INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

Machine Learning Applied to an Intelligent and Adaptive Robotic Inspection Station

Luís Sousa Pinto Variz - 32957

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Engenharia Industrial.

Work oriented by:

Prof. Paulo Leitão

Prof. Pedro João Rodrigues

Bragança

2019

Machine Learning Applied to an Intelligent and Adaptive Robotic Inspection Station

Luís Sousa Pinto Variz - 32957

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Engenharia Industrial.

Work oriented by:

Prof. Paulo Leitão

Prof. Pedro João Rodrigues

Bragança

2019

Dedication

This thesis is dedicated to my wife Ana Variz, who with so much effort and support, it was possible to conclude this important stage. . . .

Acknowledgement

First of all, special thanks to my wife Ana Variz, and to my children Beatriz Variz and Diogo Variz, for their unconditional love and support, so it could be possible to conciliate, academic, professional and family life. Special thanks to the supervisors Paulo Leitão, Pedro João Rodrigues and Luís Piardi, for all the time dispensed, by the orientations, critical analyses, motivation and confidence they have always transmitted along this path. To the IPB, mainly the professors and members from CeDRI for the support, companionship and availability in the accomplishment of the various tasks, by sharing their knowledge. To all my friends who have always support me, with friendship shown over the years. Good Luck to all.

Abstract

Industry 4.0 promotes the use of emergent technologies, such as Internet of Things (IoT), Big Data, Artificial Intelligence (AI) and cloud computing, sustained by cyber-physical systems to reach smart factories. The idea is to decentralize the production systems and allow to reach monitoring, adaptation and optimization to be made in real time, based on the large amount of data available at shop floor that feed the use of machine learning techniques. This technological revolution will bring significant productivity gains, resources savings and reduced maintenance costs, as machines will have information to operate more efficiently, adaptable and following demand fluctuations. This thesis discusses the application of supervised Machine Learning (ML) techniques allied with artificial vision, to implement an intelligent, collaborative and adaptive robotic inspection station, which carries out the Quality Control (QC) of Human Machine Interface (HMI) consoles, equipped with pressure buttons and Liquid Crystal Display (LCD) displays. Machine learning techniques were applied for the recognition of the operator's face, to classify the type of HMI console to be inspected, to classify the state condition of the pressure buttons and detect anomalies in the LCD displays. The developed solution reaches promising results, with almost 100% accuracy in the correct classification of the consoles and anomalies in the pressure buttons, and also high values in the detection of defects in the LCD displays.

Keywords: Machine learning, Artificial vision, Quality control, Industry 4.0, TensorFlow, Convolution neural network

Resumo

Indústria 4.0 promove o uso de tecnologias emergentes, como Internet of Things (IoT), Big Data, artificial intelligence (AI) e cloud computing, sustentadas por sistemas ciberfísicos, como o designio de alcançar o que chamam de fábricas inteligentes. A ideia é descentralizar os sistemas de produção e permitir que a monotorização, a adaptação e a otimização sejam feitos em tempo real, com base na grande quantidade de dados disponíveis no ambiente fabril que alimentam o uso de técnicas de machine learning (ML). Esta revolução tecnológica trará ganhos significativos de produtividade, economia de recursos e custos de manutenção mais reduzidos, pois as máquinas terão informações para operar com mais eficiência, adaptáveis e acompanhar as flutuações de procura. Esta tese discute a aplicação de técnicas supervisionadas de ML, aliadas à visão artificial, para a implementação de uma estação de inspeção robótica inteligente, colaborativa e adaptativa, que realiza o controlo de qualidade de consolas HMI, equipados com botões de pressão e displays LCD. Técnicas de ML foram aplicadas para o reconhecimento facial do operador, para classificação do tipo de console HMI a ser inspecionado, para classificar a condição do estado dos botões de pressão e deteção de anomalias nos displays LCD. A solução desenvolvida alcança resultados promissores, com quase 100 % de precisão na correta classificação das consolas e anomalias nos botões de pressão, e também valores elevados de acerto na deteção de defeitos nos displays LCD.

Palavras-chave: Aprendizado máquina, Visão artificial, Controlo qualidade, Indústria 4.0, TensorFlow, Redes neuronais convolucionais

Contents

Acknowledgement **vii**

Abstract **ix**

Resumo **xi**

Acronyms **xix**

1 Introduction **1**

1.1 Overview 1

1.2 Objectives 2

1.3 Document Structure 2

2 Related Work **5**

2.1 Quality Control (QC) 5

2.2 AI / ML Techniques 8

2.2.1 Artificial Neural Networks (ANN) 10

2.2.2 Convolution Neural Network (CNN) 13

2.3 Computer Vision 19

3 Intelligent Robotic Inspection Station **23**

3.1 Hardware Structure 23

3.1.1 UR3 Robot 24

3.1.2 FT300 force sensor 25

3.1.3 Image Acquisition System 27
3.1.4 Human Machine Interface (HMI) 28
3.1.5 Security equipment 29
3.2 Software structure 30

4 Application of Machine Learning Methods 33

4.1 User Face Recognition 33
4.1.1 Face Detection 33
4.1.2 Face Recognition 34
4.2 Classification of HMI Consoles 37
4.3 LCD Display Errors Detection 40
4.4 Classification of Buttons 43

5 Conclusions 47

Bibliography 49

A Original Project Proposal 56

B Python code 59

List of Tables

3.1 UR3 Robot Specifications 26
3.2 Specifications of the FT300 force-torque sensor 27
3.3 Mako G-125 Specifications 28

List of Figures

2.1 Example of a QC Inspection [1] 6
2.2 QC methods 6
2.3 Relationship between AI, ML, NN and Deep NN 9
2.4 Types of ML approaches 9
2.5 Neuron Model of Artificial Neural Network [3] 11
2.6 Topology of a feedforward ANN with input, hidden and output neurons . . 12
2.7 Structure of a CNN with four layers[22] 13
2.8 Representation of how an image is seen by computers compared to how humans visualize it 14
2.9 Visualisation of a convolutional layer with a 3x3 filter and a stride of 1 . . 15
2.10 Pooling example by a 2x2 filter with stride of 2 16
2.11 Vectorization of a 3x3 matrix into a row vector 17
2.12 Creating the optimal separating hyperplane with the use of support vectors [39] 18

3.1 Robotized inspection station and hardware structure 24
3.2 UR3 collaborative robot 25
3.3 FT300 force sensor [9] 25
3.4 FT300 force sensor coupled with the pressing tool 27
3.5 Mako G125b and Logitech 720p cameras 28
3.6 Human Machine Interface (HMI) 29
3.7 Inspection results in the user HMI app 29

3.8 a) Emergency stop button b) Status light c) Omron barriers sensor 30
3.9 Fluxogram for the execution of an inspection test. 31

4.1 Gradient example obtained from a single pixel 34
4.2 Face detection using HOG algorithm 35
4.3 Single triplet training step 36
4.4 128 measurements generated from image 36
4.5 Results of the face recognition procedure in the HMI app. 37
4.6 Type of classified objects a)HMI with LCD Display Stand B)One Button HMI c)One Button HMI Stand d)Empty station e)4 button Round HMI f)HMI with LCD Display 38
4.7 Example of images of dataset ImageNet [19] 38
4.8 Deep Convolutional Neural Network (DCNN) for classification [2] 39
4.9 Data augmentation image transformation: a) rotate, b) flip, c) blur and d) noise. 40
4.10 Display image acquisition 41
4.11 Display matching template example 41
4.12 Original Image vs Processed Image 42
4.13 The 22 templates used for matching 42
4.14 Match metric (R) results 43
4.15 Match results of the LCD display. 44
4.16 Robot pressing the buttons during the inspection task. 44
4.17 Real force data from the pressure action over buttons. 45
4.18 kNN performance measures using confusion matrix 46

Acronyms

AI	Artificial Intelligence.
ANN	Artificial Neural Network.
CNN	Convolution Neural Network.
DCNN	Deep Convolution Neural Network.
DL	Deep Learning.
DNN	Deep Neural Network.
DoF	Degrees of Freedom.
FC	Fully Connected.
ICT	Information and Communications Technology.
IoT	Internet of Things.
kNN	k-Nearest-Neighbor Classification.
LCD	Liquid Crystal Display.
ML	Machine Learning.
NDT	Non-Destructive Testing.

PoE Power over Ethernet.

QC Quality Control.

RMS Root Mean Square.

SPC Statistical Process Control.

SQC Statistical Quality Control.

SVM Support Vector Machines.

TQM Total Quality Management.

Chapter 1

Introduction

1.1 Overview

At the end of the XX century, industries began to automate their processes with the help of computing and automation [4]. Similarly, the Quality Control (QC) process has been modernized and automated, although in many cases it is still made by humans, which brings several problems, namely mistakes associated with the repetitive execution of tasks, resulting in high costs for the company.

Recently, the impressive progresses in Artificial Intelligence (AI), driven by exponential increases in the computing power and by the availability of vast amounts of data, leads us to a new industrial age, which we call the 4th industrial revolution, who also enabled the development of automatic and intelligent QC solutions.

Machine Learning (ML) is a specific strand of AI that trains machines to learn with data that consists in the execution of algorithms that automatically create knowledge representation models based on a data set. By letting the algorithm "learn", to iterative adjust the knowledge representation model, it is possible to improve its performance. After this training phase, the model has a potential to make predictions, quality diagnoses in future situations, related to historical patterns with minimal human intervention, optimizing processes, increasing the reliability and generating savings.

Current QC solutions also use image processing techniques allied with ML algorithms to perform more efficient and faster inspection tasks. The application of automated visual control systems has been able to improve the industry's productivity [52], establishing reliable criteria to control the quality of products and services with reduced costs.

Applying ML to this kind of systems, it is possible to make systems more flexible and adaptive, able to learn in an evolutionary way, making systems more capable of achieving higher and higher efficiency levels.

1.2 Objectives

In this new industrial revolution, the introduction of AI into industrial automation environments is of crucial importance, allows the implementation of advanced data analysis to support the tasks of monitoring, diagnostics, prediction and optimization.

This work consists in the application of ML techniques to develop an intelligent and adaptive robotic inspection station, capable to perform QC tasks in HMI consoles of different types and configurations without any human intervention, with a high accuracy rate.

The application of ML algorithms was performed for the tasks related to the operator recognition, classification of the type of HMI under inspection, detection of errors in the Liquid Crystal Display (LCD) display and classification of the push buttons condition.

With this approach is expected to provide a more reliable, flexible, adaptable and robust solution by reducing setup times, in case of condition changes and increase the system performance.

1.3 Document Structure

The rest of this work is organized as follows. Chapter II overviews the state of the art related to industrial quality control using artificial vision using ML algorithms and the theoretical background necessary for understanding the methods discussed in the next chapters and chapter III presents the intelligent and adaptive robotic inspection station. Chapter IV presents the application of ML algorithms to several situations in this robotic inspection station, namely the operator's face recognition, the HMI consoles classification, the LCD display errors detection and the pressure buttons condition classification. Finally, chapter V rounds up the work with the conclusions and points out the future work.

Chapter 2

Related Work

This work focuses mainly on the application of ML algorithms to a robotic system equipped with artificial vision as a solution to achieve an adaptable system with higher efficiency levels in the different inspection tasks comparing with other solutions that mainly uses image processing.

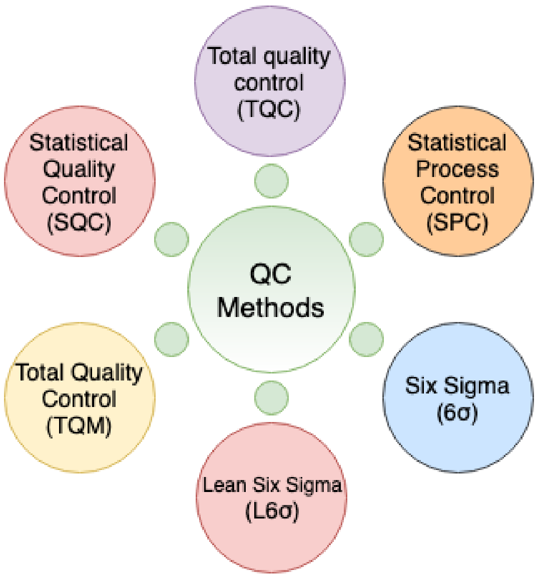
Having this in mind, this chapter reviews the concept and actual developments in QC, particularly those using AI combined with artificial vision to achieve high levels of efficiency. The necessary theoretical background related to ML techniques and computer vision will also be provided for a better understanding of the developed solutions described in the next chapters.

2.1 Quality Control (QC)

QC is a process through which a business seeks to ensure that product quality is maintained or improved with either reduced or zero errors. QC requires the business to create an environment in which both management and employees strive for perfection. This is done by training personnel, creating benchmarks for product quality and testing products to check for statistically significant variations. QC is the task of ensuring that products reach a certain standard, either defined by the company or by customers(see Figure 2.1). For consumers, the quality of a product is often considered to be the “fitness for use” of the product [31].

Figure 2.1: Example of a QC Inspection [1]

Several QC techniques have been developed since the 1930s, as represented in Figure 2.2), a few of these have ended up in widespread, like the Statistical Quality Control (SQC), based on application of statistical methods, specifically control charts and acceptance sampling [43].


Total quality control (TQC)
Statistical Quality Control (SQC)
Statistical Process Control (SPC)
QC Methods
Total Quality Control (TQM)
Six Sigma (6σ)
Lean Six Sigma (L6σ)
Figure 2.2: QC methods

Other example is the the Statistical Process Control (SPC), that consists of using control charts to monitor an individual industrial process and feed back performance to the operators responsible for that process, and the Total Quality Management (TQM), that uses in part the techniques of SQC to drive continuous organizational improvement [8]. More recently, the 6 Sigma method, originated by Motorola, consists in SQC applied to business strategy, is a data-driven methodology for eliminating defects that lead to six standard deviations between the mean and the closest specification limit in any process. Nowadays industries are using multiple Information and Communications Technology (ICT) to perform and automate several production tasks, such as scheduling and planning, process control, tracking and QC.

The field of QC was developed rapidly during the second half of the twentieth century and it is now an integral part of most manufacturing companies, being currently performed not only at the final stage with conventional techniques, like control charts and acceptance sampling, but instead along the production process [14], allowing to detect earlier quality deviations, resulting in higher quality standards and lower production cost.

With the current large amounts of data available at shop floor, there is unexplored potential in the manufacturing industry to develop more automatized and intelligent QC tasks by applying AI and particularly ML techniques.

The use of ML algorithms in inspection processes allows to identify earlier defects or deviation as well as perform diagnosis of possible problems.

There have been some practical implementations in industry with a few successful studies published on quality of fruit detected with visual machine learning algorithms [35] [41] and plastic injection molding, with production parameters as inputs [40].

Other examples of real appliance of ML solutions in the food industry, that is the one of the most dependent industries of the quality management, where the lack of quality of at least one ingredient can directly impact the quality of the final product [34], is the Kewpie Corporation (a major Japanese food company), that is using Google's TensorFlow machine learning libraries in their visual inspection system, to automatically detect anomalies in their diced potatoes[20].

Fujitsu has also developed a solution to identify potential defects in the manufacturing process by performing Non-Destructive Testing (NDT) inspection that is combined with image processing and Deep Learning (DL) techniques to realize a diagnosis in minutes [11].

Siemens is also using similar AI solutions to identify faults in glass slides up to 75 meters, sweeping each centimeter to identify any type of flaw that could cause manufacturing defects. The inspection process takes now 1 hour and half, when previously it took 6 hours to be completed [10].

These examples show how image processing and AI can be useful applied to improve the QC process. The advent of industry 4.0, and the availability of large amounts of data, allows to use AI to support the development of more adaptive and efficient inspection stations.

2.2 AI / ML Techniques

AI has intrigued people since the concept of computers was conceived. Currently AI is an exponential field of evolution with several applications in the understanding of speech or images, in scientific research and automation of the industry. AI has been good at solving problems that can be described by formal mathematical rules, but the real challenge is to deal with problems that are difficult to describe formally, which we humans intuitively resolve, such as speech recognition or image detail.

The solution to these problems is to allow computers to learn from experience and adapt to new circumstances. This can be facilitated by AI's statistical approach [14].

ML is a type of AI that aims to make machines learn solutions to specific problems without being explicitly programmed to do [30]. The concept that defines the relationship of what is AI ML and Deep Neural Network (DNN), raises several interpretations in the scientific community, the structure of this relationship in my point of view is what is represented in the Figure 2.3.

Many Machine Learning solutions have been derived from our knowledge of the human brain. Machine Learning models are used to solve two main tasks: classification and

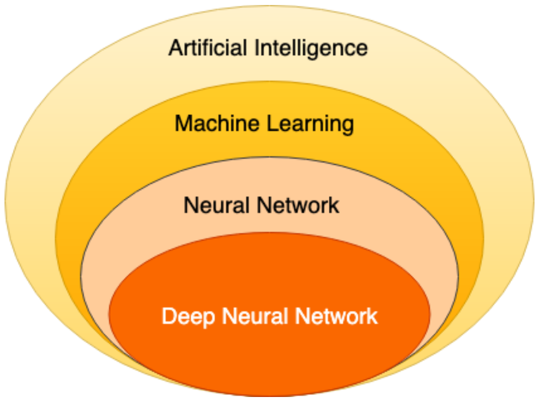


Figure 2.3: Relationship between AI, ML, NN and Deep NN

regression [21].

This thesis is about on extracting images features directly with the Machine Learning, instead of relying on extractors craft resources. Classification is the task of assigning a label to an entry, while regression generates a continuous output. A classification problem would be, for example, to detect images of cats in a large set of images, where a problem of regression would be to predict the price of a house, given its characteristics.

To address these two problems, three approaches are particularly considered, as illustrated in Figure 2.4:

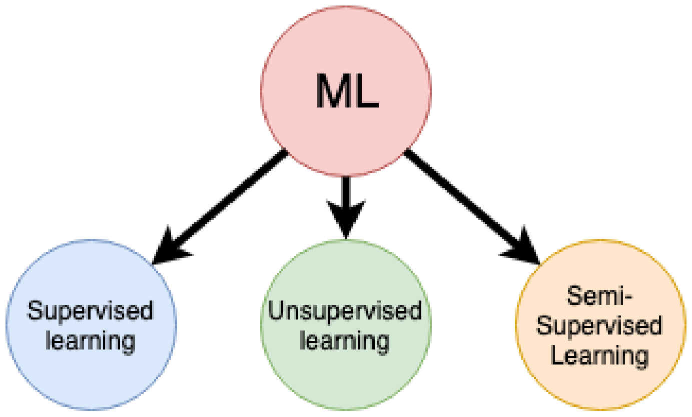


Figure 2.4: Types of ML approaches

- **Supervised learning:** the model is learned from expected input and output data. This is the most common form of learning. In fact, most of the training algorithms are computing an initial output, comparing it to the expected output and adapting the system to get closer to the expected data.

- **Unsupervised learning:** the model is learned only from input data. This approach is particularly useful in practice since unlabeled data is plentiful, while labeled data is more scarce and requires a lot of effort to collect. However, due to its unsupervised feature, this technique can not be used directly for classification. It is widely used to automatically discover structures in the input space, such as the presence of clusters.

- **Semi-Supervised Learning:** Both types of data are used to train the model. The model is pre-trained using unsupervised data and then enhanced with supervised data or the opposite. There is usually more unsupervised data available and this approach makes use of this advantage. It is also possible to train a model using both types of data at once, but this is more complex and rarely used.

In this study, the category supervised learning will be used. The training of any supervised machine learning model requires a substantial amount of diversified data. The amount of data is one of the major influences on how well a model will perform. More data leads to better training and therefore better predictions.

2.2.1 Artificial Neural Networks (ANN)

Artificial Neural Network (ANN) are powerful tools that can learn to solve problems in a way similar to how the human brain works. ANNs gather knowledge by detecting the patterns and relationships in data and learn through experience, not from programming [15].

The ANN might consist of several thousand artificial neurons, and the output of one neuron becomes an input to another neuron. The sending of signals from neuron to neuron in ANNs is facilitated by the use of a transfer function.

There are several different types of transfer functions that can be used. One of the most common transfer function is the logistic sigmoid function which is given by the following equation 2.1:

$$output = \frac{1}{1 + exp^{-}(\sum w_i x_i + w_0)} \tag{2.1}$$

where w_i is the weight for each input, x_i and w_0 is the bias weight. Bias is an additional parameter which is used to adjust the output along with the weighted sum of the inputs to the neuron.

Some of the inputs have greater importance to the output, called an activation, than others. This is modelled by weighting the input of each neuron. The weights of the inputs are summed up and the chosen transfer function sends the output signal to the next neuron[37].

The activation output is given by equation 2.2:

$$output = f(\sum_{i=0}^{3} w_i x_i) \tag{2.2}$$

A simple representation of a network with one node and activation output is shown in Figure 2.5.

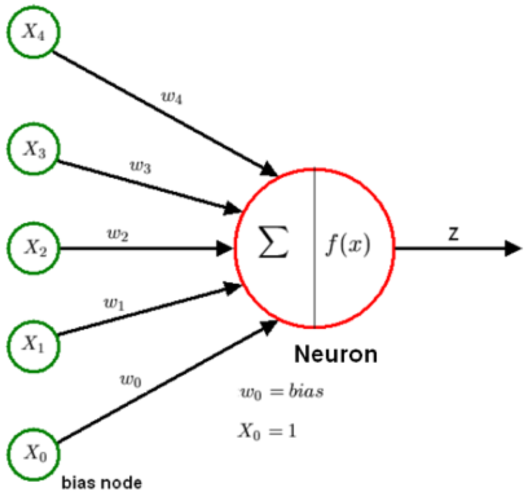


Figure 2.5: Neuron Model of Artificial Neural Network [3]

ANNs are constructed using multiple layers, divided into three groups: input layer, hidden layers and output layer. The input layer distributes the input data to the first hidden layer. The number of neurons in the input layer is equivalent to the dimension (number of variables) of the input data.

The bulk of layers in an ANN are the hidden layers. These layers process the input. The output is then sent forward to another hidden layer to function as input, where the same procedure is performed.

The output from the last hidden layer is sent as input to the output layer. This is called a feedforward neural network. The data is transferred in only one direction. Travelling from input layer through hidden layers forward to the output layer without any loops or cycles. In Figure 2.6 a general topology of an ANN is depicted.

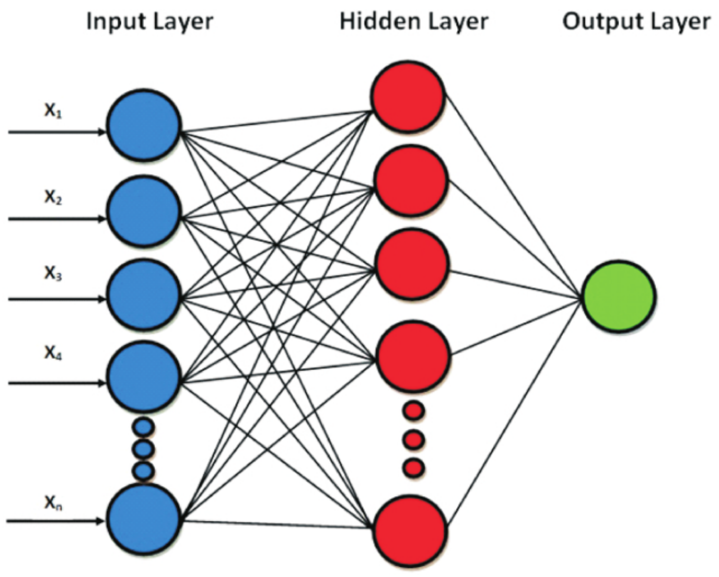


Figure 2.6: Topology of a feedforward ANN with input, hidden and output neurons

The topology in Figure 2.6 is one example of all the possible ways to structure the network. The programmer can, for example, specify the number of hidden layers, the number of neurons in the layers and the number of output categories.

The connections between neurons can be changed in order to send signals back through the network and between neurons in the same layer. These types of networks are called Recurrent Neural Networks (RNN) [3].

One topology often used is the Fully Connected Network. In such a model, all the neurons in adjoining layers are connected. Since all the neurons are connected between layers, the number of parameters increases exponentially with every added layer. The huge amount of parameters leads to a great computational cost and can also contribute to problems with overfitting [53].

As mentioned earlier, recent studies have shown that the ANN topology Convolution Neural Network (CNN) has greatly improved the accuracy of image classification. CNN requires just a fraction of the amount of parameters in a corresponding fully connected ANN [14].

2.2.2 Convolution Neural Network (CNN)

CNN are networks specialized for handling information that has a grid-like topology. As the name of the network implies, a linear operation named convolution is applied to the data.

CNNs architecture is composed of four different kinds of layers. The layers are called input, convolutional, pooling and fully connected. It is the convolutional, pooling and fully connected layers, except output, that corresponds to the hidden layers in a CNN.

The before mentioned layers will be further explained in the following subsections. In Figure 2.7, a general structure of a CNN is displayed.

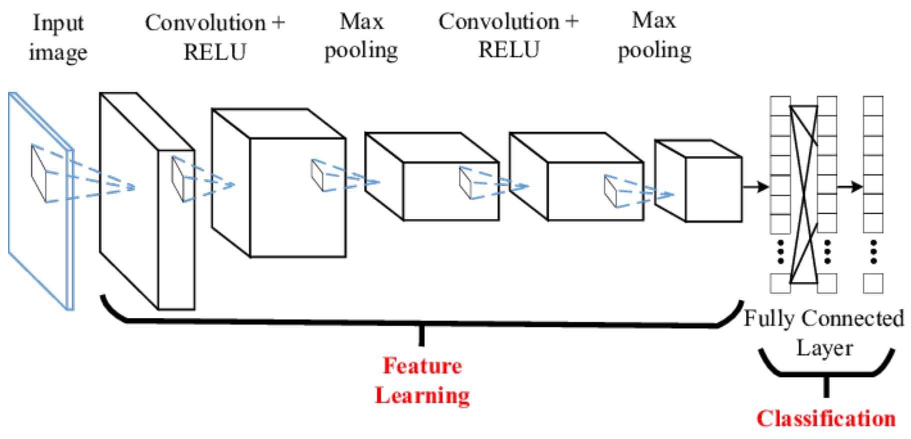
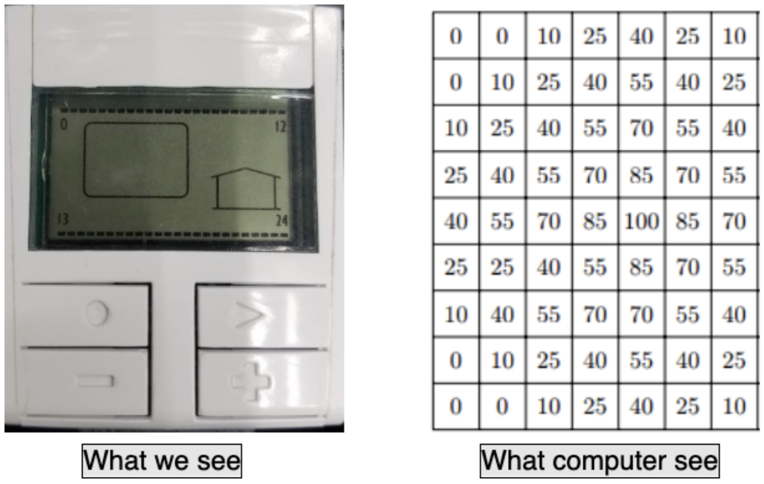


Figure 2.7: Structure of a CNN with four layers[22]

Input Layer

The input layer is where the data points serving as input is introduced to the model. In this study, the input data is images. The images will for the computer be seen as an array of pixel values. In figure 2.8 a representation of an image as humans visualize it and what the computer “visualize”.



0	0	10	25	40	25	10
0	10	25	40	55	40	25
10	25	40	55	70	55	40
25	40	55	70	85	70	55
40	55	70	85	100	85	70
25	25	40	55	85	70	55
10	40	55	70	70	55	40
0	10	25	40	55	40	25
0	0	10	25	40	25	10

Figure 2.8: Representation of how an image is seen by computers compared to how humans visualize it

The purpose of the convolutional layer is that it applies a set amount of convolutional filters on the image and performs linear computations to the input array. The number of filters is up to the programmer to specify.

The filters extract features from the image and create a corresponding feature map to each filter [33]. The extracted features correspond to the largest changes in the images, which are gradients between different regions in the image. For example, if there is an image with white background and a straight black line in the middle, the largest gradient will be found at the interface between them.

The convoluted data, i.e. the linear feature maps, is then processed by a transfer function called Rectified Linear Unit (ReLU). ReLU introduces non-linearity to the network since most of the data that CNNs process is non-linear. It corrects the feature maps with a threshold operation, any input value less than zero is set to zero thus ensuring that the feature maps are always positive [51].

The feature maps, produced by the convolving and ReLU, are the input to the pooling layer. One way to visualize a convolutional layer is to think of the filter as a microscope that amplifies a specific region, called the receptive field, on the input data. The array of the image has the dimensions W x H x D, where W is the width of the image, H is the height of the image and D is the number of colour channels.

The filter is a matrix of numbers, called weights. The filter is smaller in spatial size than the input matrix but must have the same number of dimension. The filter will move over the image and perform element wise multiplications between the values in the filter with the original pixel values. The number obtained correspond to a high activation value in the original input. How the filters move over the input array is specified by what is called strides.

Each stride moves the filter a specified length, often 2 pixels. The filter focuses on the specific region reached and calculates an activation value for each region. The specific numbers calculated create a new array of output, a feature map, which is smaller in size than the input array [33].

A visualization of the convolutional layer, with a 3x3 filter and stride of 1 on a 5x5 input array, where the filter reduces the array from 5x5 to 3x3, with the filters first two strides are depicted in figure 2.9.

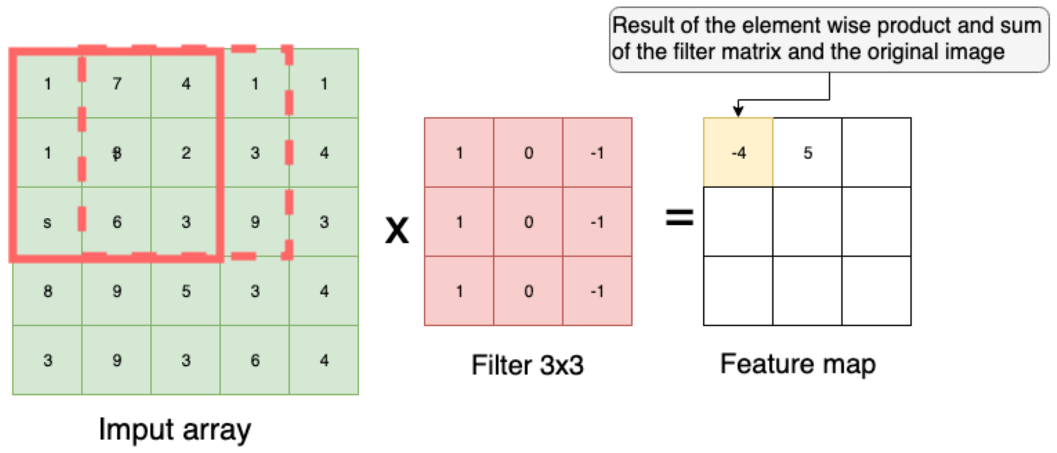


Figure 2.9: Visualisation of a convolutional layer with a 3x3 filter and a stride of 1

The output obtained from the convolutional layer is then sent to the pooling layer, or another convolutional layer, for further processing.

Pooling layer

The pooling layer purpose is to down sample the output from the feature maps. The idea is to apply a filter, usually with the size 2 x 2 and a stride with the same length, 2, on the output from the convolution layer.

By using a max pooling function, the highest value from the feature map is extracted. The idea is that once the specific feature from the original input is known, the relative location towards other features is more important than its exact location. Pooling significantly reduces the dimensions of the input volume [51]. This is shown in figure 2.10.

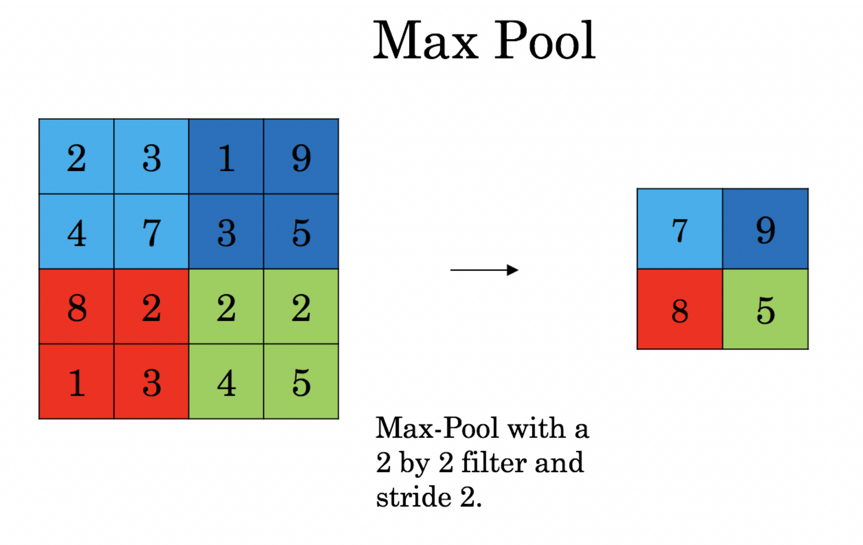


Figure 2.10: Pooling example by a 2x2 filter with stride of 2

Pooling reduces the parameters by 75% and ideally obtains the most important features. The reduction decreases the computational cost of the model. One other aspect of pooling is that it decreases the problem of overfitting [33].

Fully Connected layer

The name Fully Connected (FC) implies that every node in the previous layer is connected to every neuron in the current layer [29]. Fully-connected layers perform like a traditional neural network and contain about 90% of the parameters in a CNN.

To be able to connect every node on the first FC layer to the preceding layer, the outputs multidimensional arrays must be put in a single array. This is accomplished by applying vectorization to the matrices to perform a linear transformation into a single row vector, as ilustrate Figure 2.11.

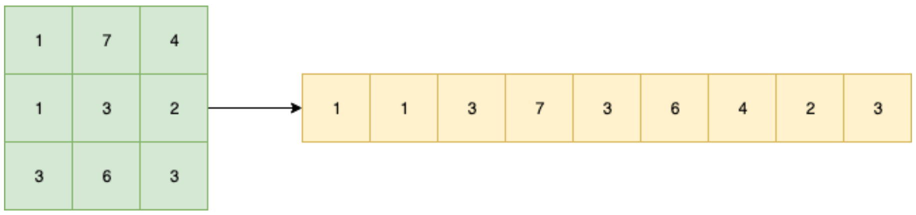


Figure 2.11: Vectorization of a 3x3 matrix into a row vector

Every neuron in the FC layer uses all the output from the previous layer as input. All the connections lead to a great increase in parameters for the network to process. The input is weighted and an activation function, normally ReLU, is applied to determine the output. The first FC layer is often followed by additional FC layers, which have fewer neurons. This is to extract the highest activation output and to decrease the amount of parameters.

Classification Layer

The output layer, or classification layer, is also an FC layer. It sorts the output from the final hidden layer into different categories using a different activation function than in the previous layers of the network.

The activation function that is commonly used in this step is the softmax function. This function compresses the output from the last hidden layer into probability values in the different categories. Therefore, the sum of the outputs is always equal to 1 [45]. The output from the classification layer corresponds to the class with the highest probability.

Support Vector Machines

Support Vector Machines (SVM) is a discriminative classification technique that derives from the Structural Risk Minimization principle from computational learning theory. The aim with SVM is to find the most optimal classification function that differentiates between units of classes in training data.

With a linearly separable dataset, the most optimal classification function can be decided by constructing a hyperplane which maximizes the margin between two datasets and thus creates the largest possible distance between datasets [48]. A visualization of this strategy is illustrated in figure 2.12.

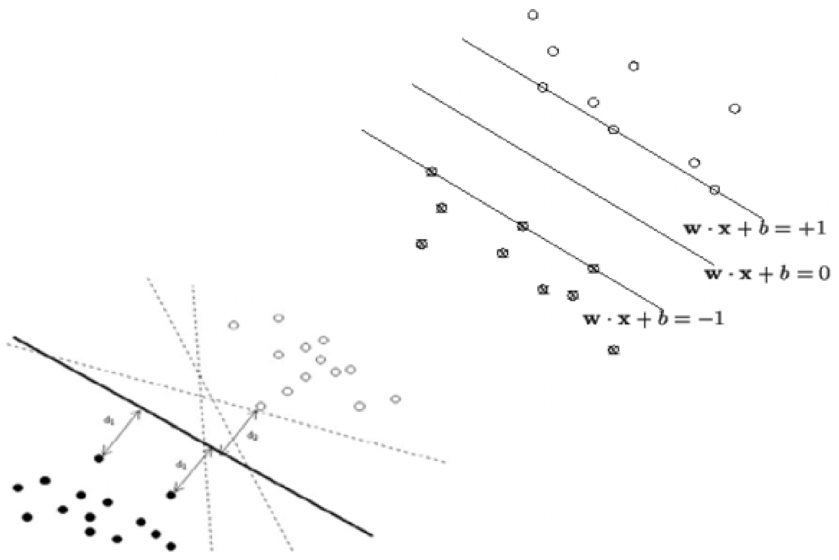

$\mathbf{w} \cdot \mathbf{x} + b = +1$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
$\mathbf{w} \cdot \mathbf{x} + b = -1$

Figure 2.12: Creating the optimal separating hyperplane with the use of support vectors [39]

The idea behind SVMs is that by finding the maximum margin and thus the most optimal hyperplane, the best generalization ability is reached. This results in the best classification performance for both the training data as well as future data [24].

k-Nearest-Neighbor (kNN)

k-Nearest-Neighbor Classification (kNN), is a machine learning algorithm that localizes a group of k objects in a training case that has the closest proximity to the test object, and then assigns a label derived from the prevalence of a class in the closest proximity. Three important components are needed for this algorithm: a group of labeled objects; a proximity metric; and the number k of nearest neighbors [50].

A popular proximity metric that is used for kNN classification is "Euclidian Distance", explained by formula 2.4:

$$D(x, y) = (\sum_{i=1}^{m} |x_i - y_i|^2)^{1/2} \tag{2.3}$$

When the k-nearest-neighbor list is acquired the test objects are categorized according to the majority class, given in formula 2.4:

$$MajorityVoting : Y = argmax \sum_{(x_i,y_i)D_z} (v = y_i) \tag{2.4}$$

Here, v represents the class label, y_i represents the *ith* nearest neighbor class label, and *I(·)* is the indicator functions that returns value one for a valid argument or zero for an invalid argument [50].

2.3 Computer Vision

Computer vision deals with the extraction of meaningful information from the contents of digital images or video. This is distinct from mere image processing, which involves manipulating visual information on the pixel level.

Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision and traffic automation [44].

Today, ML is a necessary component of many computer vision algorithms [42]. Such algorithms can be described as a combination of image processing and ML. Effective solutions require algorithms that can cope with the vast amount of information contained in visual images, and critically for many applications, can carry out the computation in real time [18].

In this thesis, it is used the object detection technique for the face recognition and image classification for the HMI consoles classification task.

What makes object detection a distinct problem is that it involves both locating and classifying regions of an image [12]. The locating part is not needed in, for example, whole image classification. Object detection is one of the classical problems of computer vision and is often described as a difficult task. In many respects, it is similar to other computer vision tasks, because it involves creating a solution that is invariant to deformation and changes in lighting and viewpoint.

To detect an object, we need to have some idea where the object might be and how the image is segmented. This creates a type of chicken-and-egg problem, where, to recognize the shape and class of an object, we need to know its location, and to recognize the location of an object, we need to know its shape [49].

Some visually dissimilar features, such as the clothes and face of a human being, may be parts of the same object, but it is difficult to know this without recognizing the object first. On the other hand, some objects stand out only slightly from the background, requiring separation before recognition[47].

Low-level visual features of an image, such as a saliency map, may be used as a guide for locating candidate objects [49]. The location and size is typically defined using a bounding box, which is stored in the form of corner coordinates. Using a rectangle is simpler than using an arbitrarily shaped polygon, and many operations, such as convolution, are performed on rectangles in any case. The sub-image contained in the bounding box is then classified by an algorithm that has been trained using ML [13].

The boundaries of the object can be further refined iteratively, after making an initial guess [44]. During the 2000s, popular solutions for object detection utilized feature descriptors, such as scale-invariant feature transform (SIFT) [27] developed by David Lowe in 1999 and histogram of oriented gradients (HOG) [5] popularized in 2005.

In the 2010s, there has been a shift towards utilizing CNN [13][12]. Before the widescale adoption of CNNs, there were two competing solutions for generating bounding boxes. In the first solution, a dense set of region proposals is generated and then most of these are rejected [26]. This typically involves a sliding window detector. In the second solution, a sparse set of bounding boxes is generated using a region proposal method, such as Selective Search [47]. Combining sparse region proposals with CNN has provided good results and is currently popular [12].

Chapter 3

Intelligent Robotic Inspection Station

Through ML techniques allied to artificial vision, it is intended to equip this inspection bench with the ability to perform the inspection of HMI consoles, with different morphology's and characteristics. The desired tasks are the user's facial recognition, detect the type of HMI console, anomalies in the display and the push buttons condition, in a more efficient and adaptable way to changing circumstances, reducing the margin of error.

In order to build this system, the first step was the development of its architecture, composed of hardware, to unite all parts of the system and use its functions, together with a software architecture and an application procedure.

In this chapter will be realized the description of the used hardware and software, and its methodology, which in an aggregated way performs the task of QC of the HMI consoles.

3.1 Hardware Structure

In this work an intelligent and adaptive inspection station was developed to perform QC of HMI consoles, as illustrated in Figure 3.1.

Having in mind the the dissemination of the concept of Internet of Things (IoT) as

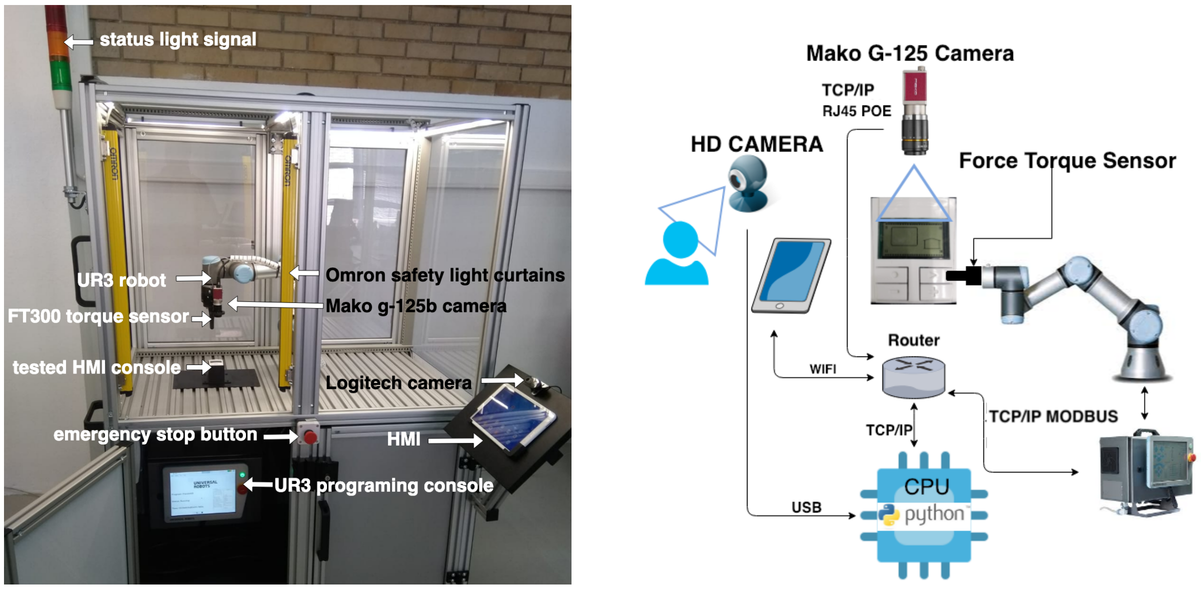


Figure 3.1: Robotized inspection station and hardware structure

illustrated in Figure 3.1, in this work manly, the TCP/IP is used to communicate between the industrial camera, the cpu and to read and write data from UR3 using the modbus protocol, interconnected by a router, also illustrated in Figure 3.1, who describes easily the hardware structure.

3.1.1 UR3 Robot

This inspection station is equipped by a Universal Robots UR3 collaborative robot, which performs the tasks of positioning the image acquisition system and inspecting the buttons of the HMI consoles.

It is a smaller collaborative table top robot for light assembly tasks and automated workbench scenarios, equipped with a control box and a programming user interface, illustrated in Figure 3.2.

As we can see at Table 3.1, the compact table-top robot weighs 11kg, with a payload of 3 kg, 360-degree rotation on all wrist joints and infinite rotation on the end joint. These unique features make UR3 the most flexible, lightweight, collaborative table-top robot to work side-by-side with employees in the market today.

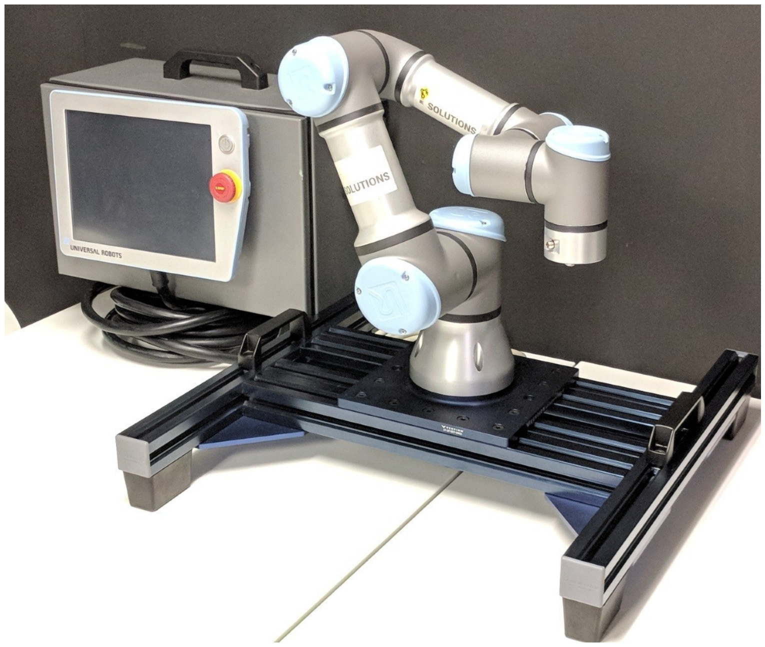


Figure 3.2: UR3 collaborative robot

It's an ideal choice for applications that require 6-axis capabilities where size, safety and costs are critical.

3.1.2 FT300 force sensor

The UR3 robot is equipped with a force-torque sensor FT300 of six Degrees of Freedom (DoF), see Figure 3.3, to measure the force of pressing the HMI buttons in the buttons inspection task.

Figure 3.3: FT300 force sensor [9]

UR3 Specifications	
Weight	11 kg / 24.3 lbs
Payload	3 kg / 6.6 lbs
Reach	500 mm / 19.7 in
Joint ranges	+/- 360 Infinite rotation on end joint
Speed	All wrist joints: 360 degrees/sec. Other joints: 180 degrees/sec. Tool: Typical 1 m/s. / 39.4 in/s
Repeatability	+/- 0.1 mm / +/- 0.0039 in (4 mils)
Degrees of freedom	6 rotating joints
Control box size (WxHxD)	16 Digital inputs 16 Digital out 2 Analog inputs 2 Analog outputs
I/O power supply	24 V 2A in control box and 12V/24V 600 mA in tool
Communication	TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX Ethernet socket & Modbus TCP
Programming	Polyscope graphical user interface on 12 inch touchscreen with mounting
IP classification	IP64
Power consumption	Approx. 100 watts using a typical program
Temperature	The robot can work in a temperature range of 0-50C*
Power supply	100-240 VAC, 50-60 Hz
Cabling	Cable between robot and control box (6 m) Cable between touchscreen and control box (4.5m)
Collaboration operation	15 advanced adjustable safety functions
Materials	Aluminum, PP plastic

Table 3.1: UR3 Robot Specifications

As illustrated in Table 3.2, it takes precise, repeatable, and high-resolution measurements. It can accomplish many different force-sensitive tasks with a fast time-to-production like pick and place, machine tending, quality testing, assembly applications.

Another tool is coupled to the force sensor to press the buttons, see Figure 3.4, providing to the robot an easy way to press the buttons with accuracy.

FT300	
Measuring Range Fx,Fy,Fz (N)	+/- 300
Measuring Range Mx, My, Mz (Nm)	+/- 30
Data Output Rate	100 Hz
Weight (Kg)	0.3
Supply Voltage	4.5 - 28V DC
Interface	RS-485 / USB
UR Caps Plug In	Plug & Play

Table 3.2: Specifications of the FT300 force-torque sensor

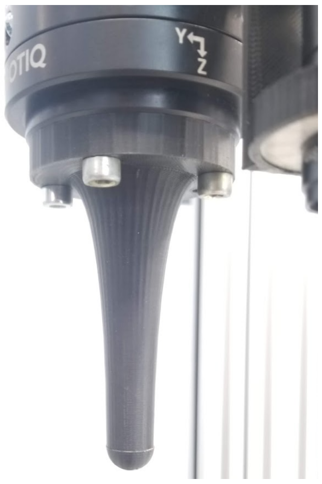


Figure 3.4: FT300 force sensor coupled with the pressing tool

3.1.3 Image Acquisition System

The image acquisition system, see Figure 3.5, is composed by an high-resolution industrial camera Mako G-125b for the image acquisition of the different inspection tasks of the HMI consoles and by a second camera, a Logitech with 720p of resolution, carries out the acquisition of the image to support the operator's face recognition by the system.

As illustrated in Table 3.3, the Mako G-125 is a monochromatic camera, with 1.2 megapixel GigE machine vision camera that incorporates the high quality Type 1/3" Sony ICX445 CCD sensor with EXview HAD technology. At full resolution, this camera runs at 30.3 frames per second. With a smaller region of interest, higher frame rates are possible.

This camera have the same ultra-compact form factor and the same mounting positions

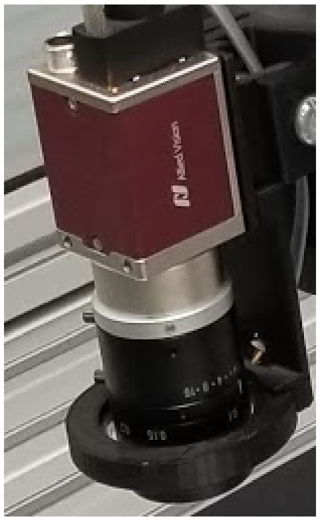


Figure 3.5: Mako G125b and Logitech 720p cameras

as many analog cameras. This model include Power over Ethernet (PoE), three opto-isolated outputs, and a 64 MB image buffer. The image quality profits from the precisely aligned sensor.

MAKO G-125C			
Resolution (MP)	1.2	Pixels (H x V)	1292 x 964
Type	Monochromatic	Camera Camera Control	GigE Vision
Camera Family	Mako	Dimensions (mm)	60.5 x 29 x 29
Exposure Control	12 s to 84 s	Frame Rate (fps)	30
Imaging Device	Sony ICX445	Camera Sensor Format	1/3"
Memory (MB)	64.00	Mount	C-Mount
Pixel Depth	8/12 Bit	Pixel Size, H x V (m)	3.75 x 3.75
Power Supply	PoE	Sensing Area, H x V (mm)	4.8 x 3.6
Synchronization	External/Software	Type of Sensor	Scan CCD
Video Output	GigE	Weight (g)	80

Table 3.3: Mako G-125 Specifications

3.1.4 Human Machine Interface (HMI)

As human interface, the inspection station is equipped with a Samsung Galaxy Tab, where the operator provides the instructions for operation and visualizes the results of the inspection task, illustrated in Figure 3.6.

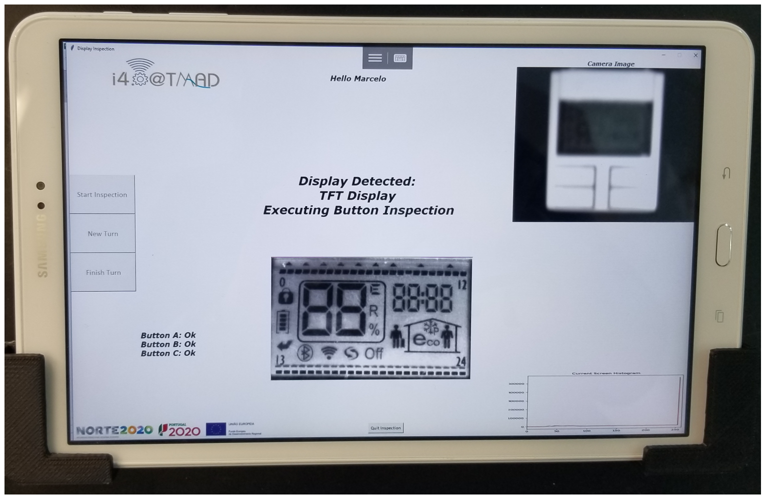


Figure 3.6: Human Machine Interface (HMI)

At the interface app is possible to visualize the results of the different tasks performed by the inspection station, namely the image of the user, the acquired image of the HMI console, histogram information of exposure control, information of the type of console acquired and the buttons inspection results as we can see ate the figures 3.7.

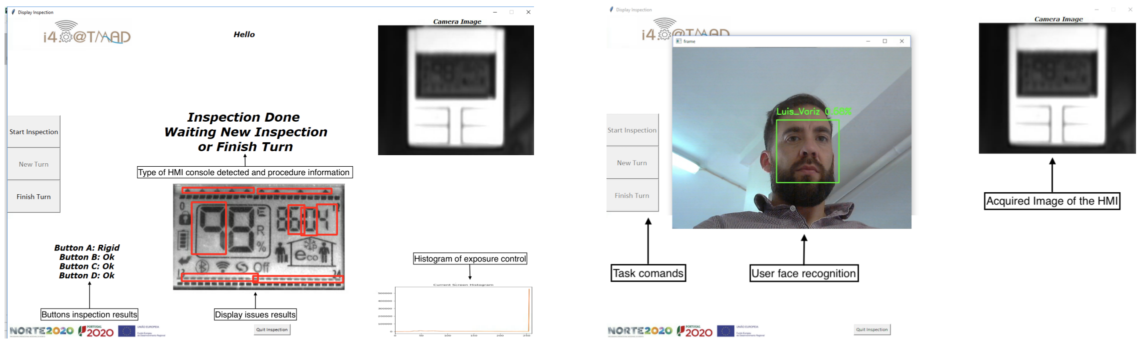


Figure 3.7: Inspection results in the user HMI app

3.1.5 Security equipment

Dealing with a collaborative area where the operator and the robot shares the same work space, to change the console that will be inspected, some security equipment are installed to improve its safety, namely an industrial barrier sensor, a status lights signal and an emergency stop button, as illustrate the Figure 3.8.

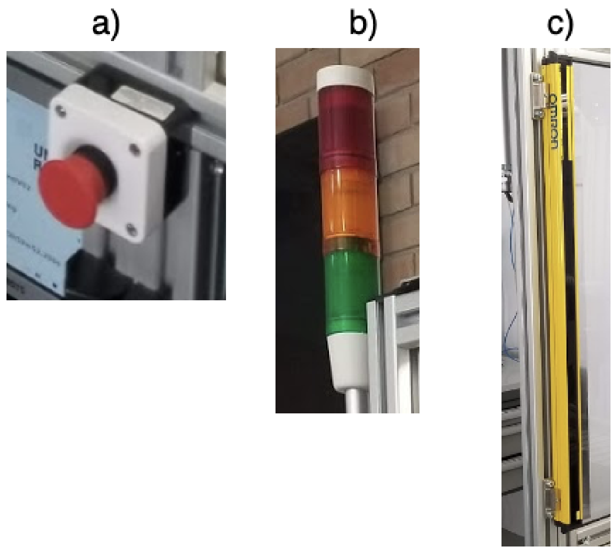


Figure 3.8: a) Emergency stop button b) Status light c) Omron barriers sensor

3.2 Software structure

These hardware apparatus is complemented, and managed, by a software app, codified in Python and running in an industrial PC, which uses ML techniques to perform several tasks during the inspection process, namely the operator recognition, the identification of the HMI console type, the classification of the buttons condition and the detection of errors in the LCD display.

This computational app provides flexibility and adaptability during the inspection tests since automatically can perform tests to consoles with different configurations, e.g., different number of buttons and different layout of the LCD display, allowing to reduce the setup time and improve the efficiency and productivity of the inspection process.

The developed app to perform the inspection testing is running continuously in cycle, following the flowchart illustrated in Figure 3.9.

In the beginning of the shift, a face recognition algorithm, using the image acquired by the Logitech camera, identifies the operator in charge by the inspection station. At this stage, the inspection station is ready to start inspecting the HMI consoles.

After detecting a new console in position to be inspected, the robot moves to the console position and the image of the console is acquired to be processed by an ML

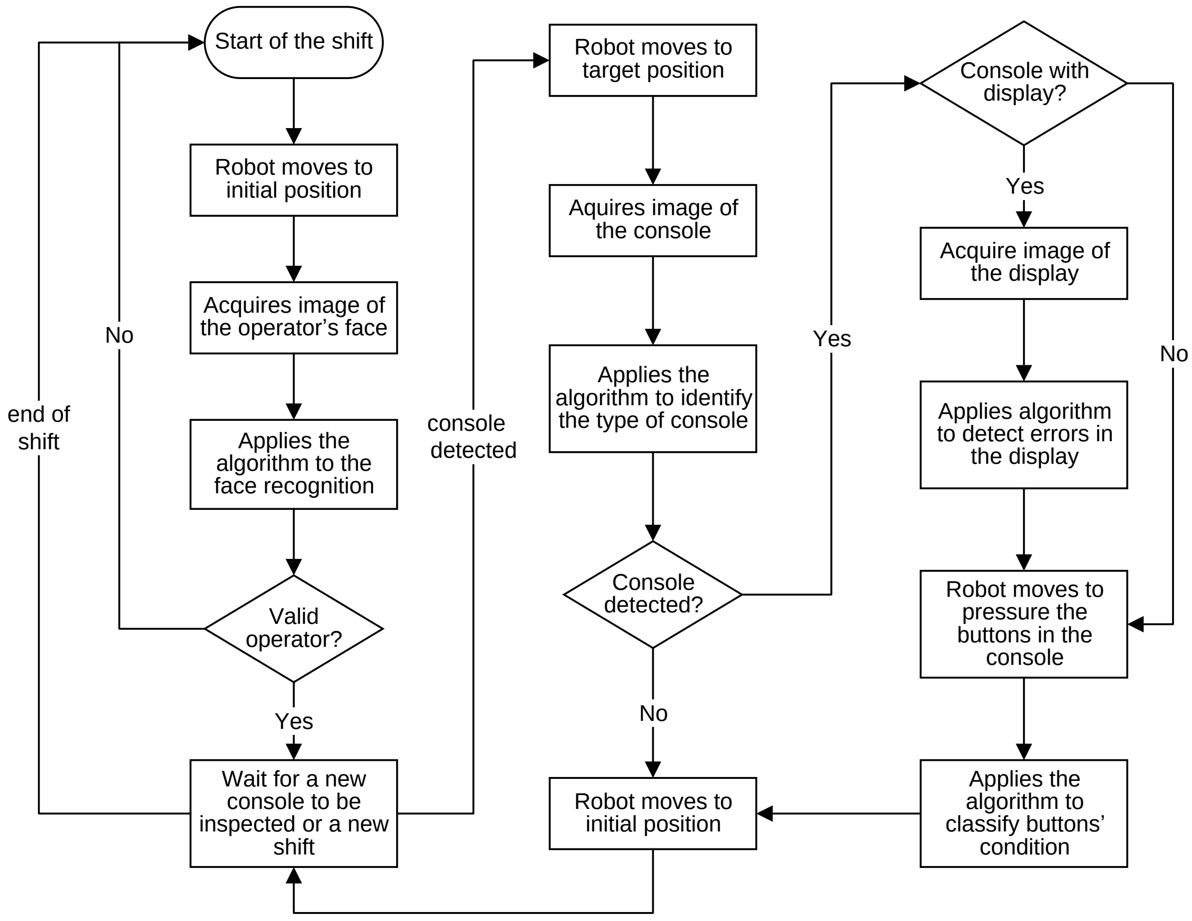

Start of the shift
Robot moves to initial position
Acquires image of the operator's face
Applies the algorithm to the face recognition
Valid operator?
No
Yes
Wait for a new console to be inspected or a new shift
end of shift
console detected
Robot moves to target position
Aquires image of the console
Applies the algorithm to identify the type of console
Console detected?
No
Robot moves to initial position
Yes
Console with display?
Yes
Acquire image of the display
Applies algorithm to detect errors in the display
Robot moves to pressure the buttons in the console
No
Applies the algorithm to classify buttons' condition

Figure 3.9: Fluxogram for the execution of an inspection test.

algorithm that will identify the type of console. This information is used to decide the next inspection actions according to the console configuration. In case of existence of buttons, the robot moves to touch the torque-force sensor in the buttons, being the acquired data feeding the ML technique that classifies the buttons condition. The image acquired by the Mako g-125b camera is used by a ML algorithm, associated to an image processing technique, to detect errors in the LCD display.

During the inspection process, the work space is shared by the robot who moves along different positions to realize the different tasks and the operator who performs the exchange of HMI consoles. This collaborative work is safeguarded by the installed safety systems, namely the robot slows down the movement speed if the security barriers are triggered, and stops its movement if the emergency button is pressed by the operator or the robot safety mechanism is triggered. New HMI consoles can easily and on-the-fly be added to the inspection catalogue provided by the inspection station, without the need of major changes in the inspection app, only requiring a retraining the ML algorithms.

In the next chapter will be described ML techniques applied and other approaches used to efficiently realize the QC of HMI consoles.

Chapter 4

Application of Machine Learning Methods

This chapter describes the application of image processing and ML algorithms applied to the automatic inspection of the HMI consoles, methodology, some difficulties and the results obtained for each task.

4.1 User Face Recognition

Face recognition is widely used in computer vision and in many other biometric applications where security is a major concern. The most common problem in recognizing a face arises due to pose variations, different illumination conditions. The automatic recognition of the operator that is associated to the inspection station is performed by applying ML techniques to the acquired image. The applied procedure comprises the face detection and the face recognition.

4.1.1 Face Detection

The face detection is performed by using the Histogram of Oriented Gradients (HOG) algorithm [7], which is provided in the dlib toolkit [6]. This algorithm looks at every single pixel in the image, one at a time, and for every single pixel searches for the surrounding pixels (see Figure 4.1). The goal is to figure out how dark the current pixel is compared to the surrounding pixels.

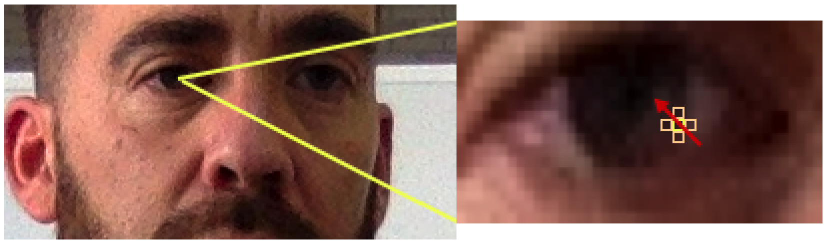


Figure 4.1: Gradient example obtained from a single pixel

An arrow is drawn to show in which direction the image is getting darker. If this process is repeated for every single pixel in the image, every pixel will be represented by its gradient direction. Thus, they show the flow from the light to dark across the entire image. But saving the gradient for every single pixel gives too much detail, being necessary to break up the image into small squares of 16x16 pixels each. In each square, counts up how many gradients point in each major direction then replaces that square in the image with the arrow directions that were the strongest. The end result is a very simple representation that captures the basic structure of a face, as illustrate in Figure 4.2.

In this work, a previous trained linear support vector machine (SVM) classifier was used to find the part of our image that looks the most similar to a known hog pattern, by sliding a window to identify and locate faces in an image [7].

4.1.2 Face Recognition

The face recognition is performed by using the deep metric learning technique, also provided by the dlib library [6], which provides as output the face characteristics. With deep learning is trained a network to accept a single input image and output a classification/label for that image. However with the deep metric learning is different, to instead of trying to output a single label, the objective is outputting a real-valued feature vector using a

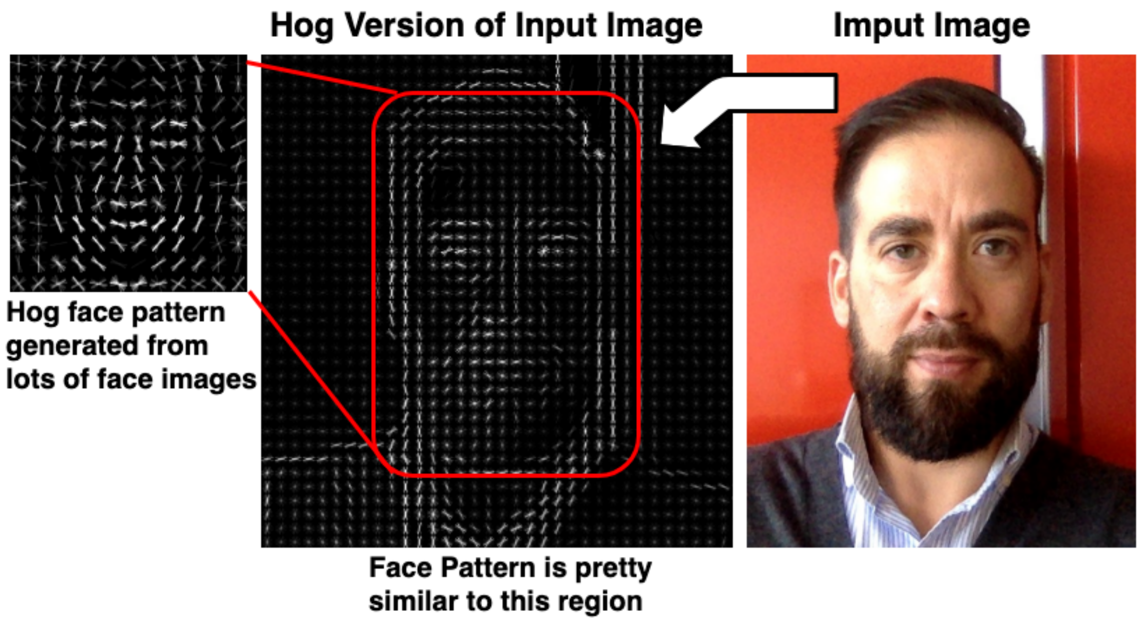

Hog Version of Input Image
Imput Image
Hog face pattern generated from lots of face images
Face Pattern is pretty similar to this region

Figure 4.2: Face detection using HOG algorithm

Deep Convolution Neural Network (DCNN).

This technique is based on a DCNN to learn distance features to assimilate different face samples [28]. For the dlib facial recognition network, the output feature vector is a list of 128 real-valued numbers, that is used to quantify the face previously detected. The training process works by looking at 3 face images at a time, loading two images of a known person and a picture of a totally different person [17], as illustrated in Figure 4.3. Repeating this step enough number of times, the neural network learns to generate 128 features that describe the face reliably.

After repeating this step millions of times, for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person.

The network architecture for the face recognition is based on the ResNet-34 network [16], with a few layers removed and the number of filters per layer reduced by half. In this application, it is used a previous trained network, which has an accuracy of 99.4% on the standard Labeled Faces in the Wild benchmark [23]. To create our dataset of features, each face is mapped by the previous trained neural network to the corresponding feature

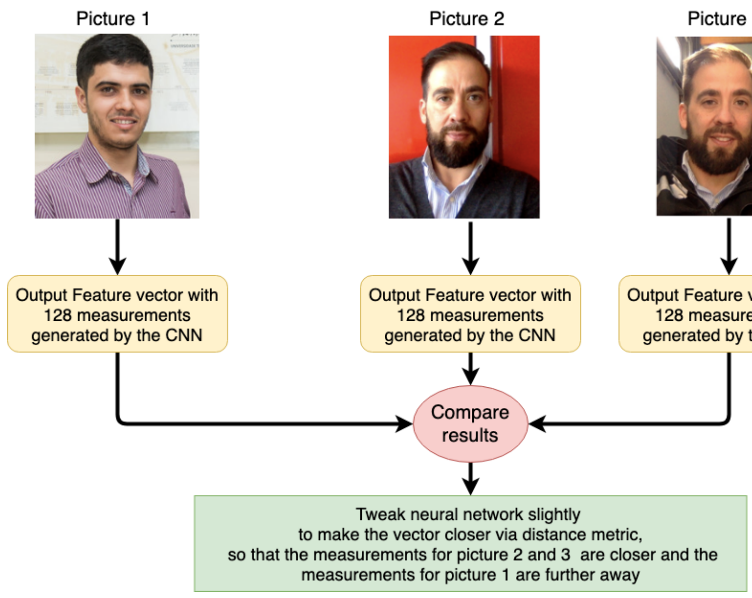
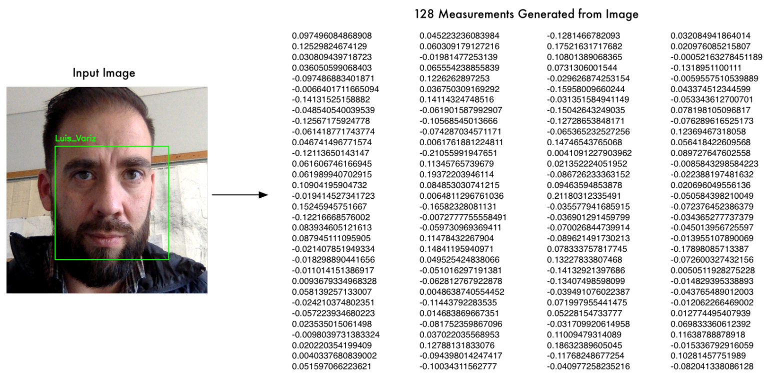

Picture 1
Picture 2
Picture 3
Output Feature vector with 128 measurements generated by the CNN
Output Feature vector with 128 measurements generated by the CNN
Output Feature vector with 128 measurements generated by the CNN
Compare results
Tweak neural network slightly to make the vector closer via distance metric, so that the measurements for picture 2 and 3 are closer and the measurements for picture 1 are further away

Figure 4.3: Single triplet training step



```
0.097496084868908     0.045223236083984     -0.1281466782093      0.032084941864014
0.12529824674129      0.060309179127216     0.17521631717682      0.020976085215807
0.030809439718723     -0.01981477253139     0.10801389068365      -0.00052163278451189
0.036050599068403     0.065554238855839     0.0731306001544       -0.1318951100111
-0.097486883401871    0.1226262897253       -0.029626874253154    -0.0059557510539889
-0.0066401711665094   0.036750309169292     -0.15958009660244     0.043374512344599
-0.14131525158882     0.14114324748516      -0.031351584941149    -0.053343612700701
-0.048540540039539    -0.061901587992907    -0.15042643249035     0.078198105096817
-0.12567175924778     -0.10568545013666     -0.12728653848171     -0.076289616525173
-0.061418771743774    -0.074287034571171    -0.065365232527256    0.12369467318058
0.046741496771574     0.0061761881224811    0.14746543765068      0.056418422609568
-0.12113650143147     -0.21055991947651     0.0041091227903962    0.089727647602558
0.061606746166945     0.11345765739679      0.021352224051952     -0.0085843298584223
0.061989940702915     0.19372203946114      -0.086726233363152    -0.022388197481632
0.10904195904732      0.084853030741215     0.09463594853878      0.020696049556136
-0.019414527341723    0.0064811296761036    0.21180312335491      -0.050584398210049
0.15245945751667      -0.16582328081131     -0.035577941685915    -0.072376452386379
-0.12216668576002     -0.0072777755558491   -0.036901291459799    -0.034365277737379
0.083934605121613     -0.059730969369411    -0.070026844739914    -0.045013956725597
0.087945111095905     0.11478432267904      -0.089621491730213    -0.013955107890069
-0.021407851949334    0.14841195940971      0.078333757817745     -0.17898085713387
-0.018298890441656    0.049525424838066     0.13227833807468      -0.072600327432156
-0.011014151386917    -0.051016297191381    -0.14132921397686     0.0050511928275228
0.0093679334968328    -0.062812767922878    -0.13407498598099     -0.014829395338893
0.058139257133007     0.0048638740554452    -0.039491076022387    -0.043765489012003
-0.024210374802351    -0.11443792283535     0.071997955441475     -0.012062266469002
-0.057223934680223    0.014683869667351     0.05228154733777      0.012774495407939
0.023535015061498     -0.081752359867096    -0.031709920614958    0.069833360612392
-0.0098039731383324   0.037022035568953     0.11009479314089      0.11638788878918
0.020220354199409     0.12788131833076      0.18632389605045      -0.015336792916059
0.0040337680839002    -0.094398014247417    -0.11768248677254     0.10281457751989
0.051597066223621     -0.10034311562777     -0.040977258235216    -0.082041338086128
```

Figure 4.4: 128 measurements generated from image

vector. During the classification, it was used a simple KNN algorithm to find the person in the database which has the closest feature vector to the probe face (mapped to its feature vector).

The data set used to train the network was constituted by some persons that are part of the Research Center in Digitalization and Intelligent Robotics (CeDRI). The results obtained were good, with success rates exceeding 50% for all elements, not generating failures in facial recognition. However, for people who are not in the database, the algorithm is able to detect as an unknown person, but with some sporadic failures.

To work around the problem the data set of pictures to train the network was increased, and also combine the confidence classification with a number of necessary matches, (>10) to avoid a false recognition event. As illustrated in Figure 4.5, the confidence is around 68%, and allied with necessary matches, increases the face recognition reliability and safety to use in a real industrial environment. Later several tests were performed, with known and unknown people, demonstrating a very high efficiency. This algorithm is fast and fluid, taking about 1,5 seconds to perform the user face recognition task.

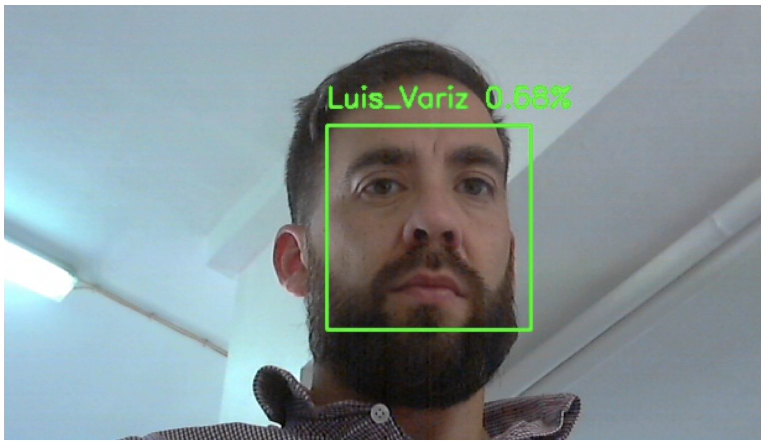


Figure 4.5: Results of the face recognition procedure in the HMI app.

4.2 Classification of HMI Consoles

In order to perform the classification task of the HMI consoles, an Image classifier using TensorFlow algorithms, created by Google was used. Is accomplished in a precise way the classification of different HMI consoles, the different bases without the respective console, and no object in the working station as you can seen at Figure 4.6.

TensorFlow is an open-source library created by Google that specializes in machine learning applications [46].

Building an image classifier from scratch is a colossal and daunting task. There are millions of things that need to be taken into consideration. Google has open-sourced one of its best image classifier models called the Inception. Inception is a DCNN trained for

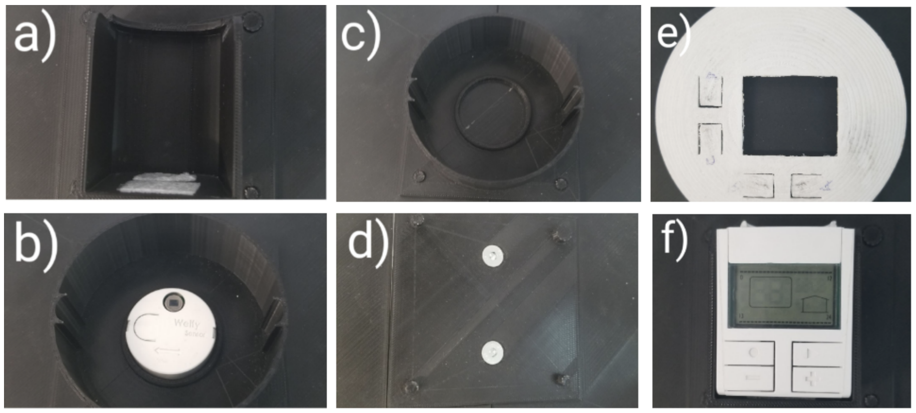


Figure 4.6: Type of classified objects a)HMI with LCD Display Stand B)One Button HMI c)One Button HMI Stand d)Empty station e)4 button Round HMI f)HMI with LCD Display

the image classification, which was trained on a staggering millions of images from the ImageNet database (see Figure 4.7 with thousand different categories [25]).

Figure 4.7: Example of images of dataset ImageNet [19]

Note that building a custom deep learning model demands extensive computation resources and a significant amount of training data. In this way, the proposed approach of using a previous trained network, brings several advantages, e.g., saving time, some reuse of the parameters that CNN has already learned allows to have a very precise classifier with much less training data. Reusing the previous trained models on different but related tasks is known as Transfer Learning, which reuse the initial and middle layers of the previous trained model to re-train only the final layers, as illustrated in Figure 4.8.

In a feedforward neural network, the neurons are organized in layers, with different

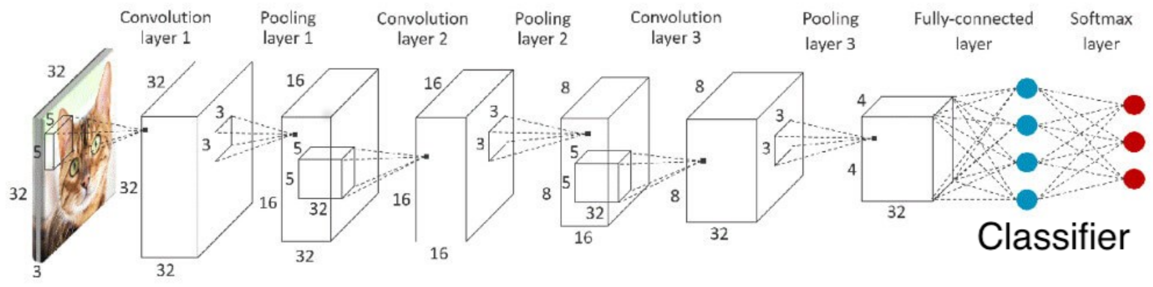


Figure 4.8: Deep Convolutional Neural Network (DCNN) for classification [2]

layers performing different kinds of transformations on their inputs. The signals travel from the first layer (input), to the last one (output), possibly after traversing the layers multiple times. As the last hidden layer, the "bottleneck" has enough summarized information obtained from the previous layers, to provide the next layer which does the actual classification task.

The reason why final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1000 classes in the ImageNet databases often also useful to distinguish between new kinds of objects. The training task for the image classifier is performed by using a database of 100 images of each type of HMI console, obtained by applying the dataset augmentation technique to a single image of each object using image processing algorithms [36].

The data augmentation is an automatic way to boost the number of different images in the database to train the Deep learning algorithms. The transformation functions for the data augmentation are the random rotation, the random noise, the horizontal flip and the blur, as shown in Figure 4.9.

The achieved experimental results, the softmax classifier, that is embedded in the DCNN of the TensorFlow algorithm, shown an accuracy higher than 97% in the correctness of the identification of the different typologies showed in Figure 4.6, in all 300 tests made. Compared with others solutions as image processing to detect different objects, the big advantage is the reduce time setup to classify new kind of objects, retraining only the network with new images, with no coding needed to explore image characteristics.

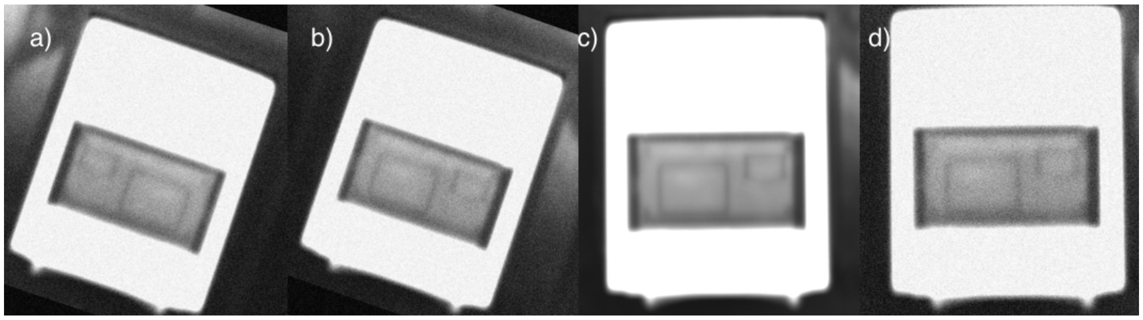


Figure 4.9: Data augmentation image transformation: a) rotate, b) flip, c) blur and d) noise.

This is a heavy algorithm, that takes about 5 seconds to perform the classification task, due to processing limitations, which can be faster if we upgrade the system with a more powerful processing unit.

4.3 LCD Display Errors Detection

Previously the camera obtains an image of a specific angle, in order to avoid reflection caused by the lighting system of the bench in the display, and to get the image with the highest possible contrast, illustrated in Figure 4.10.

According to Fig. 4.11, to detect display area to be analyzed and the errors or the missing parts of the LCD display, it's used the template matching technique using OpenCV libraries, witch finds areas of an image that are similar to a patch (template) [38].

Initially using this solution, with the normal variation of the environment light intensity, produced high variations in the detection of similarity between the template and the image. With a attempt to solve the problem, using exposure control (average of gray pixel information from image histogram to approximate the similarity of both images), the errors detection was not so accurate as expected.

The solution to increase the reliability, image processing techniques were used, using also the exposure control combined with Gaussian adaptive binarization in order to simplify the image, eliminating noise and highlight the objects to be detected from the display image, producing a clean image, as illustrated in Figure 4.12, with the advantage

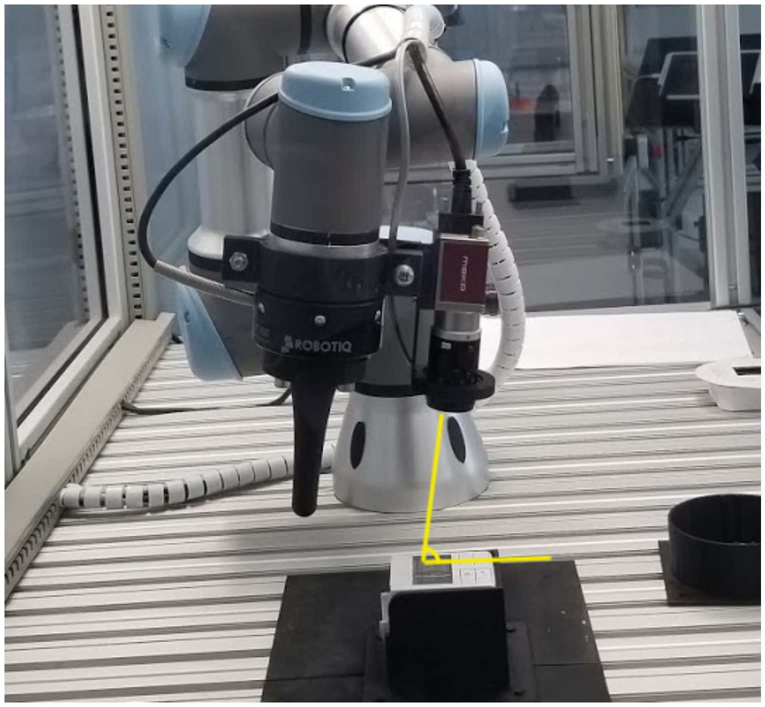


Figure 4.10: Display image acquisition

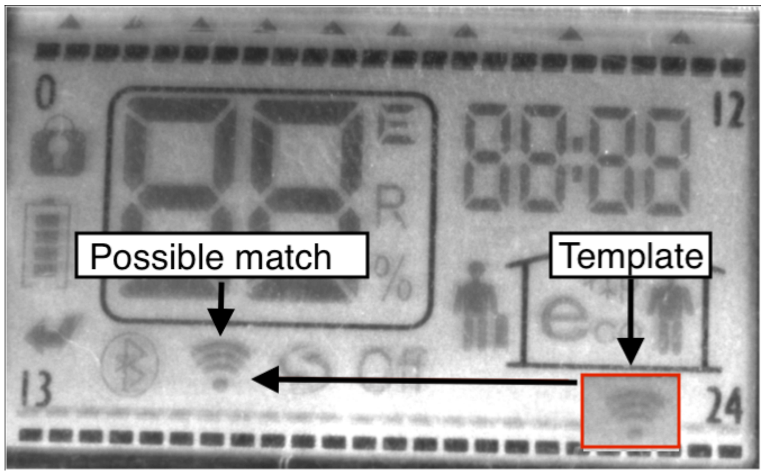


Figure 4.11: Display matching template example

of not being so susceptible to light variations.

A database was created with all the objects to be detected in the display, as represented in Fig. 4.13.

The template image (T) is slided over the source image (I), and is compared against overlapped image regions to find matches using statistics. The OpenCV function method

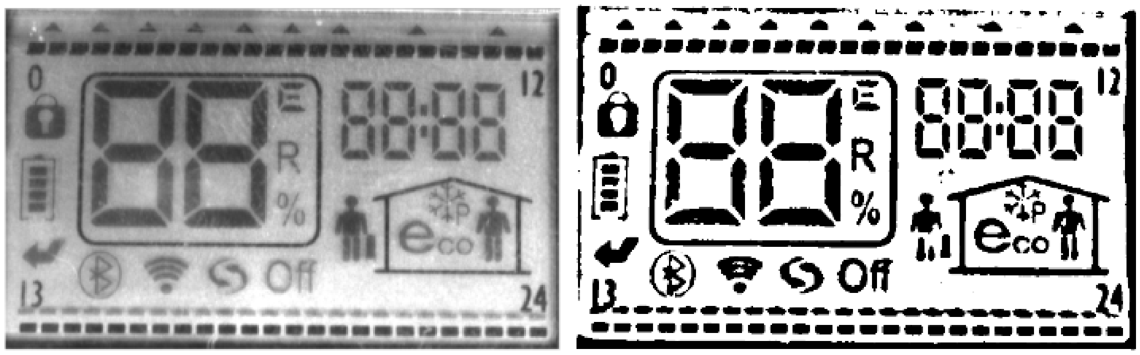


Figure 4.12: Original Image vs Processed Image

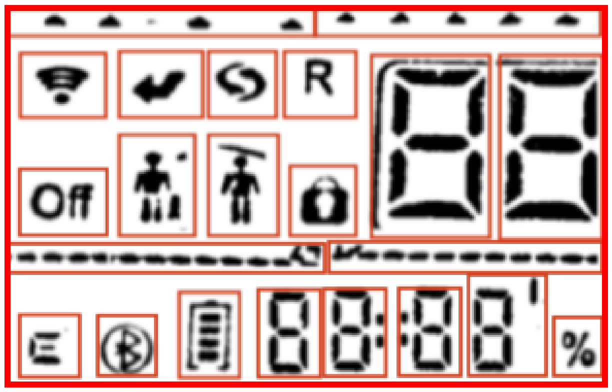


Figure 4.13: The 22 templates used for matching

used was "CV_TM_SQDIFF", that reached better results compared with other function available:

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \tag{4.1}$$

By sliding, we mean moving the patch one pixel at a time (left to right, up to down). At each location, a metric is calculated so it represents how “good” or “bad” the match at that location is (or how similar the patch is to that particular area of the source image). For each location of T over I, is stored the metric in the result matrix (R). Each location (x,y) in R contains the match metric.

Shown by Figure 4.14 is the result R of sliding the patch with a metric CV_TM_SQDIFF. The brightest locations indicate the highest matches. As you can see, the location marked

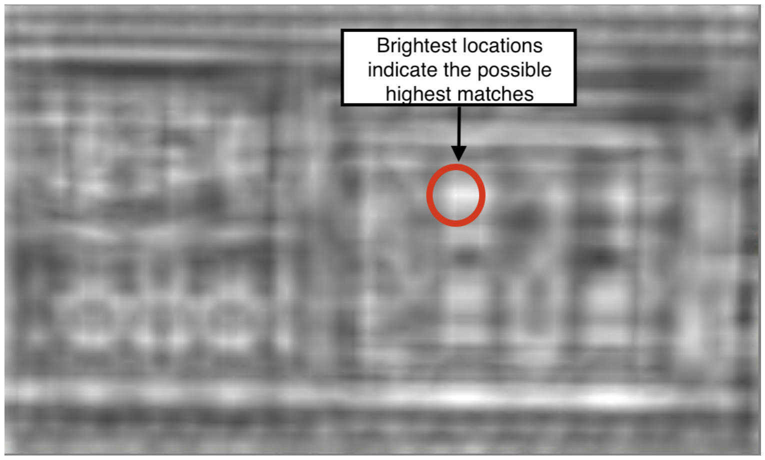


Figure 4.14: Match metric (R) results

by the red circle is an example of one with the highest value, so that location (the rectangle formed by that point as a corner and width and height equal to the patch image) is considered the match [32]. When an object is not detected, a red rectangle with the template size is marked in its position, indicating the position where the fault occurs. In case of the lower bars (green selection in the Figure 4.15), when an error occurs, the difference proportion between the input image and the bars template is very tiny, so it's very difficult to detect this kind of error with such low variation, sometimes this algorithm is not able to detect the issue. This algorithm takes about 2 seconds to perform the task. It could be faster if no previous exposure control of the acquired image needed. In more than 200 performed tests, for all other objects, this application obtained an accuracy of almost 100%.

4.4 Classification of Buttons

The detection of errors in the buttons of the HMI console was performed by using a kNN algorithm, to the signal obtained from the force sensor, when the robot is pressing the button (see Figure 4.16).

This algorithm is a supervised learning method that consists in calculating the distances between the variable to be classified and the elements in the dataset. These

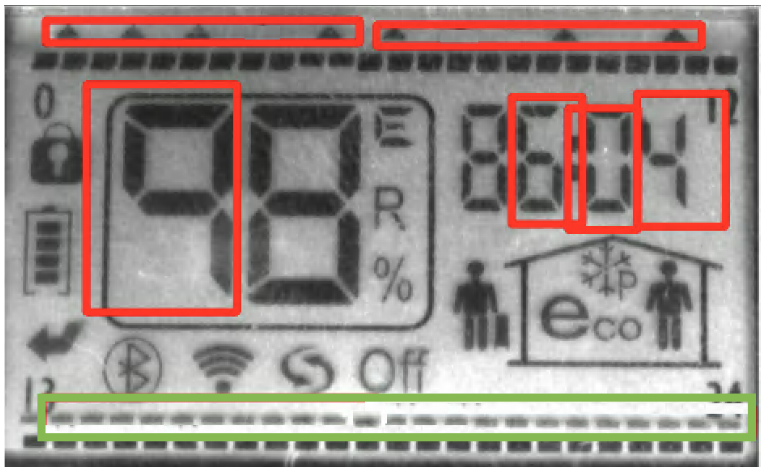


Figure 4.15: Match results of the LCD display.

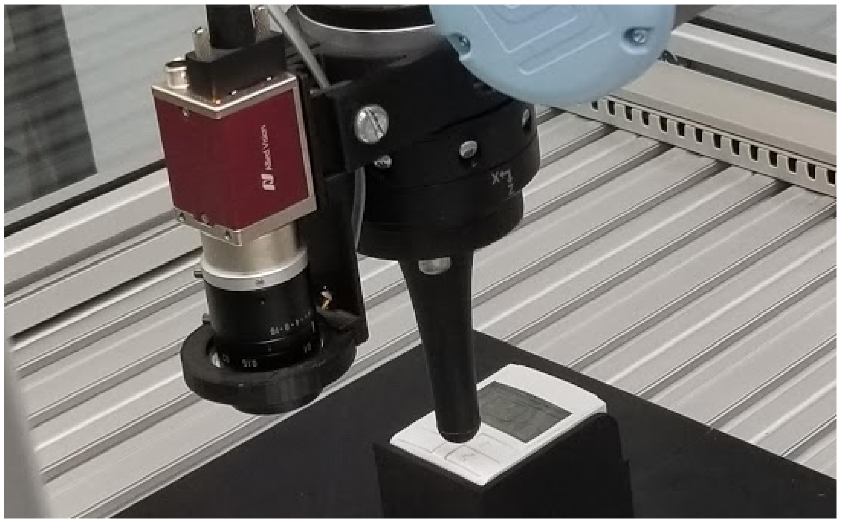


Figure 4.16: Robot pressing the buttons during the inspection task.

elements are labelled and the predicted class is given by the more voted class inside the k nearest elements subset. The algorithm learned the differences between these button's conditions with a dataset X with 700 inputs, through the test of different buttons, with and without anomalies to perform the classification between "Ok", "Not Ok" and "Rigid" of the new data obtained by the torque-force sensor.

Each input data is obtained through the action of pressing each button four times, being extracted and analyzed 7 different features, namely the maximum value (i.e. the highest force recorded), mean (i.e. the average value of the force), Root Mean Square

(RMS) (i.e. the magnitude of applied force), mean of derivative (i.e. the average value of the variation), RMS gradient (i.e. the magnitude of the applied force variation), maximum gradient (i.e. the largest positive variation) and minimum gradient (i.e. the greatest negative variation).

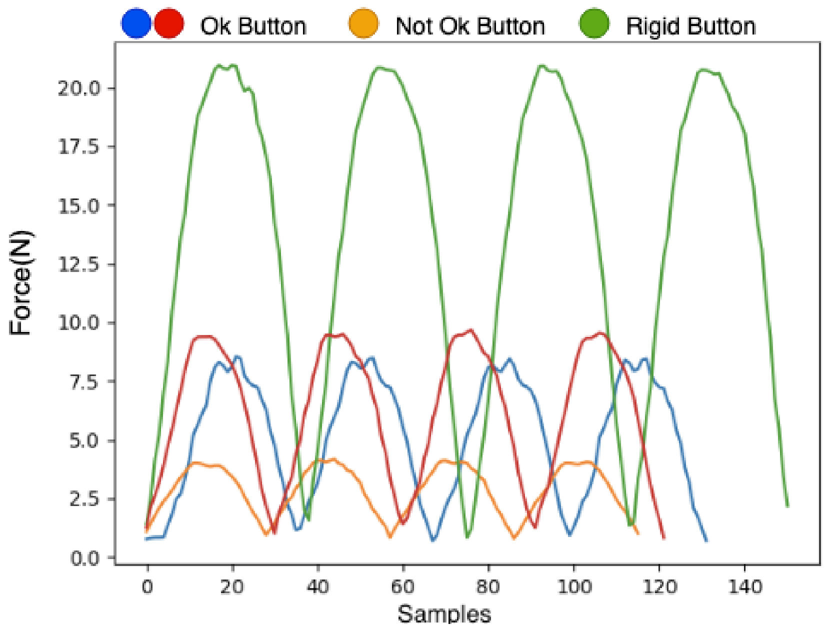
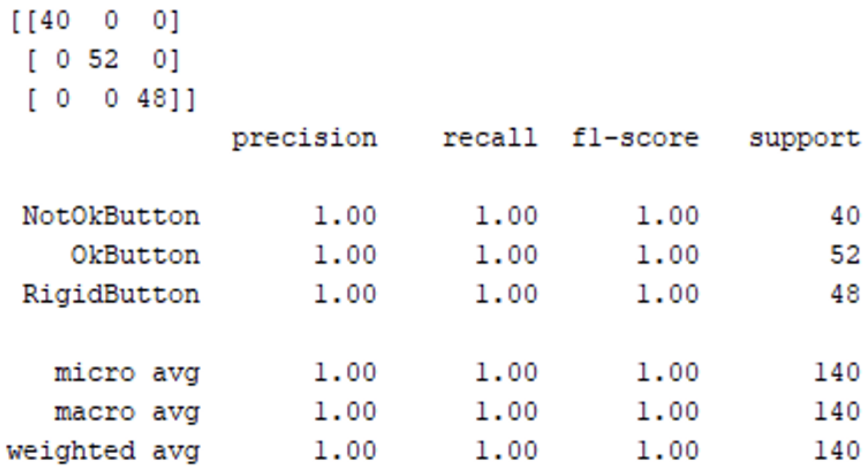


Figure 4.17: Real force data from the pressure action over buttons.

Figure 4.17 represents the data obtained from the force sensor Y in which it can be seen that the behavior has 4 peaks, representing the 4 actions of pressing the buttons. From these curves, the features to classify the button condition can be extracted. For the classification task, was also used an kNN classifier.

To test the performance of the kNN classifier, was used the confusion matrix method. Confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The implemented algorithm presented 100% in all performance measures, as you can see at Figure 4.18.

More than 300 tests were performed using new data, without fail in the correct classification of the state of the buttons, demonstrating a very high performance, being safe for an implementation in real context, with the advantage of being a very fast algorithm, that takes only 50 ms of processing time.



```
[[40  0  0]
 [ 0 52  0]
 [ 0  0 48]]
              precision    recall  f1-score   support

 NotOkButton       1.00      1.00      1.00        40
    OkButton       1.00      1.00      1.00        52
 RigidButton       1.00      1.00      1.00        48

   micro avg       1.00      1.00      1.00       140
   macro avg       1.00      1.00      1.00       140
weighted avg       1.00      1.00      1.00       140
```

Figure 4.18: kNN performance measures using confusion matrix

Chapter 5

Conclusions

This work describes the application of ML algorithms in a robotized artificial vision inspection station, to increase the flexibility and adaptability of the system to accomplish the QC tasks to a diversified set of HMI consoles. The adopted solution uses a deep CNN algorithm to perform the operator's facial recognition and the console detection, which allows to have a adaptable solution, with possibility to add new operators or different kind of HMI consoles to be inspected, in a simple and fast way, being only necessary to retrain final layer of the CNN.

The proposed approach shows the application of IA techniques to robotized inspection systems, supported by artificial vision systems, with applicability in several tasks of QC, increasing the reliability and optimization of the processes, reducing the costs and allowing to perform these kind of tasks with no human interference. With the use of larger amounts of data, the retraining of the ML algorithms will contribute to increase the accuracy of the face recognition and consoles detection algorithms, to values near of 100%.

For the detection of defects in the LCD displays, the adopted solution uses image processing techniques to perform the image correction and to reduce the impact that luminosity changes have in the image, combined with the template matching technique, which finds the areas of the image that are similar to the pre-defined template. In case of the lower bars of the display when an error occurs,the difference proportion between the input image and the bars template is very tiny, so it's very difficult to detect this kind of error with such low variation, sometimes this algorithm is not able to detect the issue. For all other objects, this application has an accuracy of almost 100% in all tests made.

For the correct classification of the buttons condition, was used a kNN algorithm, being previously trained with several anomalies examples, to be compared to the data obtained by the torque-force sensor. The applied algorithms showed promising results, with high accuracy values in the QC tasks, demonstrating the robustness and efficiency of these solutions to be applied in a real industry context.

In general, this robotized quality control station, with an artificial vision combined with ML techniques, has allowed a substantial increase in the accuracy and reliability of the inspection tasks as expected, with an increase in the adaptability of the system to new circumstances.

Adopting Python as the programming language to develop the application, it was possible to work quickly and integrate systems more effectively, developing a system capable of performing tasks in a fluid and fast way, with fast processing time of the respective algorithms, which can be even faster if we equip the system with a more powerful processing unit.

Future work is devoted to improve the detection of errors in the LCD displays by adopting another ML techniques, combined with the improvement of the image acquisition conditions, e.g., controlling automatically the luminosity of the inspection environment, and the noise in the acquired image, which could eventually help to solve the problem of non-detection of small faults in the LCD displays. Improve the algorithm in the detection of object exchange, and initialization of new inspection, in order to make it smarter, avoiding possible failures in a possible slower exchange of the console to be inspected.

Bibliography

[1] male-worker-quality-control-inspection-of-metal-part-in-a-factory, innovation professional placement. URL: https://innovationpp.net/home/male-worker-quality-control-inspection-of-metal-part-in-a-factory/.

[2] Tutorial: How to deploy convolutional NNs on Cortex-M - Processors blog - Processors - Arm Community. URL: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/deploying-convolutional-neural-network-on-cortex-m-with-cmsis-nn.

[3] Kaled Mohamed Almhdi, Paolo Valigi, Vidas Gulbinas, Rainer Westphal, and Rainer Reuter. Classification with artificial neural networks and support vector machines: application to oil fluorescence spectra. 2006. URL: http://209.197.92.148/apps/files/Oil{_}Classification{_}with{_}ANNs{_}and{_}SVMs{_}2006-06-07{_}kmai.pdf.

[4] G Chryssolouris, D Mavrikios, N Papakostas, D Mourtzis, G Michalos, and K Georgoulias. Digital manufacturing: History, perspectives, and outlook. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 223(5):451–462, may 2009. doi:10.1243/09544054JEM1241.

[5] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE. URL: http://ieeexplore.ieee.

org/document/1467360/, doi:10.1109/CVPR.2005.177.

[6] Davis E. King. dlib C++ Library. URL: http://dlib.net/.

[7] O. Déniz, G. Bueno, J. Salido, and F. De la Torre. Face recognition using Histograms of Oriented Gradients. *Pattern Recognition Letters*, 32(12):1598–1603, sep 2011. doi:10.1016/J.PATREC.2011.01.004.

[8] James R. (James Robert) Evans and William M Lindsay. *The management and control of quality*. Minneapolis/St. Paul : West Pub. Co, 3rd ed edition, 1996. Includes index. URL: http://books.google.com/books?id=SI8bAQAAMAAJ.

[9] FT300 Universal Robots. Robotiq Force Torque Sensor (FT 300) | Active8 Robots. URL: https://www.active8robots.com/shop/force-torque-sensor-ft-300/.

[10] Fujitsu Forum Munich. Artificial Intelligence Solution from Fujitsu Helps Siemens Gamesa Significantly Accelerate Quality Assurance Procedures - Fujitsu CEMEA&I, 2017.

[11] Fujitsu Laboratories of Europe Ltd. Fujitsu Develops State-of-the-Art AI Solution to Revolutionize Non-Destructive Testing Manufacturing Inspection - Fujitsu CEMEA&I, 2017. URL: http://www.fujitsu.com/fts/about/resources/news/press-releases/2017/emeai-20171002-fujitsu-develops-state-of-the-art-ai.html.

[12] Ross Girshick. Fast R-CNN. Technical report. arXiv:1504.08083v2.

[13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL: http://arxiv.org/abs/1311.2524, arXiv:1311.2524.

[14] Courville A Goodfellow I, Bengio Y. Deep Learning - MIT. 2016. arXiv:arXiv:1312.6184v5, doi:10.1038/nmeth.3707.

[15] M. Gordan, C. Kotropoulos, and I. Pitas. Application of support vector machines classifiers to visual speech recognition. 2003. `doi:10.1109/icip.2002.1038921`.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[18] T S Huang. Computer Vision: Evolution and Promise. Technical report. URL: `https://cds.cern.ch/record/400313/files/p21.pdf`.

[19] Imagenet. Convert Full ImageNet Pre-trained Model from MXNet to PyTorch. URL: `https://blog.paperspace.com/convert-full-imagenet-pre-trained-model-from-mxnet-to-pytorch/`.

[20] Jon Fingas. Google AI could keep baby food safe, 2017. URL: `https://www.engadget.com/2017/07/25/google-ai-helps-make-safer-baby-food/`.

[21] M Jordan, J Kleinberg, and B Schölkopf. Pattern Recognition and Machine Learning. Technical report. URL: `http://users.isr.ist.utl.pt/{~}wurmd/Livros/school/Bishop-PatternRecognitionAndMachineLearning-Springer2006.pdf`.

[22] Patrik Kamencay, Miroslav Benco, Tomas Mizdos, and Roman Radil. A New Method for Face Recognition Using Convolutional Neural Network. *Advances in Electrical and Electronic Engineering*, 15, 2017. `doi:10.15598/aeee.v15i4.2389`.

[23] Michal Kawulok, M. Emre Celebi, and Bogdan Smolka, editors. *Advances in Face Detection and Facial Image Analysis*. Springer International Publishing, Cham, 2016. `doi:10.1007/978-3-319-25958-1`.

[24] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press. URL: `http://dl.acm.org/citation.cfm?id=1566770.1566773`.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL: `http://arxiv.org/abs/1512.02325`, `arXiv:1512.02325`.

[27] David G Lowe. Object Recognition from Local Scale-Invariant Features. Technical report, 1999. URL: `https://www.cs.ubc.ca/{~}lowe/papers/iccv99.pdf`.

[28] Jiwen Lu, Junlin Hu, and Jie Zhou. Deep metric learning for visual understanding: An overview of recent advances. *IEEE Signal Processing Magazine*, 34(6):76–84, 2017.

[29] Grégoire Montavon, Wojciech Samek, and Klaus Robert Müller. Methods for interpreting and understanding deep neural networks, 2018. `doi:10.1016/j.dsp.2017.10.011`.

[30] Kevin P Murphy. *Machine Learning A Probabilistic Perspective*. URL: `http://mitpress.mit.edu`.

[31] P. D. T. O'Connor. Introduction to Statistical Quality Control (2nd edition), D. C. Montgomery, Wiley, 1991. Number of pages: 702. £49.35, Paperback £17.50. *Quality and Reliability Engineering International*, 2007. `doi:10.1002/qre.4680070316`.

[32] Opencv. Template Matching — OpenCV 2.4.13.7 documentation. URL: `https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template{_}matching/template{_}matching.html`.

[33] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. URL: `http://arxiv.org/abs/1511.08458`, `arXiv:1511.08458`.

[34] Caroline Liboreiro Paiva. Quality Management: Important Aspects for the Food Industry. In Innocenzo Muzzalupo, editor, *Food Industry*, chapter 9. IntechOpen, Rijeka, 2013. `doi:10.5772/53162`.

[35] Rashmi Pandey, Sapan Naik, and Roma Marfatia. Image Processing and Machine Learning for Automated Fruit Grading System: A Technical Review. Technical Report 16, 2013. URL: `https://pdfs.semanticscholar.org/aa50/c9a262d58ba876b3be2e1a0472a2dc799277.pdf`.

[36] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[37] Kevin L. Priddy and Paul E. Keller. *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68).* SPIE- International Society for Optical Engineering, 2005.

[38] Python. Object detection with templates Python Tutorial. URL: `https://pythonspot.com/object-detection-with-templates/`.

[39] R. Berwick. An Idiot's guide to Support vector machines (SVMs) R. Berwick, Village Idiot SVMs: A New Generation of Learning Algorithms. Technical report. URL: `http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf`.

[40] B. Ribeiro. Support vector machines for quality monitoring in a plastic injection molding process. *Trans. Sys. Man Cyber Part C*, 35(3):401–410, August 2005. URL: `http://dx.doi.org/10.1109/TSMCC.2004.843228`, `doi:10.1109/TSMCC.2004.843228`.

[41] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors*, 16(8), 2016. URL: `http://www.mdpi.com/1424-8220/16/8/1222`, `doi:10.3390/s16081222`.

[42] Nicu Sebe, Ira Cohen, Ashutosh Garg, and Thomas S. Huang. Machine learning in computer vision. In *Computational Imaging and Vision*, 2005.

[43] Walter A. (Walter Andrew) Shewhart and 1900 Deming, W. Edwards (William Edwards). *Statistical method from the viewpoint of quality control.* New York : Dover, 1986.

[44] Richard Szeliski. Computer Vision: Algorithms and Applications. Technical report, 2010. URL: `http://szeliski.org/Book/`.

[45] Yichuan Tang. Deep learning using support vector machines. *CoRR*, abs/1306.0239, 2013. `arXiv:1306.0239`.

[46] TensorFlow. TensorFlow. URL: `https://www.tensorflow.org/`.

[47] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. URL: `https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013`.

[48] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, Berlin, Heidelberg, 1995.

[49] Dirk Walther, Laurent Itti, Maximilian Riesenhuber, Tomaso Poggio, and Christof Koch. Attentional Selection for Object Recognition-a Gentle Way. Technical report. URL: `http://www.klab.caltech.eduhttp//www.ai.mit.edu/projects/cbcl`.

[50] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, jan 2008. URL: `http://link.springer.com/10.1007/s10115-007-0114-2`, `doi:10.1007/s10115-007-0114-2`.

[51] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. pages 818–833. Springer, Cham, 2014. URL: `http://link.springer.com/10.1007/978-3-319-10590-1{_}53`, `doi:10.1007/978-3-319-10590-1_53`.

[52] Ray Y. Zhong, Xun Xu, Eberhard Klotz, and Stephen T. Newman. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering*, 3(5):616–630, oct 2017. `doi:10.1016/J.ENG.2017.05.015`.

[53] Andreas Ziegler. An Introduction to Statistical Learning with Applications. R. G. James, D. Witten, T. Hastie, and R. Tibshirani (2013). Berlin: Springer. 440 pages, ISBN: 978-1-4614-7138-7. *Biometrical Journal*, 2015. `doi:10.1002/bimj.201500224`.

Appendix A

Original Project Proposal

Proposta de Dissertação / Projeto
Mestrado em Engenharia Industrial
2018/2019

Título Provisório: Aplicação de Técnicas de Aprendizagem de Máquina em Tempo Real para Inspeção Inteligente de Consolas HMI

Aluno de mestrado: Luís Variz

Orientador: Paulo Leitão

Co-Orientador: Pedro João Rodrigues

Principais objetivos a atingir:
Na era da 4ª revolução industrial, a introdução de inteligência artificial em ambientes industriais de automação assume crucial importância. Em particular a inteligência artificial permite implementar análise avançada de dados de forma a suportar as tarefas, entre outras, de monitorização, diagnóstico, previsão e otimização.
Tendo por base um trabalho anterior em inspeção robótica inteligente de consolas HMI (Human-Machine Interface), o trabalho proposto consiste na aplicação de diversas técnicas/ferramentas de aprendizagem de máquina visando a análise de diferentes componentes (botões e identificadores luminosos) de diferentes tipos de consolas HMI suscetíveis a erros de produção. Os classificadores deverão atuar em tempo real com uma taxa de precisão elevada.

Resultados esperados:
Pretende-se com esta proposta aplicar técnicas de aprendizagem de máquina no desenvolvimento de modelos de identificação e classificação de suporte à inspeção inteligente de consolas HMI. Este trabalho está inserido nas atividades investigação, desenvolvimento e inovação do projeto designado por I4.0@TMAD (Promoção da Indústria 4.0 na Região de Trás-os-Montes e Alto Douro) financiado pelo Norte2020.

Caracterização do trabalho
No âmbito do projeto I4.0@TMAD, que visa a promoção da Indústria 4.0 na Região de Trás-os-Montes e Alto Douro, foi construída uma bancada robótica para inspeção inteligente e adaptativa com o objetivo de demonstrar as potencialidades da robótica inteligente na realização de tarefas de inspeção industrial. Esta bancada consiste, entre outros, num robô manipulador UR3, um sensor de força que mede a força com que um êmbolo acoplado ao braço robótico pressiona um botão e uma câmara para aquisição de imagem (consola e indicadores luminosos). A principal funcionalidade da bancada é a realização do controlo de qualidade de consolas HMI, que para tal requer a execução das seguintes tarefas:

- Identificação do tipo de consola HMI que está a ser analisada (usando a imagem obtida da câmara).
- Identificação dos botões que não estejam funcionais assim como aqueles cujas forças de atuação demonstrem não serem as estipuladas e futuramente poderem apresentar deficiências (usando os dados obtidos pelo sensor de força).
- Identificação se os indicadores luminosos de displays de TFT e LCDs estão funcionais (usando a imagem obtida da camara).

O presente trabalho visa o desenvolvimento de modelos de identificação e classificação para cada uma 3 tarefas referidas anteriormente usando técnicas de aprendizagem de máquina. Assim:

- Usando imagens obtidas da câmara industrial deve ser realizada a classificação de diferentes consolas HMI, a localização de displays nas consolas e a deteção de falhas nos componentes existentes no display.

- Usando os dados obtidos do sensor de força deve ser realizada a classificação dos botões de pressão entre funcionamento correto ou defeituoso.

Pretende-se que a classificação seja realizada em tempo real e com uma taxa de precisão elevada (a rapidez da identificação é uma questão essencial a explorar).

Calendarização das fases do trabalho:
O desenvolvimento da presente proposta de trabalho será realizado através da execução das seguintes etapas:

1. Familiarização com a bancada robótica, câmara industrial e sensor de força (M1)
2. Estudo de técnicas de aprendizagem de máquina (M1 – M2)
3. Construção de um dataset com imagens das consolas (obtidos a partir da câmara) e dados dos botões (obtidos a partir do sensor de força) (M2)
4. Desenvolvimento de modelos de aprendizagem de máquina para a localização, deteção e classificação (M3 – M6)
5. Avaliação dos modelos e desenvolvimento de um protótipo para a inspeção inteligente de consolas HMI (M6 – M8)
6. Testes e validação do protótipo (M8 – M10)
7. Escrita da dissertação e defesa final do trabalho (M10 - M11)

Palavras-chave: aprendizagem de máquina, visão artificial, manipulador robótico, Indústria 4.0.

Infraestruturas a utilizar:
Está prevista a utilização dos seguintes recursos:

- Manipulador robótico equipado com sensor de força e câmara industrial para aquisição de imagem.
- Diversa literatura relativa a técnicas de aprendizagem de máquina.

Declaro que aceito realizar este projeto de dissertação segundo o plano apresentado

____
(O aluno de mestrado)

Declaro que aceito orientar a preparação deste projeto de dissertação segundo o plano apresentado

____
(O orientador)

Declaro que aceito orientar a preparação deste projeto de dissertação segundo o plano apresentado

____
(O co-orientador)

Appendix B

Python code

Link for Gitlab, with the python code applied:

https://gitlab.estig.ipb.pt/a32957/adaptive-robotic-inspection-station.git