POLITECNICO DI TORINO

Master's Degree in Electronic Engineering

Politecnico
di Torino

Master's Degree Thesis

Machine learning techniques for microwave food contamination detection

Supervisors

Prof. Mario Roberto CASU

Prof. Francesca VIPIANA

Candidate

Giacomo MARCHIONI

Accademic year 2020/2021

Summary

Ensuring the safe consumption of suitable food products is a top priority for food companies, both to obtain the necessary certificates to enter the market, and to avoid the negative consequences that the company could incur if a product contaminated by foreign bodies fails to be eliminated before it reaches the consumer, such as having to take a lawsuit, as well as losing general trust on the part of its customers. Even more serious could be the consequences for those who consume such products, from mild illness to death; therefore there are several standard prevention systems that identify and eliminate contaminated products such as x-rays or infrared systems. Such systems are not reliable enough as they cannot detect low density materials, have low depth of penetration and x-rays can be harmful to operators. An emerging technique in the food world is Microwave Imaging Technology, which allows to overcome these problems and can detect low density materials.

In this thesis, a Machine Learning algorithm is designed and implemented on a Microwave Sensing system that is able to detect foreign bodies in jars of cocoa-hazelnut spreadable cream during their passage on a conveyor belt at a speed of 30 and 50 $\mathrm{cm\,s^{-1}}$. The system consists of a convey belt on which 6 antennas are positioned to acquire the signals when a jar passes under them, a network vector analyzer that acquires the signal from the antennas and processes it directly or transmits it to a microcontroller which processes it and will enable the removal of that jar if contaminant is detected. The binary classifier based on the Multilayer Perceptron family has been trained with seven different datasets. A dataset acquired in a laboratory environment to find the hyperparameters that gave the best results, four other datasets subsequently acquired by the system installed in the company, two of which with safflower oil and two with spreadable hazelnut cream, each one of them at two different speeds, 30 $\mathrm{cm\,s^{-1}}$ and 50 $\mathrm{cm\,s^{-1}}$. Finally, the last two datasets are created by joining the datasets at two different speeds of the same material; the latter were created to verify the possibility of implementing a single algorithm that detects contamines regardless of the speed of the conveyor belt. The results obtained were very promising, using the best network found, with the

first dataset an accuracy of 99.76% was achieved while using the dataset of the two-speed spreadable hazelnut cream we reached 99.91%. The network was loaded by converting it into High Level Syntesis C code and synthesized for two different systems in order to study the inference that gave the best results:

- on the network analyzer computer whose output commands a microcontroller that signals the presence of contaminants.
- on a microcontroller that receives data from the network analyzer.

The best inference processing times were found in the case of the microcontroller, reaching an average of 3.7 ms per acquisition as opposed to the 10 ms of the network analyzer. Despite this, the performance bottleneck is in the communication of the data received by the Network Vector Analyzer, using a UART communication at 115200 bit/s and having 660 data in floating point per acquisition to be transmitted, the minimum transmission time will be 183 ms.

Potential future improvements will be an expansion of the datasets with different materials and different contaminants, in order to have a greater number of contaminants that can be detected on different products. In order to reduce the communication times, a data acquisition system could be used that is able to manage SPI or I2C protocols which would reduce the communication times to the same order of magnitude as those of inference. In addition, the system could be made cheaper by replacing the Network Vector Analyzer and assigning the data acquisition task to the microcontroller so as to definitively eliminate communication times.

Table of Contents

List of Tables

List of Figures

Acronyms

ADAM

Adaptive moment estimation

AI

Artificial Intelligence

AUC

Area Under the Curve

CV

Cross Validation

ELU

Exponential Linear Unit

HDF

Hierarchical Data Format

FN

False Negative

FP

False Positive

FPR

False Positive Rate

I2C

Inter Integrated Circuit

ISO

International Organization for Standardization

LED

Light Emitting Diode

ML

Machine Learning

MLP

Multilayer Perceptron

MWI

Microwave Imaging

MWS

Microwave Sensing

NaN

Not a Number

PCB

Printed Circuit Board

PTFE

Polytetrafluoroethylene

PXI

PCI eXtensions for Instrumentation

ROC

Receiver Operating Characteristic

ReLU

Rectified Linear Unit

SELU

Scaled Exponential Linear Unit

SGD

Stochastic Gradient Descent

SLG

Soda Lime Glass

SPI

Serial Peripheral Interface

SRAM

Static Random Access Memory

TN

True Negative

TP

True Positive

TPR

True Positive Rate

UART

Universal Asynchronous Receiver Transmitter

VNA

Vector Network Analyzer

WRP

Weight Regularization Parameter

Chapter 1

Introduction

1.1 Food Contamination

In the food industry the problem of food contamination is fundamental, because a contaminated product could reach the customer who would suffer damage, whether physical or health, would have negative economic repercussions, such as the withdrawal of the entire batch of products.
This would compromise the credibility and reliability of the company. Furthermore, respecting high food safety standards would be beneficial for a company as: it would respect the criteria and obtain the necessary certifications to enter the market, it would avoid legal fees as a result of damage to customers, it would keep its reputation at high levels, it would increase sales and exports.

There are four different types of contamination [1]:

- Chemical: Occurs when the product comes into contact with any type of chemical substance, such as residual cleaning products on surfaces or pesticides when applied close to the product.
- Microbial: It happens when the product is contaminated by bacteria, molds, viruses, fungi, following an insufficient heat treatment, an incorrect slaughtering process, cross-contamination due to proximity to other food.
- Allergenic: It happens when the product comes into contact with another product that causes allergic effects
- Physical: It happens when the product is contaminated by a foreign object

Foreign objects can be of various types:glass, fragments, bone, plastics, ceramic, wood, metals, paper, organic origins and others.

Their origin is divided into two categories [2]:

- extrinsic when their origin is external, such as a fragment of packaging or shavings of machinery in the assembly line.

- intrinsic when their origin is internal, for example the failure to remove bone fragments in a piece of meat.

Physical contamination could be the cause of the other three types of contamination, and therefore lead to a worsening of the contamination, for this reason there are systems created specifically to detect foreign objects.

The control and prevention of contaminants can be carried out in four different technological system:

1. Off-line by taking a sample that is taken to a laboratory where tests and analyzes are performed. This type of operation is time consuming, a contamination report could arrive once the production is finished with a consequent economic loss for the company.

2. At-line as for off-line, the sample is taken from the production line, but the analyzes are carried out near it.

   Both of these methods can be destructive for the product and disadvantageous [3] as they involve the presence of personnel who could introduce contamination risks in the various steps.

3. On-line uses a sub-line that takes samples from the main line and analyzes them without altering the speed of the stream. The analyzes of these three methods being of a sub-group of the total products may not be representative or some contaminated products are missing.

4. In-line analyzes each product in real time without altering the speed of the line

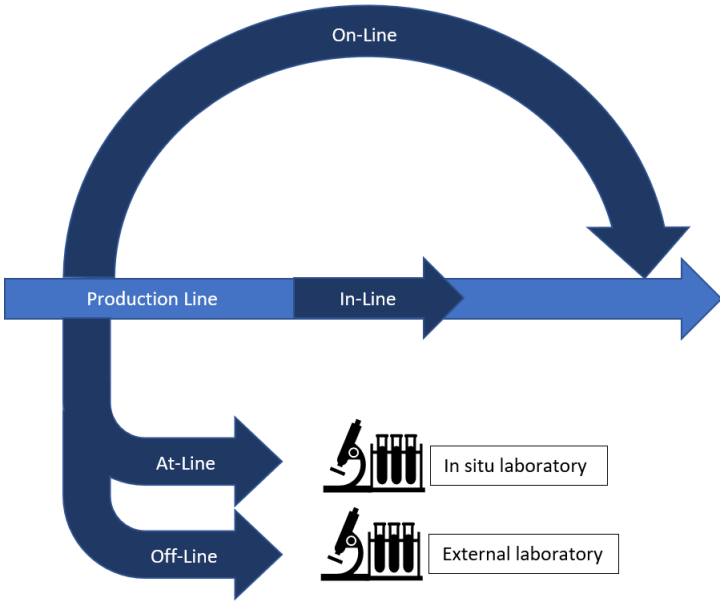


Figure 1.1: Summary of the functioning of technological system prevention

In the field of food products the techniques used are online and in-line, since the technique used in this thesis is competing in these categories, some of the most used are now exposed:

- Visual inspection: Operators eliminate products on the line that do not meet standard criteria. The selection criteria are chosen according to the client's needs and in order to comply with the regulations. Although it is a very simple method to make it is prone to be inaccurate.
- Mechanical filters: Mechanical filters which, depending on the shape of the product, eliminate any non-conforming element. Fundamental for the proper functioning of some types of machinery, economic, but its maintenance stops the production line.
- Magnets: They are effective for removing ferromagnetic materials, but contaminants must be removed from the magnet and therefore requires maintenance.
- Metal detectors: like the category of magnets, they detect ferromagnetic materials, but without eliminating the contaminant which is instead removed by a third-party machine activated by the metal detector signal.
- Near Infrared Imaging: Uses the response of different materials to infrared waves (800-2500) nm, the wave absorption variation is acquired by a sensor which is subsequently processed by an algorithm which creates an image of

it. It has a low level of penetration and does not allow the study of fresh products because its wavelength has a great absorption from water.

- X-rays Imaging: It uses a high intensity ray with a wavelength in the order of nanometers, materials with a large density absorb the rays, all the remaining rays are acquired by a sensor that creates an image. Therefore, the higher the density of the contaminant, the greater the contrast in the image. Ionizing rays are dangerous for operators, and the properties of the product may vary. Not all types of low density plastics and contaminants are detectable.
- Terahertz imaging [4]: As for the previous two methods, but with a beam with a wavelength of terahertz (0.1-10) THz. Its rays are not ionizing, but it has a high processing time of the acquired signal and a low penetration and high attenuation to water.
- Ultrasound imaging: The probe that emits sound pulses (1-30) MHz is placed in contact with the product, the signal is reflected and refracted according to the acoustic impedance of the contaminant depending on the type of material. The image is created in real time, but the probe must be in contact with the product.
- Microwave Imaging: This technique takes advantage of the dielectric difference between materials. An antenna radiates with an electromagnetic signal (0.3-300) GHz at low power, for a higher resolution multiple antennas are introduced; the product and acquires the reflected wave of the electromagnetic field, if there was a contaminating material the dielectric discontinuity would generate a dispersion of the signal, the greater their difference the greater the dispersion. The signal is then processed to create a tomographic image representing the dielectric profile of the product, which is analyzed to verify the presence or absence of a contaminant. This technique offers several advantages: it does not use ionizing rays, it is in real time, the beam can be adjusted according to the required depth or resolution level, it is not absorbed by water and is able to detect low-densities contaminants. Not requiring contact with the product and having fast analysis times, it is introduced into in-line contaminant detection systems.

1.2 Thesis Focus

The goal of this thesis is to design and analyze a microwave sensing system (MWS) that detects the presence of contaminants in a jar of hazelnut spread using a machine learning (ML) algorithm based on the binary classifier multi-layer perceptron (MLP).

As a consequence, the system under study will advance from the fourth to the sixth level in the ISO Technology Readiness Level scale, from "Technology validated in the laboratory" to "Technology demonstrated in an industrially relevant environment", approaching a possible industrial commercialization.
The neural network was trained with seven datasets summarized in the table:

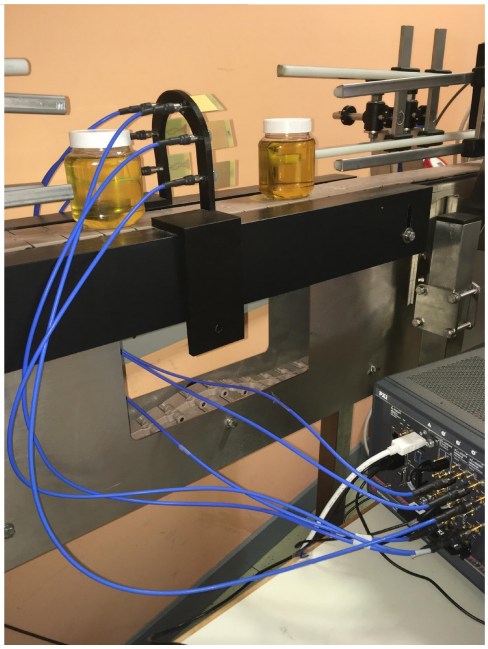
	Medium	Conveyor Belt Speed $\{\mathrm{cm\,s^{-1}}\}$	Free	Contaminated	Total
1	Safflower Oil	30	1702	3753	5455
2	Safflower Oil	30	1000	2000	3000
3	Safflower Oil	50	3000	1802	4802
4	Hazelnuts Cream	30	3723	1251	4974
5	Hazelnuts Cream	50	1997	1597	4594
6	Safflower Oil	30 and 50	4000	3802	7802
7	Hazelnuts Cream	30 and 50	6720	2848	9568

Table 1.1: Table containing the material used (safflower oil, hazelnuts cream), the speed of the conveyor belt (30 $\mathrm{cm\,s^{-1}}$, 50 $\mathrm{cm\,s^{-1}}$) and the number of acquisitions for contaminated and non-contaminated product

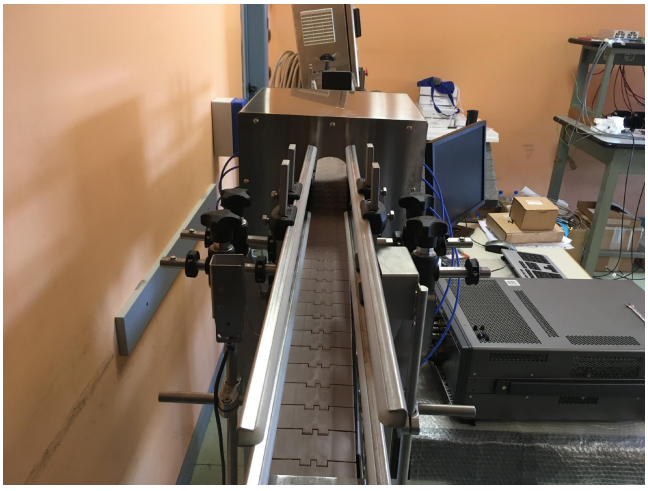
Each dataset has a group of acquisitions without contaminant and one with the presence of contaminants of different types: polytetrafluoroethylene (PTFE), soda-lime glass (SLG), nylon, wood, 3 different plastic splinter.
The first dataset was acquired in the laboratory environment in order to verify the effectiveness of the implementation of the system in the laboratory phase, the second, third, fourth and fifth were acquired in the industrial environment, while the sixth and seventh are respectively the union of the second and third and the union of the fourth and fifth; the latter two have the purpose of verifying whether it is possible to distinguish the presence of contaminants on a line that can vary speed.

The acquisition system consists of the following parts:

- An arch formed by six equidistant antennas and positioned so as to effectively cover the jars profile of dimensions (11.43 x 10.67 x 8.69) cm
- A structure for the protection and containment of measurements from electromagnetic signals
- A vector network analyzer with six inputs, one for each antenna, which acquires the scattering parameters of the network
- A cotroller, in this case the computer integrated in the VNA, thanks to a matlab script, saves the acquisition data



(a) Two sample jars placed on the conveyor belt and the arch with six antennas connected to the six inputs of the VNA.



(b) The insulating box positioned around the antennas.

Figure 1.2: The acquisition system.

Finally, once the neural network has been trained with the optimal hyperparameters found through Bayesian research, the algorithm inference is performed both by the computer integrated in the VNA and by a NUCLEO-F401RE microcontroller, subsequently compared to verify which of the two allows a more reliable and fast processing of the acquired data.

Chapter 2

Machine Learning Theory

2.1 Machine Learning Introduction

ML is a sub-field of artificial intelligence (AI) that has become more relevant since 1980,when the potential of developing the ability in a computer to learn and build models that could predict results by interpreting data was understood. This technology demonstrates its strength and synergy for an engineer in three fundamental aspects.
First of all, it allows you to reduce programming times. for example, to create a program that corrects lexical errors that should consider all the rules and the exception of the language, it would take months of work-hours, much more simply by feeding a sufficient number of examples to an ML tool, you would have a reliable product in much less time.
Secondly, it allows you to customize a product for different market needs. If I decided that the lexical error correction program could have another selectable language, it would be enough to use the same ML algorithm, training it with examples of the new language.
Finally, it allows to solve problems that would be unsolvable using the traditional engineering method; the classic example is facial recognition, it is a task that an ML algorithm trained with a dataset of facial images can solve relatively easily, while it can be much more difficult to obtain the same accuracy with a traditional algorithm.

2.2 Multi-layer Perceptron

A learning problem considers a dataset containing n samples, each single sample value are called attributes or features.
Two different categories are used to extrapolate properties from this unknown data:

- Supervised learning, in which the data used for training belong to known target values. This problem is classified in two ways:
  - Classification: The data belongs to multiple labeled classes or categories and you want to learn how to predict unlabeled data. This is the discrete type of supervised learning, where each data has a precise number of categories in which to be tagged. The example of use of this category is, as in this thesis, the recognition of contaminant in food, where the categories are "contaminated food" and "uncontaminated food".
  - Regression: The data belong to one or more continuous variables. Predicting prices of a house given the features like size, living area, number of room etc.
- Unsupervised learning, in this case the training data consist of a set of vectors to which no target value is associated. The goal in this case is to find similar groups within the data (clustering), or to determine the distribution of data in the input space (density estimation), or to project data from a high-dimensional space down to fewer dimensions for the display purpose.

A neural network is a computing system consisting of interconnections of artificial neurons. Each connection transmits the processed and weighted signals to other neurons, increasing or decreasing the strength of the signal at that connection.
The architecture of the neural network is composed of an input layer that interfaces with the feature values of a sample of external data, a single or a series of intermediate layers called hidden layers which have a variable number of neurons, and an output layer that provides the prediction with a certain probability.
The classifier used in this thesis is an example of artificial neural network called the Multi-layer Perceptron (MLP).

The MLP classifier is a supervised learning algorithm [5] that learns a function $f(\cdot)$: $\mathbb{R}^m \to \mathbb{R}^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $x = x_1, x_2, ..., x_m$ a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

Given a set of training examples $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $x_i \in \mathbb{R}^n$ and $y_i \in \{0, 1\}$, an MLP with one hidden layer and one neuron learns the function

$$f(x) = W_2 g(W_1^T x + b_1) + b_2$$

where $W_1 \in \mathbb{R}^m$ and $(W_2, b_1, b_2) \in \mathbb{R}$ are model parameters. W_1, W_2 represent the weights of the input layer and hidden layer, respectively; and b_1, b_2 represent the bias added to the hidden layer and the output layer, respectively.
$g(\cdot) : \mathbb{R} \to \mathbb{R}$ a nonlinear "activation" function.

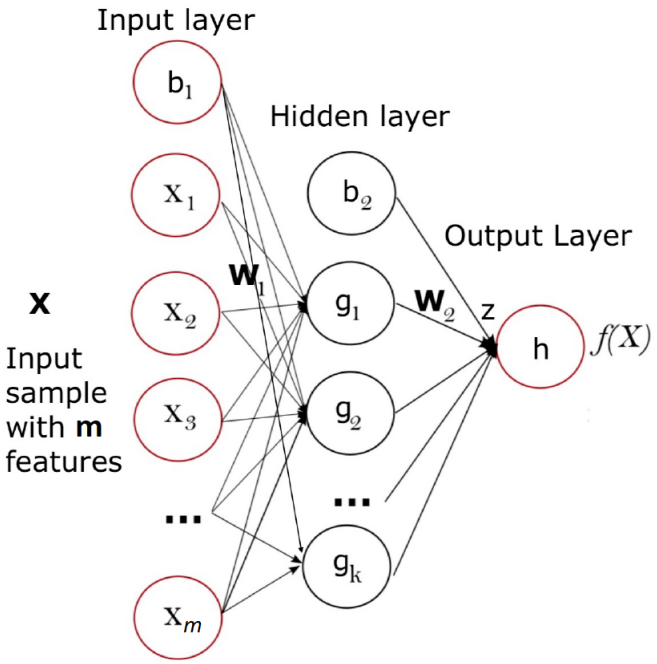


Figure 2.1: MLP architecture with one input layer, one hidden layer and one output layer.

The nodes of the input layer are equivalent to the number of features that the dataset has, while the nodes of the output layer are equivalent to the number of classes to predict. In the case of a binary classifier the output node is one.
The logistic function is one of the possible functions used in the output layer of the binary classifier, where the output is a value between zero and one, $f(x) \in \{0,1\}$.

$$g(z) = \frac{1}{(1 + e^{-z})}$$

A threshold value is used to distinguish between the two cases, usually its value is 0.5, positive class if the output is greater than the threshold, negative class if it is

less.

When the classifier is not binary $f(x)$ would be a vector of size $n_{classes}$, and the softmax function can be used:

$$g(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

where z_i represents the $i-th$ element of the input to softmax, which corresponds to class i, and k is the number of classes. The result is a vector containing the probabilities that a given sample x belongs to each class. The output is the class with the highest probability.

To find the optimized values of the weights ($\mathbf{W}$) and biases ($\mathbf{b}$), the neural network uses the loss function during training. A typical loss function for classification is the cross-entropy, which in binary case is given as:

$$Loss(\hat{y}, y, \mathbf{W}) = -y \ln \hat{y} - (1-y) \ln (1-\hat{y}) + \lambda\, WRP(\mathbf{W})$$

where $\hat{y}$ is the predicted label, y is the label of the input data, $\lambda\, WRP(\mathbf{W})$ is a regularization term called weight regularization parameter (WRP) that penalizes complex models to avoid the overfitting of the dataset; and $\lambda > 0$ is a non-negative hyperparameter that controls the magnitude of the penalty.
The overfitting of a model happens when during the training the algorithm learns the detail of the dataset, including the data noise. It negatively impacts the performance of the model on new incoming data because the new data have a different random fluctuations that are not recognise by the trained model. The effects of overfitting are a good performance on the training data, but a poor generalization to any other data.
The most used regularization terms are:

- L1: $WRP = \sum_{i=1}^{n} |w_i|$
- L2: $WRP = \frac{1}{2} \sum_{i=1}^{n} w_i^2$
- Elastic Net: $WRP = (1-\alpha) \sum_{i=1}^{n} |w_i| + \frac{\alpha}{2} \sum_{i=1}^{n} w_i^2$
  where $\alpha \in \{0,1\}$ controls the combination of the two regularization.

L2 encourages weight values toward 0 (but not exactly 0), also it encourages the mean of the weights toward 0, with a normal (bell-shaped or Gaussian) distribution.

The training of a neural network starts from initial weight initialization to find the proper weight value, that can be zero, one or random values [6]; since there are several local minima, different initializations lead to different accuracies.

A typical training algorithm for an MLP is based on gradient descent. It adjusts parameters iteratively, gradually finding the best combination of weights and bias to minimize loss.

In gradient descent, the gradient $\nabla Loss_W$ of the loss with respect to the weights is computed and deducted from W. This is expressed as:

$$W_{i+1} = W^i - \epsilon \nabla Loss_W^i$$

where i is the iteration step, and ϵ is the learning rate with a value larger than 0. If the learning rate is too small, learning will require too many iterations and so potentially a long time to finish, while if the learning rate is too large the learning algorithm might miss the minimum because at each iteration it would bounce around it, as shown pictorially in Fig. 2.2.

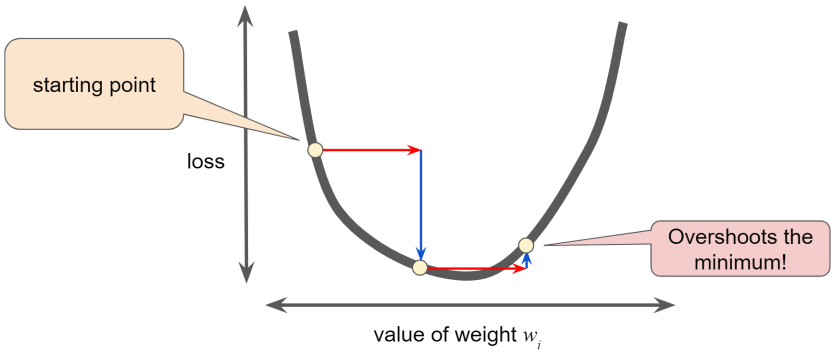


Figure 2.2: High values of learning rate leads to minimum overshooting.

The algorithm stops when it reaches a preset maximum number of iterations, or when the improvement in loss is below a small, predefined threshold.

The datasets used to train the ML algorithm contain from thousands to billions of data, the time to calculate the gradient using a batch containing the entire dataset (batch gradient descent) would be costly in computational terms, even if the total of the contained data presents some redundancy that help to smooth out noisy gradients it is not a practical method.
A less costly and faster method to compute the gradient is the stochastic gradient descent (SGD) algorithm [6]. It evaluates the gradient only in a random subset of the dataset at each iteration with a batch size of one, this means that the result is

not the best obtainable, but it ensures a fast convergence around the minimum loss value [5].

A compromise between SGD speed and full-batch accuracy is the Stochastic Gradient Descent Mini-batch (SGD mini-batch), in which a batch is composed of between 10 and 1000 elements, always chosen at random.
There are other algorithms such as Adaptive moment estimation (Adam).
The Adam algorithm calculates an exponential moving average of the gradient and the squared gradient which are weighted by two parameters that control the decay rates of these moving averages [7]. This method computes a learning rate for every parameter (weight) and it adapts it each iteration. Thanks to this the learning rate can be discarded from the list of hyper-parameters to be tuned, thus making the search for the remaining hyper-parameters faster as explained at chapter 2.5.

After finding the weights to be updated, a backpropagation algorithm [8] [9] updates them from the output to the previous layers.

Backpropagation is the algorithm for determining how a single training example would like to tune the weights and biases, in terms of what relative proportions to those changes cause the most rapid decrease to the cost function. An approach to study backpropagation is to understand how sensitive the cost function is to the parameters, thus to know which adjustment to those term will cause the most efficient decrease to the cost function.

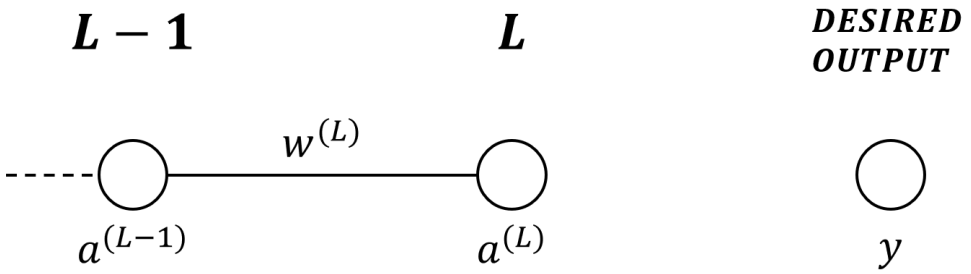


Figure 2.3: Network composed by a single neuron for each layer.

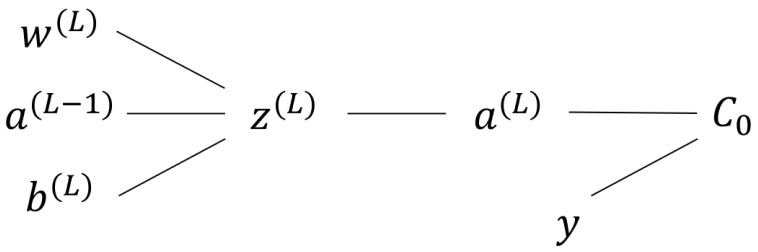
Starting with a simple network compose by a single neuron for each layer (Fig. 2.3) the cost function is defined as:

$$C_0 = (a^{(L)} - y)^2$$

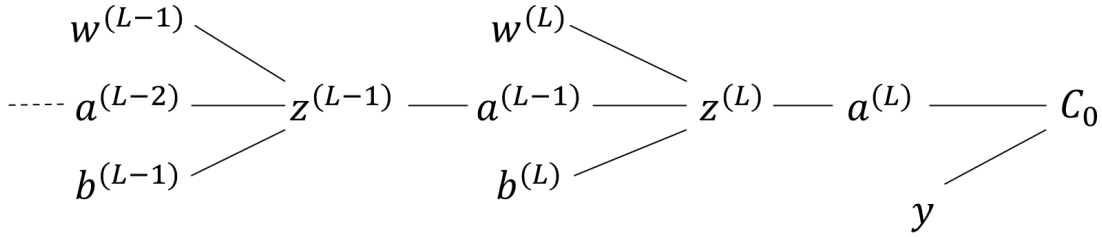
where y is the desired output, and $a^{(L)}$ is the activation of the neuron at the last layer L. The activation is:

$$a^{(L)} = \sigma(z^{(L)})$$

$\sigma()$ is the activation function and $z^{(L)} = w^{(L)}a^{(L-1)} + b^{(L)}$
where $w^{(L)}$ is the weight, $a^{(L-1)}$ is the previous activation and $b^{(L)}$ is the bias.



(a) Visualization of the hierarchy to compute C_0 in which $w^{(L)}, a^{(L-1)}, b^{(L)}$ are used to compute the value $z^{(L)}$ which thorough the activation function cumpute $a^{(L)}$ which along the constant desired output value y compute the cost.



(b) Generalized hierarchy showing the dependence of C_0 on previous weights and biases.

Figure 2.4: Hierarchy for computing C_0

The sensitivity of the cost function C_0 depends to a change in $w^{(L)}$, $a^{(L-1)}$, $b^{(L)}$ (Fig. 2.4a), mathematically are represented here:
Sensitivity to the weight

$$\begin{aligned}\frac{\partial C_0}{\partial w^{(L)}} &= \frac{\partial z^{(L)}}{\partial w^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial C_0}{\partial a^{(L)}} \\ &= a^{(L-1)}\sigma'(z^{(L)})2(a^{(L)} - y) \\ &= a^{(L-1)}\delta^L\end{aligned}$$

Sensitivity to the bias

$$\begin{aligned}\frac{\partial C_0}{\partial b^{(L)}} &= \frac{\partial z^{(L)}}{\partial b^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial C_0}{\partial a^{(L)}} \\ &= \sigma'(z^{(L)})2(a^{(L)} - y) \\ &= \delta^L\end{aligned}$$

Given those last two equation it is important to notice that the cost function is directly proportional to the value of the activation in the previous layer.

Sensitivity to the activation of the previous layer

$$\begin{aligned}\frac{\partial C_0}{\partial a^{(L-1)}} &= \frac{\partial z^{(L)}}{\partial a^{(L-1)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial C_0}{\partial a^{(L)}} \\ &= w^{(L)}\sigma'(z^{(L)})2(a^{(L)}-y) \\ &= w^{(L)}\delta^L\end{aligned}$$

δ^L is the error in the output neuron.

This equation allows to implement the idea of backpropagation, just iterating the same chain rule idea backwards as shown in Fig.2.4b to calculate how the cost function is sensitive to previous weights and biases.

Those equations give the value of the sensitivities of a single neuron of a single layer only for a single training example. The component needed to compute the gradient derives from the averaging of all the n training examples:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial C_k}{\partial w^{(L)}}$$
$$\frac{\partial C}{\partial b^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial C_k}{\partial b^{(L)}}$$

The gradient needs this operation to be computed for each couple w, b at each neuron.

$$\nabla Loss = \left\{\frac{\partial C}{\partial w^{(1)}}, \frac{\partial C}{\partial b^{(1)}}, ..., \frac{\partial C}{\partial w^{(L)}}, \frac{\partial C}{\partial b^{(L)}}\right\}$$

The cost function requires averaging the sensitivities values over all the training examples for each neuron of each layer of the neural network architecture.
When the neural network is generalized (Fig. **??**) with multiple layer and multiple neurons the equations become:

$$\begin{aligned}C_0 &= \sum_{j=0}^{n_L-1}(a_j^{(L)}-y_j)^2 \\ a_j^{(L)} &= \sigma(z_j^{(L)}) \\ z_j^{(L)} &= \sum_{k=0}^{n_L-1} w_{jk}^{(L)}a_k^{(L-1)} + b_j^{(L)}\end{aligned}$$

The backpropagation in form of an algorithm becomes:

1. Input: it sets the corresponding activations a_i^1 for each i-th neuron in the input layer.
2. Feedforward: for each layer $l = 2,3,...,L$ computes $a_i^l = \sigma(z_i^l)$ thus calculating $z_i^l = w_i^l a_i^{l-1} + b_i^l$.
3. Output error: Compute the values
$$\delta_i^L = \sigma'(z^{(L)})2(a^{(L)} - y)$$
4. Backpropagates the error: for each $l = L-1, L-2, ...,2$ computes
$$\delta_i^l = w_i^{l+1}\delta_i^{l+1}\sigma'(z_i^l)$$
5. Output: The gradient of the cost function is given by
$$\frac{\partial C}{\partial w_{jk}^{(l)}} = a_k^{(l-1)}\delta_j^l$$
$$\frac{\partial C}{\partial b_{jk}^{(l)}} = \delta_j^l$$

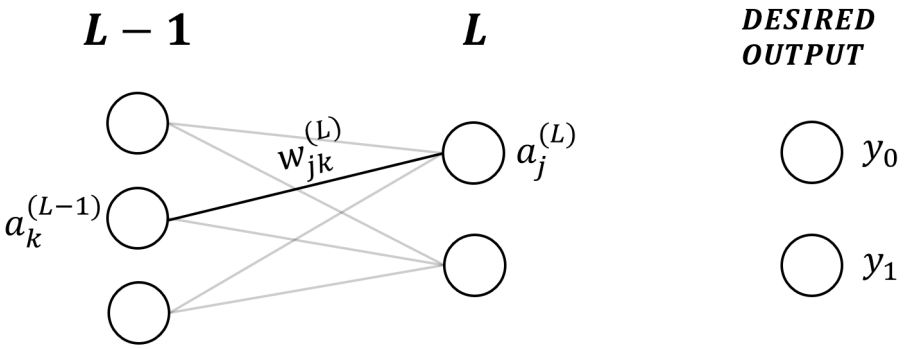


Figure 2.5: Generalized neural network with multiple neurons for each layer.

The hyper-parameters used to maximize the performance are listed here:

Number of hidden layers: There are no rules that establish the number to use. Already with two layers it is possible to represent functions of each shapes, generally no more than four layers are used [10].

Number of hidden units: As in the case of hidden layers there are no precise rules. To understand the general behavior of the network three rule of thumbs are used, from which the optimal number of neurons can then be found. Those rules describe the total number of neuron n_{TOT} in the neural network:

- It is used the sum of the size of the input layer and the size of the output layer $n_{TOT} = n_{feature} + n_{output}$
- It is used $\frac{2}{3}$ the sum of the size of the input layer and the size of the output layer $n_{TOT} = \frac{2}{3}(n_{feature} + n_{output})$
- It is used less than twice the size of the input layer $n_{TOT} < 2n_{feature}$

The choice of these two hyper-parameters is very important as it will decide the architecture of the network. Although they do not directly interact with the external data, they have a strong influence on the final prediction.
Using too few layers and neurons, underfitting occurs, i.e. complicated signals from the dataset may not be detected. On the contrary, using too many of them would create two problems: overfitting when there are too many model parameters compared to the number of data available to train the network, which results in poor performance with data outside the training set; when instead the number of data is sufficient to avoid overfitting, the training time can become too long for practical purposes.

Activation functions: is a non-linear function that allows the neural network to learn complex patterns from data [5]. The most important are shown in Fig 2.6:

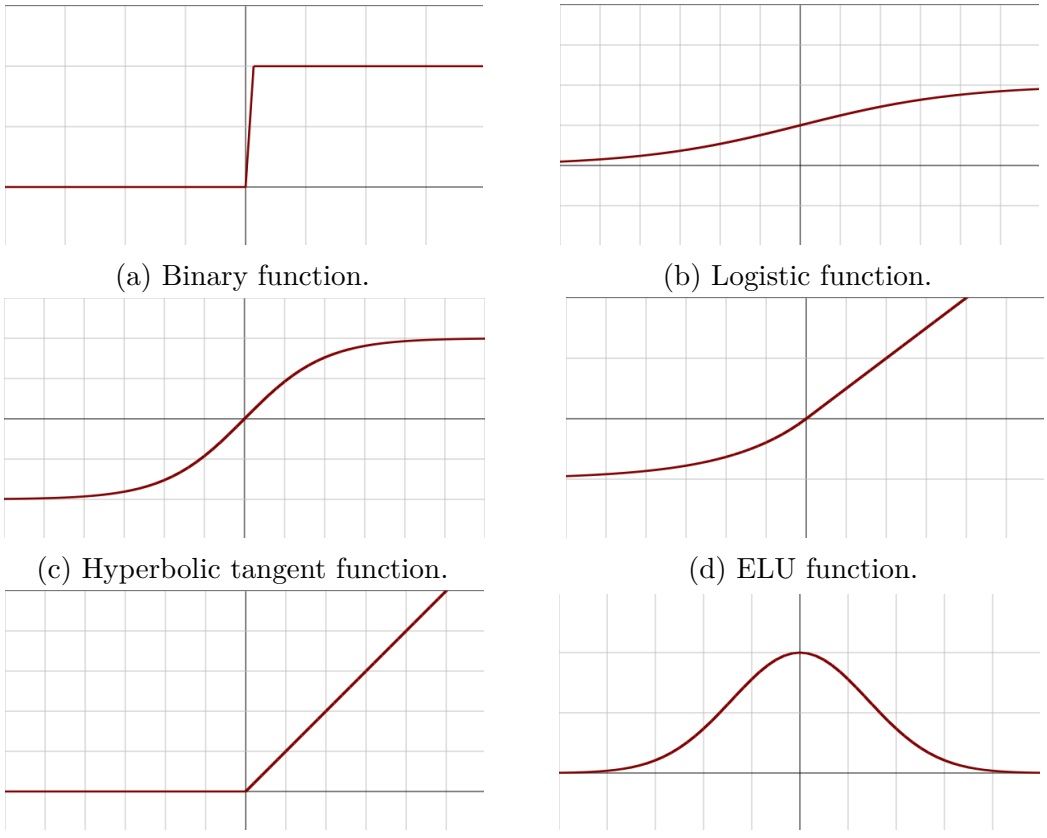
Binary	$g(x) = \begin{cases} 0 \text{ if x}<0 \\ 1 \text{ if x}\geqslant 0 \end{cases}$
Logistic	$g(x) = \frac{1}{1+e^{-x}}$
Hyperbolic tangent	$g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Exponential linear unit (ELU)	$g(x) = \begin{cases} \alpha(e^{x}-1) \text{ if } x > 0 \\ \text{x if x} \leqslant 0 \end{cases}$
Rectified linear unit (ReLU)	$g(x) = \begin{cases} \text{x if x}>0 \\ 0 \text{ if x} \leqslant 0 \end{cases}$
Scaled exponential linear unit (SELU)	$g(x) = \lambda \begin{cases} \alpha(e^{x}-1) \text{ if } x > 0 \\ \text{x if x} \leqslant 0 \end{cases}$
Gaussian	$g(x) = e^{-x^2}$

Learning rate: it determines the step size at each iteration while moving toward a minimum of a loss function.

Regularization rate: it is the value that weighs the importance of the regularization function. Higher values reduce overfitting but make the model less accurate [11].

Drop out rate: is a technique used during the training phase for the regularization of the model. during a step for the calculation of the gradient a random group of neurons is deactivated, this forces the other nodes to probabilistically take on more or less responsibility for the inputs and to learn a sparse representation of the data.

Batch size: it is the number of samples that the network sees at each iteration for the calculation of the gradient. In the case of SGD it is of a single element, while for mini-batch SGD they are between 10 and 1000. The value is fixed during training and inference even if there is also the possibility of having dynamic batch sizes. This feature might be useful when batch size is unknown beforehand, and using extra large batch size is undesired or impossible due to resource limitations.



(a) Binary function. (b) Logistic function.

(c) Hyperbolic tangent function. (d) ELU function.

(e) ReLU function. (f) Gaussian function.

Figure 2.6: Shape of the activation functions.

2.3 Data Preprocessing

In ML it is essential that the data that train the network are significant. This means that they must not have mislabeled values, the values are scaled to fit the model, those that are disproportionate are eliminated, and that the training sets represent the data that the network will have to predict. By following these rules a network learns faster and has accurate predictions.

Data scaling

Feature scaling is a technique that brings all data to have the same range of values, so as not to train the network with disproportionate data. This makes the gradient descent converge more quickly [12]. It also avoids that a value exceeds the floating-point precision limit during training and becomes NaN. It helps the model to learn appropriate weights for each feature.
There are several scaling techniques:

Scaling to range: it means converting feature values from their natural range to a standard range, usually [0,1] or [−1,1].

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x' are the scaled input values, and x are the input value.
It is helpful when the upper and lower bounds are known and the values are uniformly distributed across that range.

Logarithmic scaling: it is used to compress a wide range into a narrow one.

$$x' = log(x)$$

It is used when features have a power law distribution in which handful of values have many points, while most other values have few points.

Clipping: It Caps all feature values above or below a certain value to fixed value. it is used when the dataset contains very few values that are extremely out of range (more distant than 3 standard deviations from the mean).

Bining: It Divides the range into boolean features, each bin contains a narrow range of values.

Standardization: used when the dataset has a limited amount of out-of-range values [6].

$$x' = \frac{x - \mu}{\sigma}$$

μ and σ are the mean and standard deviation of the feature, respectively.

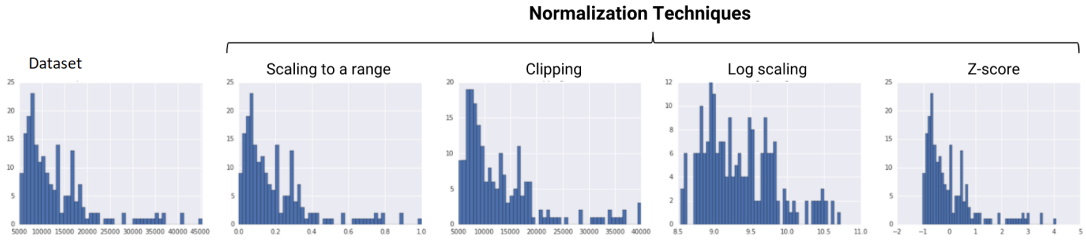


Figure 2.7: Example of data cleaning over a dataset [13].

2.4 Methods for Training ML Models

The ultimate goal of the ML algorithm is to predict the output with the greatest confidence. To do this, the trained network must be tested on a sub-set of the main dataset in order to verify and possibly change the hyper-parameters, otherwise the observed results would not be representative of the model's response to new data. The division of the dataset into sub-sets each with different purposes implies an increase in the complexity of test and training, the most significant are described here.

Held-out test set

Figure 2.8: Held-out test set schema. The whole dataset is divided in train and test sets.

To train and evaluate a model, the dataset is divided into two smaller sets, one for training and one for testing.
Before being divided they are randomly mixed to avoid having unrepresentative sub-sets. The larger the training set the better the model trained, the larger the test set the better the confidence in evaluation metrics that have a more precise confidence interval.
The percentage into which the sub-sets are divided depends on the amount of data that the dataset is formed from. With datasets of millions of data, even a 10% test set has a valid confidence interval, while with small datasets there is a risk of overfitting.
Other techniques such as held-out validation and test sets or cross-validation are used to mitigate this problem.

Held-out validation and test sets

Figure 2.9: Held-out validation and test sets schema. The training set is sub-divided into train and validation sets.

The dataset is split into three different sets: training, validation and test dataset. The hyper-parameters are optimized on the training set, the model is tested on the union of the training and validation set, and finally the performance on the test set is evaluated. The validation set is created from the test set, thus reducing the test set number of training samples.

K-fold Cross-Validation

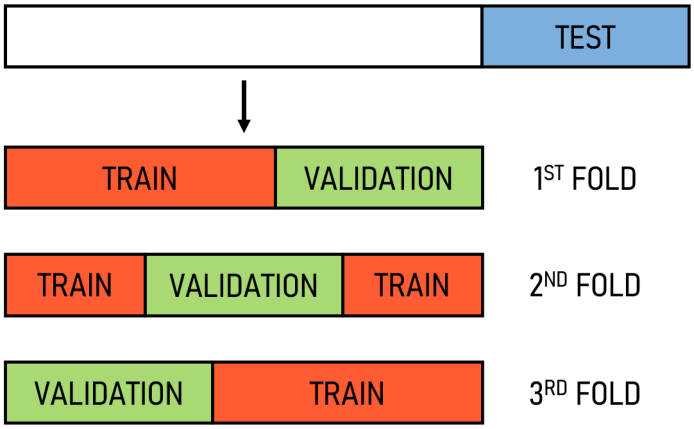


Figure 2.10: K-fold Cross-Validation schema with $k = 3$.

The training set is divided into k sub-sets, k-1 are used as training set for hyper-parameters tuning and the last sub-set is used as validation set set to control performance evaluation.
After k iterations the average of the models obtained is extracted, which have a low variance estimation.
To strengthen the result of the hyper-parameters obtained, the value of the k factor is increased, with standard values from $k = 3$ to $k = 10$ [14].

Nested k-fold Cross-Validation

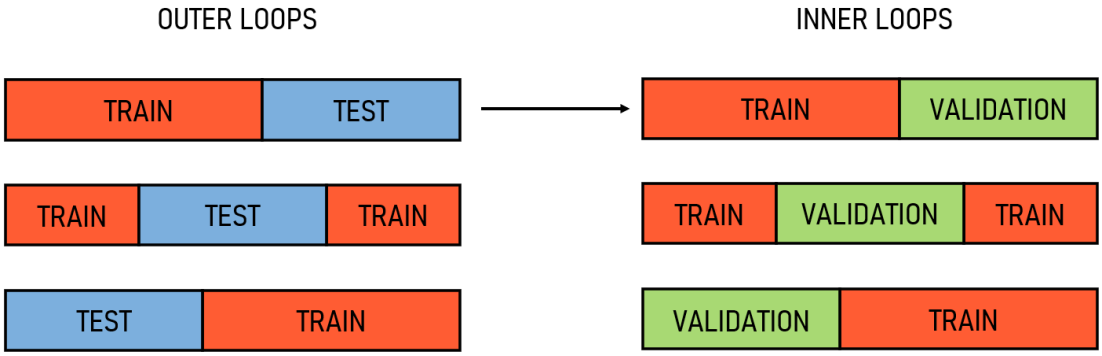


Figure 2.11: Nested k-fold Cross-Validation schema with $k_{outer} = 3$ and $k_{inner} = 3$. It is shown the first iteration out of three of the outer loop and the three iterations of the inner loop, in which the training set are sub-divided in three sub-set.

This solution is the best solution in term of robustness of the model. The performance measurements of the best found models have a low generalization error [10] [6] on the test set because it is distributed on the whole dataset.
The k_{inner}-fold cross-validation procedure for model hyper-parameter optimization is nested inside the k_{outer}-fold cross-validation procedure for model selection. An outer loop is applied k_{outer} times, the dataset is divided in training and test set. For each k_{outer} times an inner loop is applied k_{inner} times to the training set as in the k-fold Cross-Validation.
This allows to have from the inner loop a model with the best hyper-parameter evaluated on the average of the best model found against the validation set splits. Each model found in the inner loop is saved, and if the same as the previous one it is overwritten. So from the outer loop there will be at most k_{outer} model, among which the most robust is chosen.
A downside of this technique is the computational time used, which is $k_{inner} \times k_{outer}$, so it is used for fast training models or with few hyper-parameters [15].

2.5 Hyper-parameters Tuning

The Hyper-parameters cannot be estimated by the model from the given data, are used to estimate the model parameters like the weights, which instead are extrapolated from the data.
A correct Hyper-parameter Tuning allows you to maximize model performance, as well as being the only way to extract maximum performance from a model.

There are two ways of setting the hyper-parameters. manual tuning, in which a few hyper-parameters are adjusted to understand a sensible workspace in which they

have a significant effect on the model's performance, and the automated tuning which is used in case the hyper-parameters are too many to use the manual tuning, therefore the hyper-parameters are chosen by an algorithm that optimizes their selection process.

The most used algorithms are:

Grid search

A grid is created for each hyper-parameter, the distribution of values chosen is linear equi-spaced or logarithmic.
The algorithm fits the model on each and every combination of hyper-parameter possible and records the model performance, and finally they are compared. Once the optimal values have been found, the whole process can be repeated using a grid with denser values around those found promising.
Definitely better than manual search, but for neural networks that require many hyper-parameters the search space becomes unmanageable and the times too long.

Random search

Instead of fitting every possible grid value, the chosen value are sampled from the grid with a random distribution.
Since the number to be sampled are chosen from the grid the increase of values in the hyper-parameters grid doesn't decrease the efficiency of the algorithm [6].

Bayesian search

It is best-suited for optimization over continuous domains of less than 20 dimensions [16] [17].
Its structure consists of two main components [18]: a Bayesian statistical model for modeling the objective function (the target function to minimize), and an acquisition function for deciding where to sample next.
The algorithm builds a surrogate (a probability model) for the objective and quantifies the uncertainty in that surrogate using a Bayesian machine learning technique, the Gaussian process regression, and then uses an acquisition function defined from this surrogate to decide where to sample.
After evaluating the objective according to an initial space-filling experimental design, often consisting of points chosen uniformly at random, they are used iteratively to allocate the remainder of a budget of N function evaluations.
So it can be summarized in few key points:

1. Building a surrogate probability model of the objective function f
2. Observing the results $y_n = f(x_{n0})$ using the initial space filling design.
3. Updating of the posterior distribution $f(x_n)$.

4. Researching the hyper-parameters x_n that maximize the acquisition function. The acquisition function measures the value that would be generated by evaluation of the objective function at a new point x, based on the current posterior distribution over f with much less time expenses.
5. Observing the results $y_n = f(x_n)$ of the objective function to the hyper-parameters found previously.
6. Incrementing n and iterating from 3) until the budget is reached.

2.6 Performance Evaluation Metrics

It was explained how to process the data, how to divide them for a correct training and how to tuning the hyper-parameters of the model. The metrics that allow an evaluation of the model's performance are now explained.

In binary classifiers is defined a classification threshold (also called the decision threshold), it is the value beyond which a prediction falls into the positive category, otherwise the prediction is categorized as negative. Typically it is set to 50% but it is problem-dependent, therefore it is a values that must be tuned to maximize the performance.
The prediction can fall into positive or negative categories, but the data can have two labels: true or false.
This allows to evaluate different metrics considering the following definitions:

- A true positive (TP) is an outcome where the model correctly predicts the positive class.
- A true negative (TN) is an outcome where the model correctly predicts the negative class.
- A false positive (FP) is an outcome where the model incorrectly predicts the positive class.
- A false negative (FN) is an outcome where the model incorrectly predicts the negative class.

Accuracy

It can be now define the accuracy as the ratio of number of correct predictions to the total number of input samples.

$$accuracy = \frac{num_{totright}}{num_{tot}} = \frac{TP + TN}{TP + TN + FP + FN}$$

In the event of an unbalanced class dataset, where the number of positive and negative labels are significantly unbalanced, the number of FN or FP would not be evaluated properly. For this reason precision and recall are defined.

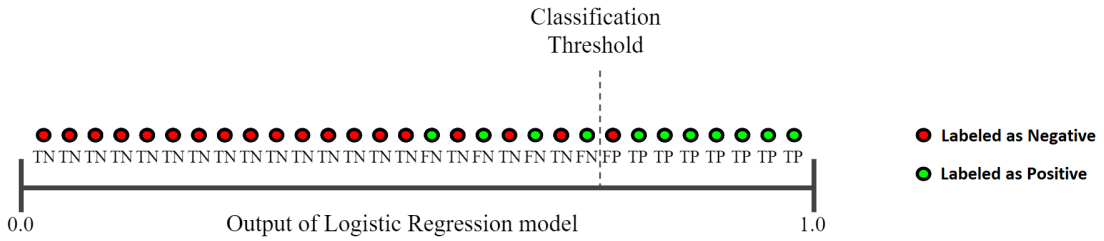
Precision and Recall

The precision defines the proportion of positive identifications that are actually correct, the higher the precision, the lower the FP rate.

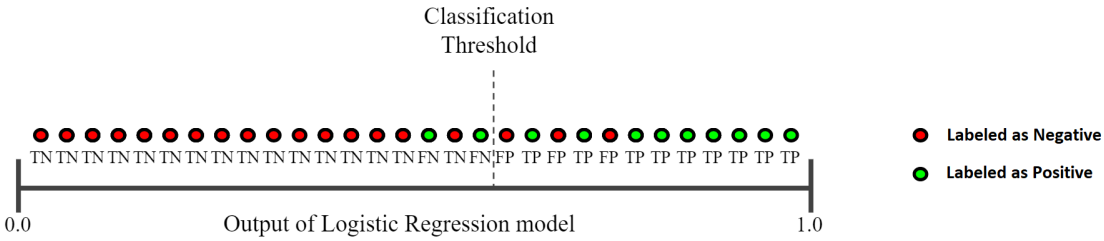
$$precision = \frac{TP}{TP + FP}$$

On the other hand the recall is defined as the proportion of actual positives that are identified correctly, the higher the recall, the lower the FN rate.

$$recall = \frac{TP}{TP + FN}$$



(a) It is shown how a high threshold value leads to decreasing the cases of FP but increasing the FN, thus increasing *precision* and decreasing *recall*.



(b) Here the decrease of the threshold decreases the cases of FN increasing those of FP, therefore decreasing the *precision* and increasing the *recall*

Figure 2.12: Two cases showing the effects of the variation of threshold on *precision* and *recall*.

Varying the threshold will vary the precision and recall values, which are often in tension. It is easy to observe on Figure 2.12a that using a high threshold value

increase the precision lowering the recall. Vice versa on Figure 2.12b it is shown how a low threshold increase the recall lowering the precision.

The threshold is chosen depending on the type of system on which the model will work.
In the case of this thesis, a FP represents a jar eliminated from the production line effectively without contaminant, which represents an economic loss; while an FN represents a jar containing contaminant which has not been detected and which has potentially entered the sales cycle, which would create the disadvantages seen in Chapter 1.1. for this reason, greater importance has been given to the detection of FN using recall as the main metric.

ROC and AUC

The ROC curve, also called receiver operating characteristic curve, is a graph showing the performance of the model at all threshold values. the True Positive Rate (TPR), synonymous with recall, and the False Positive Rate (FPR) are plotted, which is nothing more than the complementary of the precision.

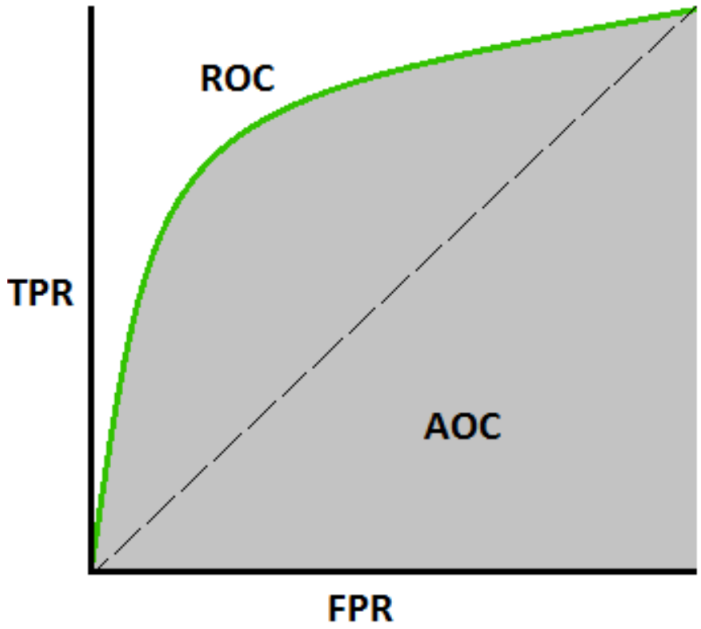


Figure 2.13: On the axis the parameters TPR and FPR that define the ROC function on the interval [0,1][0,1].

A good model tends to be towards the point (0,1) where the FPR is minimum (maximum precision) and recall is maximum.

A metric obtained from this graph is the Area under the ROC Curve (AUC), it

is the integral of the function that starts from the point (0,0) and arrives at the point (1,1).
The AUC gives the measure of the performance obtained along every possible threshold, it can be seen as the probability that the model ranks a random positive example more highly than a random negative example.
The AUC has two main property [13]:

- It is scale-invariant: it measure the general ranking of the prediction, not their absolute value.
- It is classification-threshold-invariant: if measure the quality of model not in a specific threshold value.

for those reason it is a good indicator of the model general behaviour, but in this thesis case it is important to give more importance to the recall metric since it is critical to minimize the presence of FN.

Confusion Matrix

A good representation of the model outputs is the confusion matrix in which are represented the four possible outcome.
in the main diagonal are placed the correct prediction TP and TN, the other places are dedicated to the wrong predictions FP and FN.

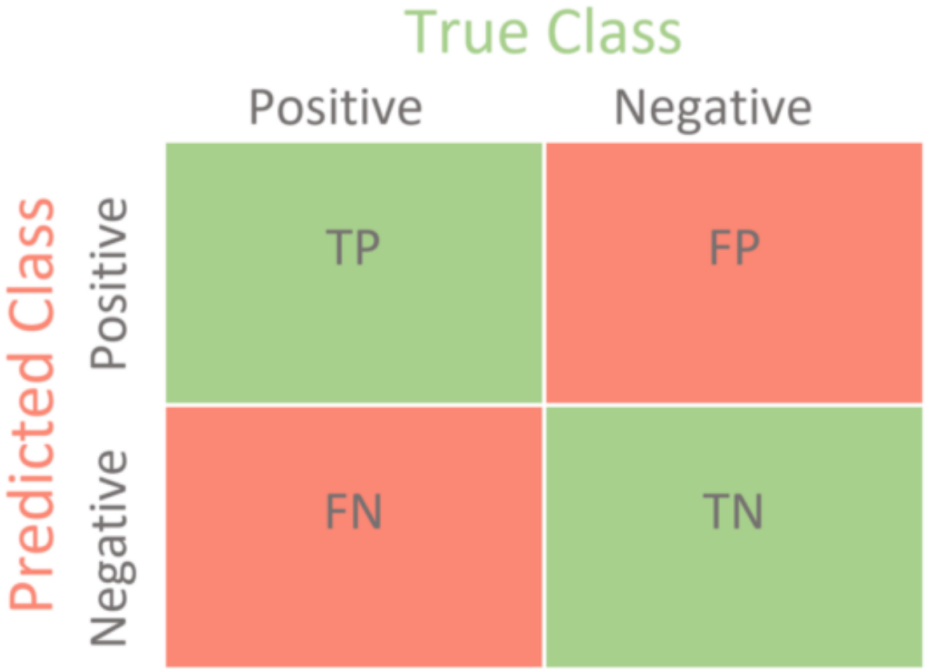


Figure 2.14: Representation of a confusion matrix and its parameters.

Chapter 3

Machine Learning Training And Testing

3.1 Dataset acquisition

This chapter explains how to acquire a dataset with the MIT-Food prototype measurement system, and the specifications of the components of this system.
The MIT-Food prototype measurement system can be used to generate tomographic images, but for the purpose of this thesis the data acquired by the antennas are used without reconstructing an image, and are divided into two classes "contaminated" and "not contaminated" to train the ML algorithm. This means that the system, although it can be used for microwave imaging, is instead defined as a MWS system.

The characteristics of the components in the MIT-Food prototype measurement system are now described:

Antennas

The six antennas are placed in a resin support and they are positioned mirrored, the top one is inclined by 60°, the middle one by 15° and the bottom one by 0°. This architecture allows two things: to cover the entire surface of the jar on the conveyor belt and to be able to install it in an in-line detection system. The antenna PCB size is $3cm$x$4cm$, the fact that they are made of PCB make them very cheap and easy to produce.
The antennas are designed to radiate at a frequency of $10GHz$, due to the manual connections and soldering they radiate in the range from $10GHz$ to $10.6GHz$. This frequency was chosen in order to optimize the penetration depth and resolution of the signal [3] [19] with respect to the dielectric properties of the hazelnut cream.

Vector Network Analyzer

The VNA model is the Keysight M9037A PXI, it is used to acquires the measurements from each of the 6 antennas which are directly connected to its ports. Thanks to the PXI trigger port, the measurements and the passage of the jar under the antenna arc are synchronized.

The dataset acquired in the laboratory is composed of 5455 samples of scattering parameters of the antennas measurements: 1702 uncontaminated and 3753 contaminated jars. The material in the jar used to acquire the measurements is safflower oil instead of the hazelnut cream, because being translucent it allows to control the position of the contaminant. The safflower oil has the same dieletric properties of hazelnut cream, in fact $\epsilon_{safflower\ oil} \simeq \epsilon_{hazelnut\ cream} = 2.86$ @ $10GHz$. The speed of the conveyor belt is $30\,\mathrm{cm\,s^{-1}}$. The "contaminated" class contains several materials as intruders:

- A PTFE sphere
- A SLG sphere
- A Nylon sphere

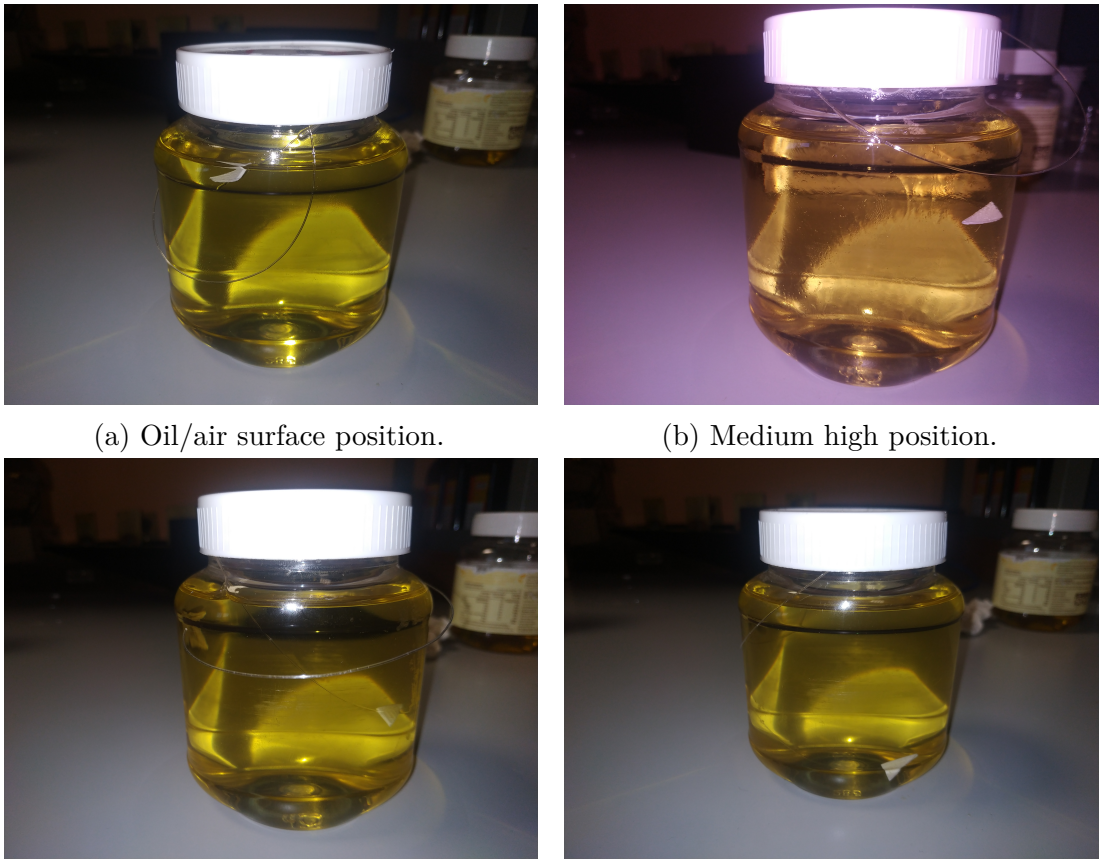
All of them have a 2 mm radius. They were placed at the bottom of the jar at random distance from the jar edge at each measurements acquisition.

- A piece of wood
- Two plastic splinter of the jar cap
- A plastic splinter 3D printed

The wood and plastic splinter contaminants of approximately of dimension $1\,\mathrm{cm}^2$ where placed at random distance from the edge and at four different heights: at the bottom, in the middle, at the top submerged by at least 1 cm of safflower oil, floating on the air-oil interface.

The following steps have been performed for each category:

1. **Contaminant preparation**: The contaminant glued to a nylon thread, so as to be positioned at the predetermined height.
2. **Contaminant placement**: Once the contaminant is positioned at the required height, the cap is closed, and by friction with the nylon thread it remains in position. Every 125 measurements the height is changed, in the middle of which the position in the horizontal plane is changed. In the case of the spheres that cannot be glued to a wire, the horizontal position at the



(a) Oil/air surface position.

(b) Medium high position.

(c) Medium low position.

(d) Bottom position.

Figure 3.1: The four height of the contaminant during the acquisition.

bottom of the jar has been changed every 50 measurements, while for the measurements of the wood it was left to float on the oil surface.

3. **Jar positioning**: The jar is positioned on the conveyor belt, before passing under the arch of antennas its position is mechanically adjusted by the metal rods parallel to the direction of the production line, so as to simulate a production line where the angular deviation of the jar is minimal.

4. **Acquisition**: After the metal bars, the jar passes in front of a photocell, which through a timer circuit signals to the VNA to trigger the measurement the exact moment in which the jar is under the antenna arc. The VNA acquires a scattering matrix for 11 equidistant frequencies in the range starting from $9GHz$ up to $11GHz$. A dataset is obtained from each frequency, and an extra one is created containing the values of all frequencies.

5. **Scattering matrix conversion**: The values of the scattering matrix are converted into samples useful for training the ML algorithm. Since the matrix is reciprocal, only the values of the above diagonal are extracted, excluding those of the diagonal as it has been verified that they lead to a worsening of the results. The real and imaginary parts are extracted from the remaining matrix values and saved in a text file in sequence for a total of 60 values and therefore 60 features, while for the dataset containing all eleven frequencies the features are 660.

The implementation of points 1 and 2 is omitted for the acquisition of the measures for the uncontaminated category.

3.2 MLP Training

The dataset obtained in the laboratory is used to find the optimal hyper-parameters of the model which are then applied to the training of the models for the real and synthetic datasets acquired in the company.
The training was performed for a 60 features model with the $10GHz$ dataset and for a 660 features model with the dataset containing all eleven frequencies. For both cases the hyper-parameters were searched using the Bayesian search with the aim of minimizing the validation loss, and the results of three MLPs with 1, 2 and 4 hidden layers [10] were compared in order to find with which frequecy and which architecture the best MIT-Food system can be obtained.

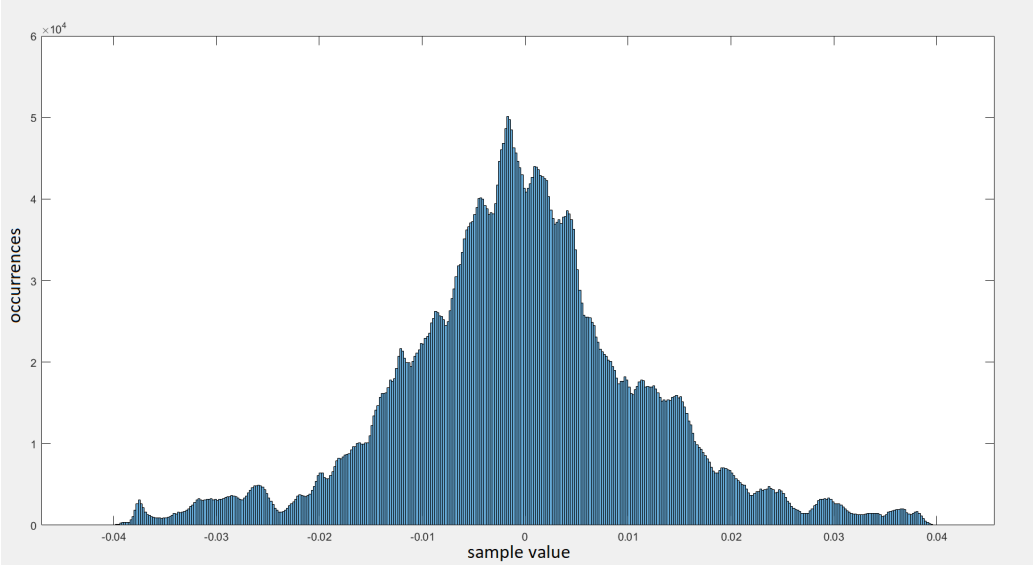
Before training the model each dataset is mixed and divided into training set and test set, respectively 75% and 25%. Standardization is applied to them, which despite the dataset values being in the same order of magnitude brings better performance, it is also noted from figure figure 3.2b that it is possible to distinguish a diversification of the values for the two classes to be predicted in the standardized dataset.

The steps performed to obtain each models are now explained:

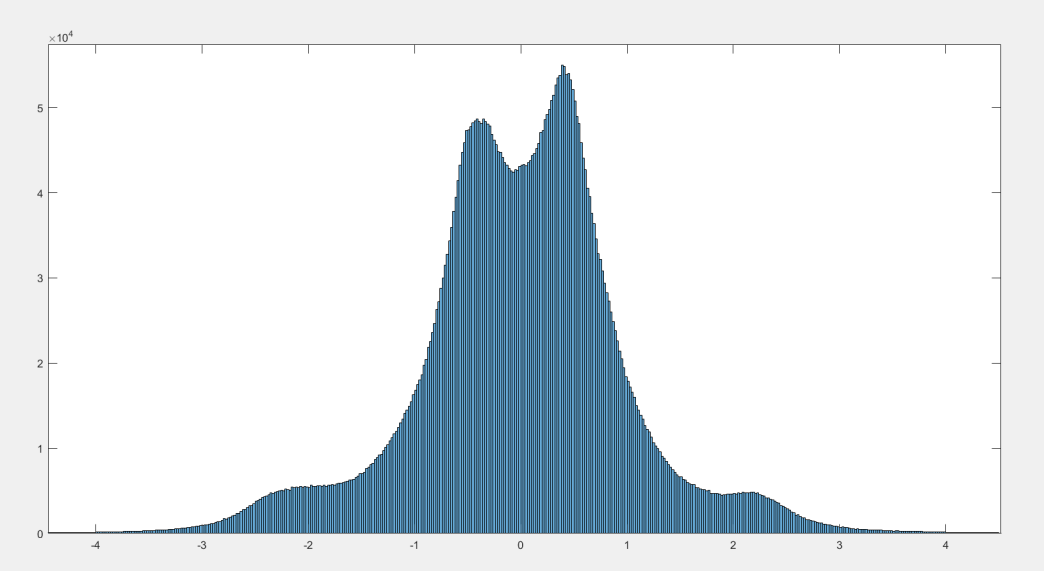
1. **Grid definition**: for each Hyper-Parameter two grids of possible values are defined, a loose one to find the order of magnitude and trends of the model and a fine one to search around the optimal values found by the previous grid. These values are used by the Bayesian search algorithm during training.

2. **Constant hyper-parameter**: constant hyper-parameters are defined, which have fixed value during the training.

3. **Bayesian search**: Bayesian optimization along with 3 fold-CV are performed to find the best hyperparameters which minimize the loss of validation.



(a) Distribution unprocessed, values are in the range [-0.0427, 0.0413].



(b) Distribution preprocessed, values are in the range [-7.8853, 9.3992].

Figure 3.2: Distribution of samples acquired in the laboratory.

4. **MLP training:** the dataset is divided in a 25% test set, a 52% training set and a 23% validation set. The MLPs are trained with the training set and verified with the validation set using the best values found from the fine grid.
5. **MLP testing:** the MLPs classifier with the best hyper-parameters is tested on the test set and all the performance evaluation metrics necessary to compare

it with other solutions are extracted.

Once the best models from the two dataset (10GHz and all eleven frequencies) for each hidden layer is found, the learning rate is added to the fine grid and the best out of all the models is tested with various activation functions: ELU, ReLU, SELU, sigmoid, softmax, softsign, hyperbolic tangent.

Grid type	Number of hidden layer	Number of neurons per hidden layer	Learning rate	Weight regularization parameter	Dropout rate
Loose	[1,2,4]	[1,2,4,8,16,32,64, 128,256,512,1024]	x	[0.005,0.01, 0.03,0.05,0.1]	[0.0,0.1, 0.2,0.3,0.4]
Fine	[1,2,4]	[2,4,8,16,32,64,128]	[0.001,0.003, 0.005,0.01,0.03]	[0.0005,0.001, 0.005,0.01,0.05]	[0.0,0.1, 0.15,0.2,0.3]

Table 3.1: The two types of grids used to search the hyper-parameters of the six MLPs classifier.

These Hyper-parameters are present as constants in the definition of the grids:

- Number of Epochs: 1000.
- Batch size: 256.
- Weight regularization: L2.
- Loss function: binary cross-entropy.
- Optimizer: Adam, in order to have the learning rate out of the searching grid.
- Activation: ReLu.

The Hyper-parameters not constant in the search space are:

- Number of hidden layer.
- Number of neurons per hidden layer.
- Learning rate.
- Weight regularization parameter.
- Dropout rate.

Dataset frequencies	Best number of neurons per hidden layer				Best learning rate	Best weight regularization parameter	Best dropout rate	Best accuracy	Best recall	Best validation loss
	First	Second	Third	Fourth						
$10GHz$	64				0.003	0.005	0	97.48%	98.82%	0.303
	64				0.001	0.005	0.2	99.43%	99.64%	0.117
	512				0.001	0.01	0.3	98.62%	98.58%	0.296
All	32				0.001	0.005	0.1	98.78%	98.82%	0.115
	64				0.001	0.01	0.2	99.22%	99.58%	0.043
	64				0.001	0.005	0.2	99.51%	99.76%	0.041

Table 3.2: The most valuable results from the training with the lab dataset for a MLP with 1 hidden layer.

Dataset frequencies	Best number of neurons per hidden layer				Best learning rate	Best weight regularization parameter	Best dropout rate	Best accuracy	Best recall	Best validation loss
	First	Second	Third	Fourth						
$10GHz$	256	1024			0.001	0.005	0.1	99.35%	99.76%	0.156
	256	1024			0.001	0.01	0.2	98.53%	99.05%	0.198
	512	1024			0.001	0.005	0.2	99.51%	99.76%	0.190
All	64	16			0.001	0.005	0.2	99.27%	99.64%	0.092
	64	**4**			**0.001**	**0.005**	**0.1**	**99.76%**	**99.88%**	**0.040**
	256	128			0.001	0.01	0	97.56%	98.58%	0.110

Table 3.3: The most valuable results from the training with the lab dataset for a MLP with 2 hidden layer.

Dataset frequencies	Best number of neurons per hidden layer				Best learning rate	Best weight regularization parameter	Best dropout rate	Best accuracy	Best recall	Best validation loss
	First	Second	Third	Fourth						
$10GHz$	64	16	4	256	0.001	0.005	0.1	99.19%	99.05%	0.135
	64	1024	512	256	0.001	0.005	0.2	99.27%	99.29%	0.143
	64	2	128	1024	0.003	0.005	0.2	98.45%	98.46%	0.173
All	8	4	1024	1024	0.001	0.005	0	99.84%	99.17%	0.121
	512	256	512	256	0.003	0.005	0	99.84%	99.76%	0.110
	8	128	64	4	0.003	0.005	0	99.51%	99.12%	0.192

Table 3.4: The most valuable results from the training with the lab dataset for a MLP with 4 hidden layer.

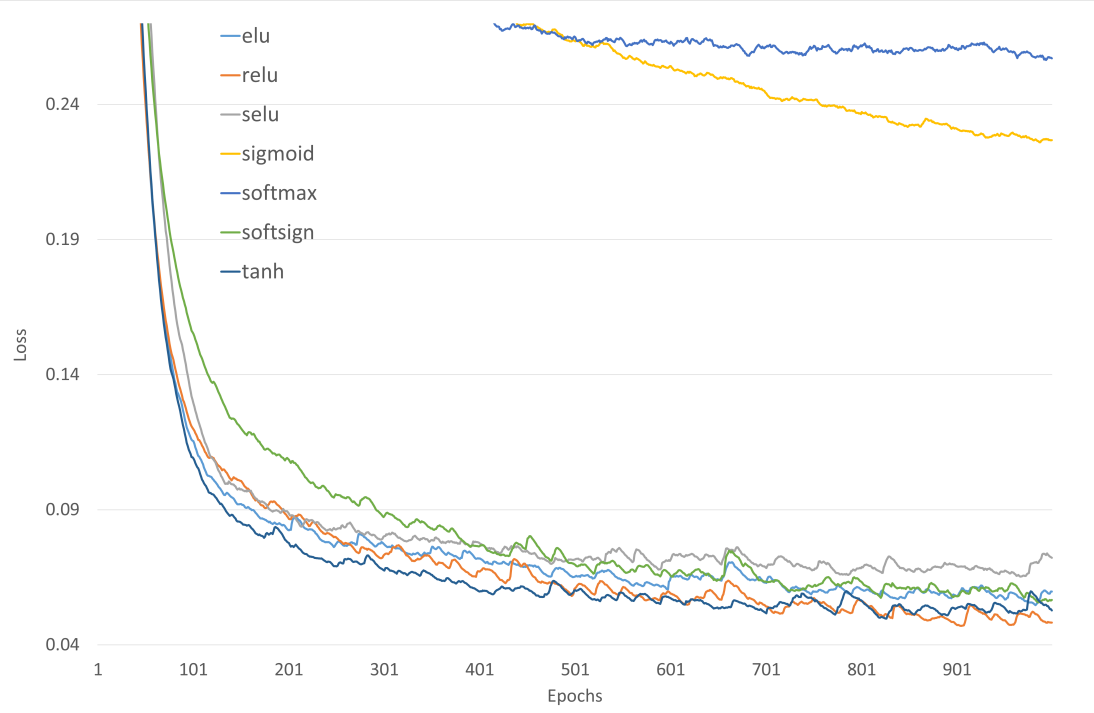


Figure 3.3: Loss performance of the activation functions applied to the MLP selected by the Bayesian search.

The tables 3.2, 3.3 and 3.4, respectively, represent the best results from MLP training with 1, 2 and 4 hidden layers. Among all these the model selected is the one with two hidden layers, the first with 64 neurons and the second with 4, tranined with the dataset with the samples from all the eleven frequencies. Its performance is the best as it has the lowest validation loss value of 0.040, the highest recall value of 99.88% and the third highest accuracy value of 99.76%.
This model is trained with different activation functions to verify which one is best. From the results shown in the figures 3.3, 3.4 and 3.5 it is clear that the best activation function is ReLU, as it has a lower loss, a better accuracy, and a recall comparable with the values of hyperbolic tangent and softsign.

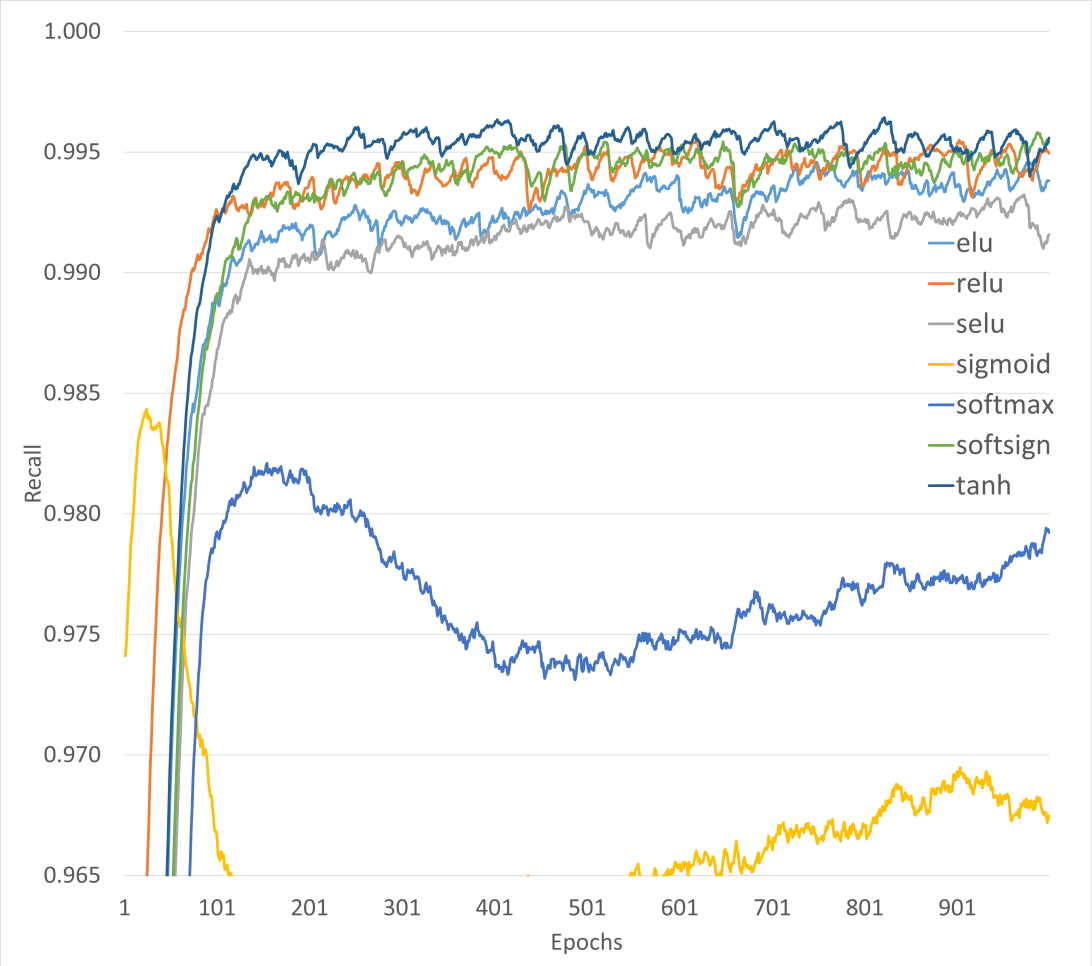


Figure 3.4: Recall performance of the activation functions applied to the MLP selected by the Bayesian search.

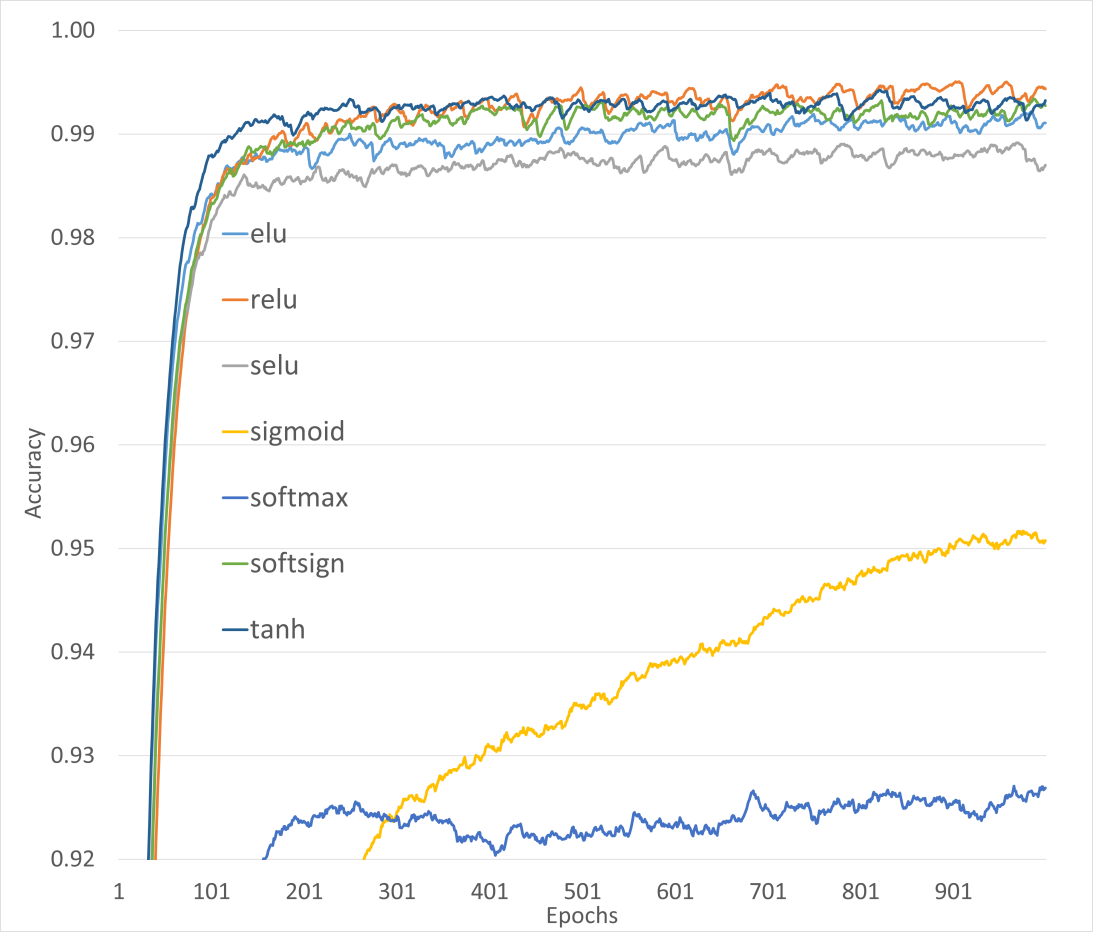


Figure 3.5: Accuracy performance of the activation functions applied to the MLP selected by the Bayesian search.

3.3 MPL Testing and Performance Evaluation

The performance referred to the test set of the selected MLP with 2 hidden layers (64, 4) and Relu Units are given with the following plots.
The confusion matrix shows how selecting a threshold value of 25% decreases the FP cases while remaining unchanged the cases of FN compared to a threshold of 50%.

10GHz th=50%	**True Class Positive**	**True Class Negative**
Predicted Class Positive	419	7
Predicted Class Negative	2	936

10GHz th=25%	**True Class Positive**	**True Class Negative**
Predicted Class Positive	421	5
Predicted Class Negative	3	935

All Freq. th=50%	**True Class Positive**	**True Class Negative**
Predicted Class Positive	938	5
Predicted Class Negative	0	421

All Freq. th=25%	**True Class Positive**	**True Class Negative**
Predicted Class Positive	938	1
Predicted Class Negative	0	425

Table 3.5: Comparison between confusion matrix with 50% and 25% threshold for both 10GHz and all frequencies datasets.

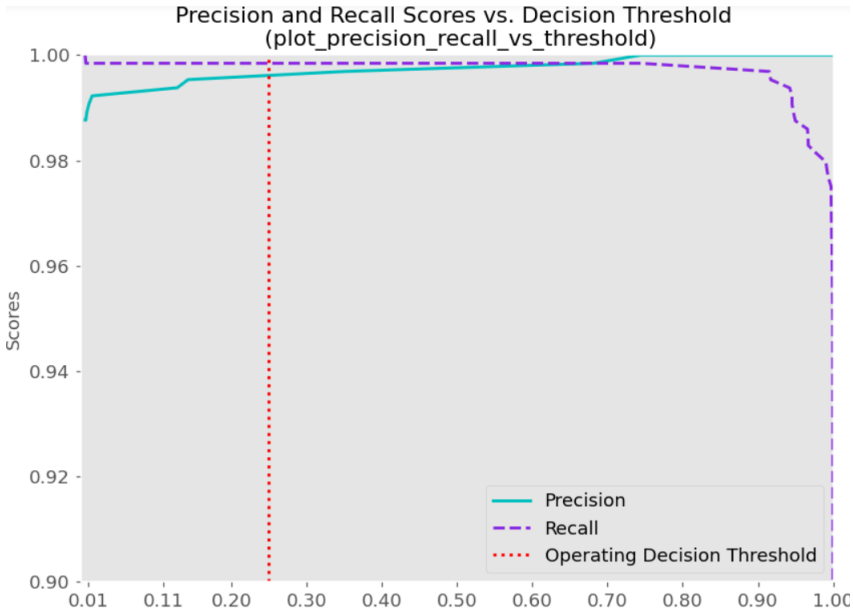


Figure 3.6: Precision and recall scores vs. decision threshold of the best found MLP with 2 hidden layers (64,4) and Relu Units, calculated with the test set with samples from all the frequencies.

In figure 3.6 precision and recall scores are plotted versus the decision threshold of the MLP. The decision threshold at 25% improves recall, slightly worsening the accuracy. While in the figure 2.13 the ROC curve is shown, highlighting the threshold point. The operating point of this MLP allows the AUC to be high at a value of 0.998.

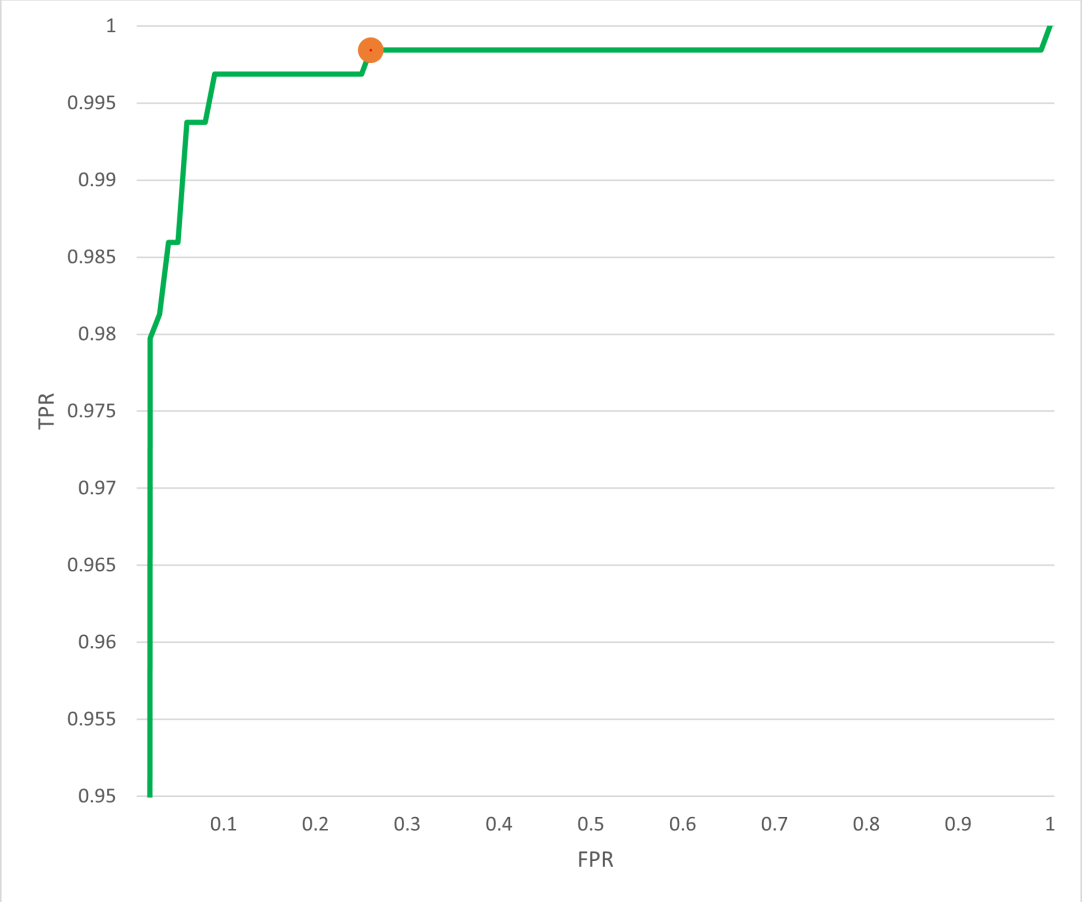


Figure 3.7: ROC of the best found MLP with 2 hidden layers (64,4) and Relu Units, calculated with the test set with samples from all the frequencies.

The model found in the previous paragraph 3.2 with the same threshold value and activation functions is used to train and test also the datasets acquired in the company (from 2 up to 7 of table 1.1), each of which is compared in their two acquisitions datasets, the 10GHz and the one with all frequencies.

The Confusion matrix of the MLPs are in the tables 3.6 and 3.7, which undoubtedly show how network training using all eleven frequencies leads to better results than

using the single $10GHz$ frequency.

$10GHz$

Safflower oil $30cms^{-1}$	**True Class Positive**	**True Class Negative**
Predicted Class Positive	250	0
Predicted Class Negative	1	499

Safflower oil $50cms^{-1}$	**True Class Positive**	**True Class Negative**
Predicted Class Positive	739	11
Predicted Class Negative	1	450

Hazelnut Cream $30cms^{-1}$	**True Class Positive**	**True Class Negative**
Predicted Class Positive	925	6
Predicted Class Negative	5	308

Hazelnut Cream $50cms^{-1}$	**True Class Positive**	**True Class Negative**
Predicted Class Positive	750	0
Predicted Class Negative	0	399

Safflower oil both speeds	**True Class Positive**	**True Class Negative**
Predicted Class Positive	989	11
Predicted Class Negative	4	947

Hazelnut Cream both speeds	**True Class Positive**	**True Class Negative**
Predicted Class Positive	1678	2
Predicted Class Negative	3	709

Table 3.6: Comparison between confusion matrix of MLPs trained with the safflower oil and hazelnut cream at two speeds datasets with the 10GHz sample.

All Frequencies

Safflower oil $30cms^{-1}$	**True Class Positive**	**True Class Negative**	**Safflower oil** $50cms^{-1}$	**True Class Positive**	**True Class Negative**
Predicted Class Positive	247	3	**Predicted Class Positive**	746	4
Predicted Class Negative	0	500	**Predicted Class Negative**	0	451

Hazelnut Cream $30cms^{-1}$	**True Class Positive**	**True Class Negative**	**Hazelnut Cream** $50cms^{-1}$	**True Class Positive**	**True Class Negative**
Predicted Class Positive	930	1	**Predicted Class Positive**	750	0
Predicted Class Negative	0	313	**Predicted Class Negative**	0	399

Safflower oil both speeds	**True Class Positive**	**True Class Negative**	**Hazelnut Cream both speeds**	**True Class Positive**	**True Class Negative**
Predicted Class Positive	998	2	**Predicted Class Positive**	1679	2
Predicted Class Negative	0	951	**Predicted Class Negative**	0	712

Table 3.7: Comparison between confusion matrix of MLPs trained with the safflower oil and hazelnut cream at two speeds datasets with the sample from all frequencies.

To conclude, MLP chosen for the hardware implementation has:

- An input layer with 660 nodes, one for each features of the datasets with the samples from all the eleven frequencies.
- Two hidden layer, the first with 64 nodes and the second with 4.
- A sigmoid output neuron.

Chapter 4

Hardware Implementation

This chapter talks about all the steps involved in the hardware implementation of the best MLP discovered in Chapter 3, whose architecture is (64, 4), with 660 features. It consist in the Keras model conversion in a synthesizable MATLAB and C code with 3 different level of compression and the physical implementation on two different systems for the estimations of latency.

4.1 Inference Systems

Model inference is performed on two systems. The first is the computer integrated into the VNA, in which a matlab code is used for inference. The VNA acquires the scattering matrix which is processed and sent in input to the model, the result of the prediction is sent to an external microcontroller which signals the result via a led. In this case the system is responsible for both data acquisition and their inference, so a lower latency is expected from when the acquisition takes place until the LED signals the result.
The second system in which the inference occurs is the Nucleo-F401RE microcontroller, on which a C code is used. The microcontroller receives the acquisitions of the antenna arc from the VNA, preprocesses the data and, as in the other system, the prediction it is signaled by the lighting of a LED in case a contaminated product is identified while it remains off if the product is recognized as not contaminated. From this system a higher latency is expected not having the possibility to acquire the data directly, but receiving them via UART communication.

4.2 Model Synthesis and Implementation

The model that is used for the inference in the two systems is trained with the dataset acquired in the company containing hazelnut cream with the samples of all eleven frequencies. To fully define an MLP it is necessary to know the type of architecture: the values of the number of neurons per layer, the number of layers, the activation functions of the neurons, the number of input and output features. During the training of the MLP the values of the weights, biases and the architecture were saved in a hierarchical data format file HDF5. Both information are necessary for the conversion of a keras model into synthesizable C or MATLAB code.

VNA System
The HDF5 file is converted via the MATLAB function `importKerasNetwork()` into a MAT file used for the inference in the algorithm explained here:

1. **Initialization**: The model together with the vectors of mean and standard deviation of 660 elements each are loaded.
2. **Data Acquisition**: When the VNA receives the trigger signal, it acquires the scattering matrix of the antennas for each of the eleven frequencies.
3. **Matrix Sampling**: The data from the upper triangular matrices are saved.
4. **Standardization**: Each element of the samples is standardized.

$$feature_i = \frac{sample_i - \mu_i}{\sigma_i}$$

5. **Inference**: The inference of the standardized data is computed, and the value of the sigmoid neuron output is saved.
6. **Output Elaboration**: The prediction is obtained through the threshold set at the value of 0.25, and the result is sent to the external microcontroller.

Points 2 to 6 are iterated for each product.

Nucleo-F401RE System
The Nucleo-F401RE board (figure 4.1) has $512kB$ of flash memory, maximum internal clock frequency of $84MHz$, $96kB$ of SRAM, it needs a power supply of $3.3V$, $5V$, $7V$ or $12V$.
The HDF5 file is converted to C code using the X-CUBE-AI 7.0.0 package of the STM32CubeIDE 1.4.0. Three different C code models are obtained which performances are compared: uncompressed, compressed with a factor of 4 and 8. Since the microcontroller receives the data from the VNA its algorithm differs from that of the previous system:

1. **Initialization**: Initialization of: The vectors of mean and standard deviation of 660 elements each, the structure that adapts the data received from the VNA via UART, a portion of memory used to hold intermediate values of the neural network activations, two buffers used to store input and output tensors, the pointer to the model, a wrapper structs that hold pointers to data and info about the data (tensor height, width, channels), all configured peripherals (GPIO , DMA, USART), the instance of neural network.
2. **Data Reception**: Once the reception of the 2640 bytes (4 bytes per sample) is completed, an interrupt is activated in which the structure of 660 float data is saved. The flag that allows to start the inference is activated.
3. **Standardization**: Each float data is standardized. $feature_i = \dfrac{sample_i - \mu_i}{\sigma_i}$
4. **Inference**: The inference of the standardized data is computed, and the value of the sigmoid neuron output is saved.
5. **Output Elaboration**: The prediction is obtained through the threshold set at the value of 0.25, and the result activates or deactivates a led depending on whether the threshold value is exceeded or not. The flag that allows to start the inference is deactivated.

The GPIO peripheral is used to drive the output signal for the LED, The USART and DMA are used to receive data from the VNA in direct memory access mode. Points 2 to 5 are iterated for each product.

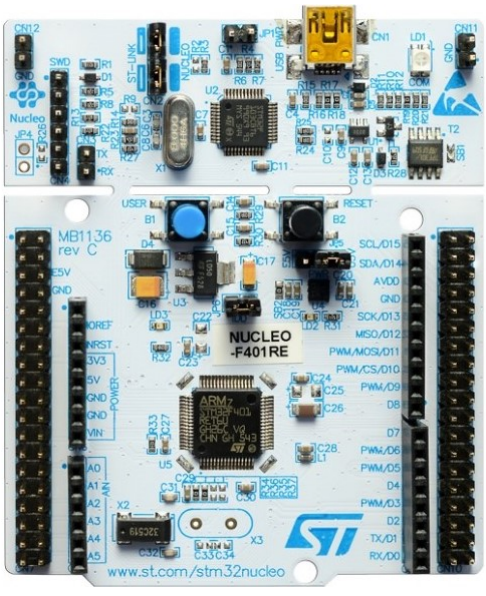


Figure 4.1: Pictures of the Nucleo-F401RE board.

4.3 Performance Comparison

A lower latency allows to have a higher production, the biggest production is obtained at the maximum speed of the conveyor belt possible and without spacing between products. Assuming that the distance between one jar and the next is zero, in order to have a continuous flow of products on the conveyor belt at a speed of $50cms^{-1}$ the maximum latency is $228.6ms$ if the jars are positioned along the distance of $10.67cm$, while it is $173.8ms$ if positioned along the distance of $8.69cm$. Another constraint to be respected is the size of the model, in fact it must be such as not to exceed the limit of hardware resources of the microcontroller, much more limited than that available in the VNA.

The performance metrics are acquired during a simulation of the operation of the MIT-Food system.

Compression Factor	**ROM** $\{kib\}$	**Relative Memory Reduction**	**Inference Time** $\{ms\}$	**Relative Inference Time Increase**	**Recall Loss**
none	170276	-	3.767	-	-
4	44580	73.80%	4.308	14.36%	1.67%
8	22500	86.78%	4.505	19.59%	2.51%

Table 4.1: The performance metrics of the Nucleo-F401RE system compared to its two levels of compression of weights and biases.

Compression Factor	**ROM** $\{kib\}$	**Relative Memory Reduction**	**Inference Time** $\{ms\}$	**Relative Inference Time Increase**	**Recall Loss**
none	163840	-	10.030	-	-
4	42008	74.36%	7.779	22.45%	3.03%

Table 4.2: The performance metrics of the VNA system compared to its level of compression of weights and biases.

Although the reduction of the memory occupied in the case of a compression factor 4 is greater than 70% in both systems (even more than 85% in the compression factor 8), as shown in tables 4.1 and 4.2, the fact of not having exceeded the limit of available memory does not justify the choice of configurations that increase the

inference time and degrade the recall.

To calculate the total latency of a specific configuration in a system, T_{acq} the acquisition time of the scattering matrix, T_{std} the time to perform the standardization of samples, and T_{led} the time to send a signal to the led must be considered, which respectively are $T_{acq} = 50ms$ on average, $T_{std} = 28\mu s$ for the VNA, $T_{std} = 629\mu s$ for the Nucleo, and $T_{led} = 70\mu s$.

To calculate the latency in the configurations of the Nucleo system there is also the time necessary for the complete transmission of data from the VNA, since the data are 660 and the UART transmission has a speed of $115200bit/s$, the entire transmission takes $T_{UART} = 183ms$. So the latencies become:

$$\begin{aligned} T_{VNA} &= T_{aqc} + T_{std} + T_{inference} + T_{led} \\ &= 50ms + 0.028ms + 10.03ms + 0.07ms \\ &= 60.128ms \end{aligned}$$

$$\begin{aligned} T_{Nucleo} &= T_{aqc} + T_{UART} + T_{std} + T_{inference} + T_{led} \\ &= 50ms + 183ms + 0.629ms + 3.767ms + 0.07ms \\ &= 237.466ms \end{aligned}$$

The system with the best latency turns out to be the VNA, but it should be noted that if the Nucleo system were able to acquire the scattering matrices through an acquisition circuit there would be no T_{UART} factor, which would reduce its latency by 77% down to $T_{Nucleo} = 54.466ms < T_{VNA} = 60.128ms$.

Chapter 5

Conclusions

In this thesis it has been shown that it is possible to detect physical contaminants inside a food product with the joint use of MWS and ML technology, resulting in a system with less latency than a traditional MWI technology.
By training the network with a dataset comprising acquisitions of products at 10 frequencies in the range [9,11] GHz, a 3 folds CV MLP was obtained with a recall of 99.84% and an accuray of 99.76%, while with the conveyor belt dataset at two different speeds, no cases of FN misprediction have even been recorded, proving that the best acquisition technique to train the neural network is with 10 frequencies. The latency of the implementation with the MLP in the VNA system ensures continuous in-line production, while the micro controller does not guarantee it, even if using a different data acquisition system the latency would be reduced and it would be even more efficient than the VNA system. Therefore MIT-Food prototype system has the potential to become a valid technique for the recognition of contaminants in the production lines of food industrial companies.

Possible changes that would lead to further improvements and validation of the MIT-Food prototype system are:

- To use larger datasets with different contaminants that may be present in industrial environments and different conveyor belt speeds.
- To train a classifier for each antenna [20].
- To create an acquisition system that allows the microcontroller to eliminate the latency due to UART communication. This would allow to lower the prices of the system by several orders of magnitude, being the price of a VNA in the thousands euros while that of a microcontroller around tens of euros
- To validate the behavior of the system on mediums other than hazelnut

and chocolate spreadable cream, such as dairy products or non-homogeneous products that are therefore formed by materials of different dielectric constants, thus expanding the possible uses of the system.

Bibliography

[1] Quansheng Chen, Chaojie Zhang, Jiewen Zhao, and Qin Ouyang. «Recent advances in emerging imaging techniques for non-destructive detection of food quality and safety». In: *TrAC Trends in Analytical Chemistry* 52 (2013), pp. 261–274 (cit. on p. 1).

[2] Friedrichs R. Wright D. «Foreign Bodies - Techniques for Investigation and Identification.» In: (2017). URL: `https://www.rssl.com/%5C%7E/media/%20rssl/en/files/documents/white-paper/rssl-foreign-bodies.pdf?la=en.` (cit. on p. 2).

[3] Kaiqiang Wang, Da-Wen Sun, and Hongbin Pu. «Emerging non-destructive terahertz spectroscopic imaging technique: Principle and applications in the agri-food industry». In: *Trends in Food Science & Technology* 67 (2017), pp. 93–105 (cit. on pp. 2, 27).

[4] Binbin B Hu and Martin C Nuss. «Imaging with terahertz waves». In: *Optics letters* 20.16 (1995), pp. 1716–1718 (cit. on p. 4).

[5] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019 (cit. on pp. 8, 12, 16).

[6] scikit-learn developers. «scikit-learn user guide. Release 0.21.3.» In: (2019) (cit. on pp. 11, 18, 21, 22).

[7] BaAdam J. Kingma D. «A Method for Stochastic Optimization. 3rd International Conference for Learning Representations, San Diego.» In: (2015) (cit. on p. 12).

[8] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, 2015 (cit. on p. 12).

[9] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (cit. on p. 12).

[10] Peter E Hart, David G Stork, and Richard O Duda. *Pattern classification*. Wiley Hoboken, 2000 (cit. on pp. 15, 21, 30).

[11] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*. Vol. 1. MIT press Massachusetts, USA: 2017 (cit. on p. 17).

[12] Sergey Ioffe and Christian Szegedy. «Batch normalization: Accelerating deep network training by reducing internal covariate shift». In: *International conference on machine learning*. PMLR. 2015, pp. 448–456 (cit. on p. 18).

[13] Google Developers. «Machine Learning Crash Course.» In: (). URL: `https://developers.google.com/machine-learning/crash-course/` (cit. on pp. 19, 26).

[14] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*. Vol. 26. Springer, 2013 (cit. on p. 20).

[15] Mohammed J Abdulaal, Alexander J Casson, and Patrick Gaydecki. «Performance of nested vs. Non-nested SVM cross-validation methods in visual BCI: Validation study». In: *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE. 2018, pp. 1680–1684 (cit. on p. 21).

[16] James Bergstra, Dan Yamins, David D Cox, et al. «Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms». In: *Proceedings of the 12th Python in science conference*. Vol. 13. Citeseer. 2013, p. 20 (cit. on p. 22).

[17] Peter I Frazier. «A tutorial on Bayesian optimization». In: *arXiv preprint arXiv:1807.02811* (2018) (cit. on p. 22).

[18] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. «Algorithms for hyper-parameter optimization». In: *Advances in neural information processing systems* 24 (2011) (cit. on p. 22).

[19] Zhanke Yan, Yibin Ying, Hongjian Zhang, and Haiyan Yu. «Research progress of terahertz wave technology in food inspection». In: *Terahertz Physics, Devices, and Systems*. Vol. 6373. International Society for Optics and Photonics. 2006, 63730R (cit. on p. 27).

[20] Yunpeng Li, Emily Porter, Adam Santorelli, Milica Popović, and Mark Coates. «Microwave breast cancer detection via cost-sensitive ensemble classifiers: Phantom and patient investigation». In: *Biomedical Signal Processing and Control* 31 (2017), pp. 366–376 (cit. on p. 48).