.1974908>.

33 SEDRA, A.; SMITH, K. *Microelectronic Circuits*. Sauders College Publishing, 1991. ISBN 9780030526138. Available at: <https://books.google.it/books?id=y28dBAAACAAJ>.

34 ARBURG GMBH + CO KG. *ALLROUNDER 220/270 S Datenblatt*. [S.l.], 2004.

35 SENSOPART. *V10 / V20 Smart Cameras*. [S.l.], 2014.

36 SENSOPART. *In-Sight Micro Series Vision System Installation Manual*. [S.l.], 2014. Rev. B.

37 BOSCH CONNECTED DEVICES AND SOLUTIONS GMBH. *XDK 110 Cross Domain Development Kit*. [S.l.], 2015.

38 IMAGE Preprocessing - Keras Documentation. 2019. Available at: <https://keras.io/preprocessing/image/>.

39 OPENCV library. 2019. Available at: <https://opencv.org/>.

40 HARBOUR, M. G. Real-time posix: An overview. 1993.

41 BRADBURY, J. et al. Quasi-recurrent neural networks. *CoRR*, abs/1611.01576, 2016. Available at: <http://arxiv.org/abs/1611.01576>.