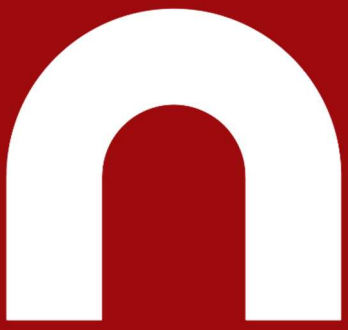


isec
Engenharia

MESTRADO EM INFORMÁTICA E SISTEMAS

DEFINITIVO

Real-Time Quality Control of Heat Sealed Bottles

Autor

Samuel Silva da Cruz

Orientadores

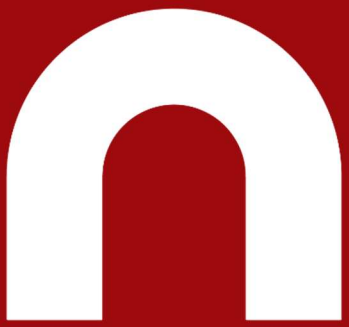
João António Pereira Almeida Durães

Mateus Daniel Almeida Mendes

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, dezembro 2021



isec
Engenharia

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

Real-Time Quality Control of Heat Sealed Bottles

Relatório de Trabalho de Projeto para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Tecnologias da Informação e do Conhecimento.

Autor

Samuel Silva da Cruz

Orientadores

João António Pereira Almeida Durães

Mateus Daniel Almeida Mendes

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, dezembro 2021

Acknowledgments

During this long journey I was helped by many people and, without them I would have never reached where I am now.

I would like to thank...

- Professor Mateus Mendes for giving me this opportunity and the guidance and knowledge he taught me during this journey.
- Professor João Durães for the guidance and knowledge he taught me during this journey.
- Professor António Paulino for giving me this opportunity and the knowledge he taught me during this journey.
- Professor Jorge Almeida who was not directly involved in this project but gave me the opportunity to start a new academical path which brought me here.
- André Vicente for helping me multiple times across this journey.

Finally I would like to thank my family and my friends for always supporting me.

Abstract

The present document describes a system for controlling the quality of heat sealed bottles. The system detects defective seals to identify bottles that can not be sold. A prototype was developed to validate and test the system proposed.

In the production line, the bottles are filled with a toxic substance and can only be sold when properly sealed. A leak can be harmful to humans and the environment. Because the seals are not visible from outside the bottle, images from each seal are obtained using a thermal camera. The hot glue used in the sealing process makes the seal visible in the infrared image. The image is cleaned and converted to black and white only keeping the seal in the final image. Black pixels present the value 0 and white pixels present the value 1. Then a signature composed by two arrays containing the sum of the number of white pixels in each column and in each row is calculated. Both arrays present a U shape when the bottle is sealed. The signature is then fed to an artificial neural network which was trained to identify correctly sealed bottles. The classification results are stored in a database. The trained neural net presented an *accuracy* of 98.7 % and an *F1 score* of 96.0 % in the testing phase.

The results shows the inspection process is effective in identifying defective seals and because it is automated it can be scaled up to large bottle processing plants. All classified images can be seen though a web application where a user has the option of validating the operation and identifying errors which will be individually fitted to improve the machine learning model performance.

The system is non invasive, automated, and can be applied to common conveyor belts currently used in industrial plants. It can also be adapted to detect different problems in bottles of different shapes.

Keywords: quality-control; machine learning; computer vision; thermal images; artificial neural networks

Resumo

Nesta dissertação é descrito um sistema de controlo de qualidade de selos em garrafas. Foi contruído um protótipo com o objetivo de testar e validar o funcionamento do sistema.

Na linha de produção, as garrafas são cheias com uma substância tóxica e apenas podem ser vendidas quando corretamente seladas pois uma fuga põe em risco a saúde do utilizador.

A dificuldade deste processo deve-se ao facto de o selo não ser visível pois encontra-se debaixo da tampa opaca da garrafa. Dado o uso de cola quente no processo de selagem, com uma câmara térmica é possível obter uma imagem do selo. Esta imagem é depois processada com o intuito de isolar o selo na imagem final. Da imagem final gera-se uma assinatura que consiste na junção de duas listas contendo a soma do número de pixels brancos por coluna e por linha. Ambas as listas apresentam uma forma de 'U' quando a garrafa está corretamente selada.

Uma rede neuronal utiliza a assinatura para classificar a imagem, indentificando garrafas mal seladas. O resultado obtido é registado numa base de dados. A rede neuronal treinada apresentou uma *accuracy* de 98,7 % e um *F1 score* de 96,0 % na fase de treino mostrando que é eficiente na identificação de selos defeituosos.

O sistema inclui a possibilidade de validar as classificações usando uma aplicação web onde é possível analisar o histórico de imagens. Quando uma imagem incorretamente classificada é identificada, esta deve ser selecionada e novamente treinada para corrigir o erro e permitir que o modelo tenha capacidade de aprendizagem.

Este método não é invasivo nem destrutivo, é automatizado e pode ser usado na produção de produtos diferentes desde que o processo de selagem seja semelhante.

Palavras-Chave: controlo da qualidade; machine learning; visão por computador; redes neuronais

Contents

List of Figures

List of Tables

Acronyms

ANN - *Artificial Neural Network*
API - *Active Pharmaceutical Ingredients*
AS/RS - *Automated Storage and Retrieval System*
BFGS - *Broyden-Fletcher-Goldfarb-Shanno*
BGR - *Blue, Green and Red*
BSD - *Berkeley Software Distribution*
CNN - *Convolution Neural Network*
CPMS - *Cyber-Physical Manufacturing Systems*
CPU - *Central Processing Unit*
FIFO - *First In First Out*
GPU - *Graphics Processing Unit*
HSV - *Hue, Saturation and Value*
HTML - *HyperText Markup Language*
HTTPS - *Hypertext Transfer Protocol Secure*
HTTP - *Hypertext Transfer Protocol*
IIoT - *Industrial Internet of Things*
IR - *Infrared*
ISEC - *Instituto Superior de Engenharia de Coimbra* - Coimbra Engineering Academy
ITWI - *Infrared Thermal Wave Imaging*
IT - *Information Technology*
IoT - *Internet of Things*
LBFGS - *Limited-memory BFGS*
LDA - *Linear Discriminant Analysis*
MLP - *Multi-Layer Perceptron*
ML - *Machine Learning*
MMTC - *Massive Machine Type Communication*
ODBC - *Open Database Connectivity*
PLC - *Programmable Logic Controller*
RGB - *Red, Green and Blue*
SGD - *Stochastic Gradient Descent*
SQL - *Structured Query Language*
SSL TLS - *Secure Sockets Layer Transport Layer Security*
SVM - *Support Vector Machines*
TCP - *Transmission Control Protocol*

UCI - *University of California at Irvine*
URLLC - *Ultra-Rand Low Latency Communication*
USB - *Universal Serial Bus*
VPN - *Virtual Private Network*
YOLO - *You Only Look Once*
eMBB - *Enhance Mobile Broadband*

Chapter 1

Introduction

This project describes an approach to assess the quality of seals of bottles that contain pesticide. Infrared technology is used to retrieve images of the seals and a machine learning model is responsible for evaluating the seal condition, indicating if the bottle can be sold or should be removed.

1.1 Context

Industry 4.0 [1] is a large growing concept present in modern daily life. Computers and machines are now able to communicate with each other in order to exchange data and process it to produce information otherwise hard to find. The processing power of computers nowadays allows them to make decisions faster to help improving the organization productivity. Quality control is one of the sectors to benefit from the advances in machine learning.

The control of the quality of bottle sealing is a very important step during bottled products' processing line to assure that the products are safe to transport and store. This applies to numerous products, including food, beverages, detergents, disinfectants and pesticides among other examples. The assurance of the seal effectiveness is particularly relevant when products are potentially harmful to humans or the environment. Thus, processes to automate the bottle seal quality are very important and many techniques have been proposed, using various approaches.

The present document describes a prototype of a system to control the quality of seals in bottles that contain pesticide. The prototype was developed at the Coimbra Engineering Academy (ISEC) and the Institute of Systems and Robotics in partnership with Tula Labs, a company with headquarters in Lousã, Portugal.

[1] Industry 4.0 definition by IBM (last access on 2021-07-08): `https://www.ibm.com/topics/industry-4-0`

Figure 1.1: Type of bottles used in the process.

1.2 Problem

Organizations who contain a sealing stage on their production line face a generic problem which is assuring the quality of the seal to prevent leaks. In this project, an organization that produces bottles containing pesticide must ensure the quality of the cap sealing. The sealing process must be correctly done otherwise could endanger humans and the environment. Figure 1.1 presents the type of bottles used in this process [2]. The bottle in the figure is different from the ones used in the process but have a similar structure. For privacy concerns, the original bottles used in the process will not be shown. This figure demonstrates the seal fixed to the bottle opening and the opaque white lid which was removed to show the seal. Once sealed, the opaque lid hides the seal which makes it impossible to visually verify the seal condition without removing the lid as the figure demonstrates.

1.3 Proposed Approach

The sealing is done using hot glue in a production line. An InfraRed (IR) camera is used to take images from a conveyor belt that moves freshly sealed pesticide bottles. A

[2] Image retrieved from (last access on 2021-04-10): `https://www.ebottles.com/showcap-familyid-1417-kw-FINE_RIBBED_WITH_PRESSURE_SENSITIVE_INNER_SEAL_WHITE.htm`

single frame of each bottle is stored and the bottle cap must be clearly visible on it.

The image is then analyzed and edited to highlight the circle corresponding to the hottest area of the cap. The image is cleaned and converted to black and white. After being cleaned of the background noise, the sums of the horizontal and vertical image lines are created to assemble an image signature. Then the signatures are fed to an Artificial Neural Network (ANN). The output of the classifier determines the acceptance/rejection of that particular bottle. This approach assesses the quality of the sealing based on classifying the geometry of the circle imprinted in the thermal image.

Bottles having defective seals are identified as they leave the sealing stage and removed later in the conveyor belt where they are being moved. This approach can be implemented using commonly available technology and components and has low operational costs.

The results of the classification are stored and can be analyzed later by a user to inspect the results and identify mistakes (incorrectly classified images) done by the classifier. The user is also given the option to retrain those mistakes in order to correct the model and improve its performance over time. It can also adjust to different shapes of bottles.

1.4 Goals and Scope

The goal of this project is to propose a system that can fully solve the described problem. The system should acquire images of the seals without damaging the bottles, prepare the images to be classified and classify them. All these tasks should be automatic and require no human interaction, if possible.

A machine learning model should be trained to differentiate defectively sealed bottles from properly sealed bottles. Incorrect classifications should be identified and used to improve the machine leaning model performance creating a model that keeps evolving.

A prototype of the system should be developed and used to validate the system. The complete architecture of the prototype should be detailed allowing its use for experimental purposes.

This project can be generically split in three subjects: image processing, machine learning and system implementation.

The main scope of this project includes the video processing in order to retrieve images of seals which will then be used to create a dataset and the training and testing of the machine learning algorithm.

The complete system architecture should be proposed but, since the system will not be used in a real shop floor environment, the prototype can be simple which means the design, security and generic application functionalities – such as login, logout, and user management actions – are not essential to the scope.

The project scope does not include the implementation and testing of the prototype in a real company shop-floor. It does not include the configuration of the infrared camera neither the configuration of a Programmable Logic Controller which will act as a rejection system to remove the unsealed bottles.

1.5 Dissertation Outline

In Chapter 2 the problem is detailed, a method to solve it is introduced and the requirements are presented.

In Chapter 3, a review of the literature is done presenting a list of methods already used to solve similar problems. Chapter 4 presents the architecture of the system and in Chapter 5 the data is analyzed. In Chapter 6 the machine learning model is trained, optimized and tested and in Chapter 7 the prototype development is described.

The results are discussed in Chapter 8 and in Chapter 9 the conclusions and future work are presented.

Chapter 2

Case Study, Methods and Requirements

In this chapter, we start by describing the industrial process that originates this problem and then the methodology used to achieve the goal of this dissertation – developing a quality control system to assess the quality o bottle seals – is defined. The chapter ends with the requirements necessary for the implementation of the solution.

2.1 Case Study

The bottles are filled and sealed on a conveyor belt following a process common to most mass production lines.

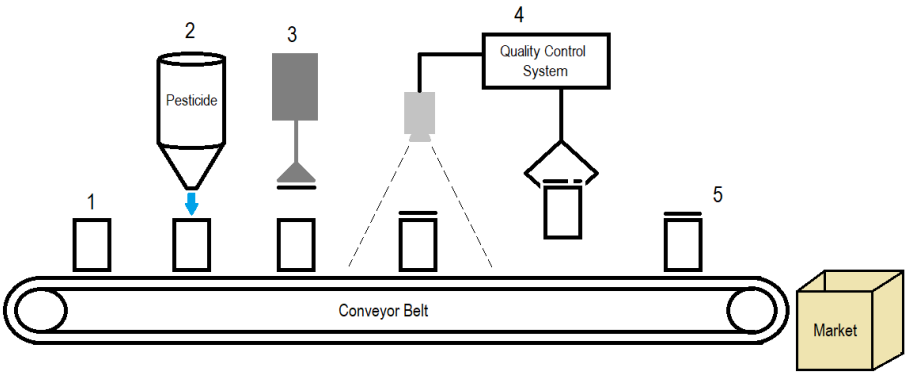


Figure 2.1: Production line process to be studied.

Figure 2.1 presents a simple scheme of the process.

1. The open bottle (represented by the rectangle) is placed on top of the conveyor belt which will move it through all steps of the process.

2. The first stop is a liquid filling machine where the bottle is filled with a given substance (in this case it is a toxic pesticide).

3. The filled bottle goes through a capping machine, where a robotic arm places and attaches the cap to the bottle. Inside the cap there is a thin aluminium circle covered with previously melted glue which will seal the bottle once the cap is closed. This aluminium seal is the target of the quality control system developed in this thesis.

4. The capped bottle goes through the quality control system that will be developed in this thesis. It includes a camera to retrieve images of the seal, a system to classify the image and a robotic hand to remove the unsealed bottles.

5. Finally, if the bottle was classified as sealed it can be packed and sold.

Figure 2.1 is a very simple image to generically explain the stages of the production line. Defective seals can not be easily seen as shown in the Stage 4 of this image and Figure 2.2 further explains the sealing process.

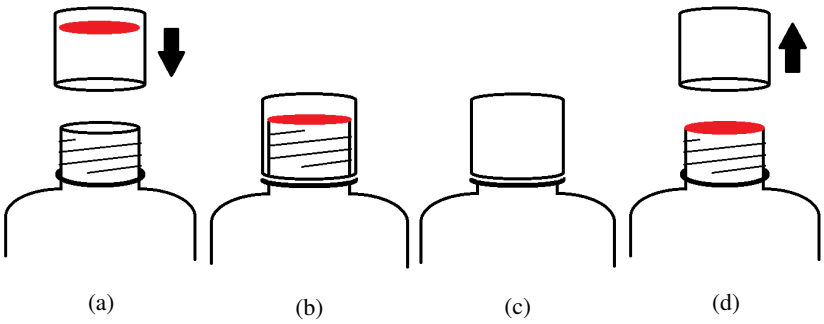


Figure 2.2: Details of the sealing process. a) The plastic cap contains an aluminium sheet with glue. b) After the bottle is capped, the aluminium circle is glued to the bottle opening. c) Final result of the process where the bottle is expected to be sealed and the cap is closed hiding the seal. d) Additional image to indicate that once the plastic cap is removed, the aluminium seal should remain in place until the user removes it.

Figure 2.2a shows the position of the aluminium seal inside the lid as it is being placed on top of the bottle opening. The seal has a thin layer of glue in the surface which will contact with the bottle opening. The lid is then closed (as shown in Figure 2.2b) and the seal is fixed and the glue should be evenly spread across the sealing surface. Figure 2.2c presents the final stage of the production line where the bottle is expected to be sealed and the lid is closed. When the lid is removed by the user, the seal

must be correctly placed over the bottle opening as shown in Figure 2.2d and remain there until the user removes it, to prevent leaks of the substance.

This process presents a serious problem that is fact that the plastic cap can appear in its correct place even if the seal underneath in broken or incorrectly placed. Once the bottle is closed, the seal is unreachable for direct inspection because the lid is made of opaque plastic (Figure 2.2c). Because once the bottle is capped the seal is invisible to a human observer, it is difficult to directly inspect the bottle and assess the quality of the seal. An indirect method of inspection must be used to assure the sealing was done correctly and, since the sealing process uses hot glue, an infrared camera can be used to obtain images of the seal right after the sealing process is finalized.

2.2 Methods

In order to achieve the goal, the system needs:

- An infrared camera to obtain images of the seals right after the sealing process is finished.
- A process that prepares the image with the goal of keeping only the highlighted seal on it.
- A machine learning model capable of classifying the processed images received.
- A database where the image path will be stored alongside details such as the date and time of insertion and the prediction result.
- Mistakes done by the model (incorrect classifications) should be identified and corrected - the incorrectly classified images should be retrained to improve the model performance.

The system will behave as explained below:

- As bottles are coming out on the production line, thermal pictures of the caps are taken, using an infrared camera mounted above the conveyor belt.
- When sealing process is finished, the bottle is led by the conveyor belt to the visual range of the infrared camera where the camera is continuously recording.
- The frames are sent to the computer where they are processed in order to keep only the seal. The hot locations in the frame (created by the hot glue) should be all identified with the color white and everything else with the color black.
- A single frame portraying each bottle is stored. This is achieved by detecting the white seal in a specific position of the frame. The remaining frames of the bottle should be ignored because the seal is the same.
- The number of white pixels in the frame creates a unique pattern that can be verified through the cumulative sum of the pixels along each row and the cumulative sum of the pixels along each column - this unique pattern will be called image signature.

- The model will classify the seal in the frame according to the image signature received as input.
- When a defective seal is identified by the classifier, a signal is sent to a mechanical actuator that removes the bottle from the conveyor belt.
- The result of each seal classification is stored in a database, regardless of being defective or not. All pictures analyzed are also stored in the hard disk drive. This allows for later inspection (using a web application) in order to identify incorrect classifications.
- Incorrectly classified images - either correctly sealed bottles classified as unsealed (False Positives) or defectively sealed bottles classified as sealed (False Negatives) - should be used to retraining the machine learning model, in order to improve the classifier performance over time.

The database contains two tables: one for storing information about the users who have access to the web application such as username, email and password and another for storing information related to the classified images such as the image name, classification result and date and time of classification.

To conclude, the methodology proposed allows for the online and continuous assessment of bottle seals, i.e., it is able to inspect and classify the quality of the seals as the bottles are coming out in the conveyor, checking the caps at exit of the thermal camera.

2.3 System Requirements

The following high-level requirements were identified for the system:

1. The system must occupy as little physical space as possible because there is limited space available on the shop floor;
2. All classification decisions must be logged to enable manual inspection for validation and refinement purposes via a control interface;
3. The control interface must be remotely accessible, both inside and outside the plant local network via web browser.
4. The architecture should be scalable in order to monitor one or more production lines.

Minimizing the operator physical presence at the production plant, which is very much in line with the trend experienced in the new Industry 4.0, and the possibility of checking the production line at any hour of the day or days of the week are the two advantages of this interface.

In order to support the methodology proposed the system also needs the following requirements:

- Interact with an infrared camera and process the video output to extract images of individual bottles;
- Process pictures of the bottles to remove noise, binarize and create image signatures with the sum of the rows and columns;
- Store pictures in a database and retrieve them for inspection and retraining of the classifier;
- Control interaction with the classifier to i) send pictures for classification; ii) obtain classification results, and iii) retrain the classifier;
- Interact with the user to control the system, to: i) query the database; ii) inspect the classifier predictions; and iii) retrain the classifier;
- Control a mechanical actuator and signal it to remove a bottle with a defective seal.

To enable application of the system to different industrial scenarios, the implementation should be modular and independent from specific hardware vendors and the system should be able to run in low cost hardware such as low end industrial computers. Thus, interaction with the infrared camera and with an actuator should be implemented in modules independent from the database and classifiers.

2.4 Use Cases

The diagram in Figure 2.3 presents the system use cases. The functionalities shown in the diagram are accessed via a web application. The administrator and the standard user are the only two actors that interact with the system. There is only one administrator account but there could be multiple standard user accounts – one for each operator performing the task. Both actors must log into ("Log In") the web application to access their functionalities.

The web application controls the quality control system which requires restricted access. The number of users should be minimum to prevent sensitive information from being leaked. The administrator is responsible for managing the users of the web application. When an operator is assigned to perform the task, the administrator creates a new account ("Create User"). When the operator stops performing this activity, the administrator deletes or freezes the account ("Remove User"). This operation is done using the account information such as the ID or username which can be accessed by the administrator ("List Users"). The administrator also has access to the system logs that contain information about some errors that can occur ("Read System Logs").

The standard users can recover their password and username if needed using their enterprise email ("Recover Credentials"). They are also able to change the password ("Change Password"). The main functionality of a user is the selection of incorrectly classified images. First they must select the classification result they want to analyze ("Select Classification Result"). The two options are: classified as sealed or classified as unsealed. The images can be load using a FIFO methodology ("Load Images by

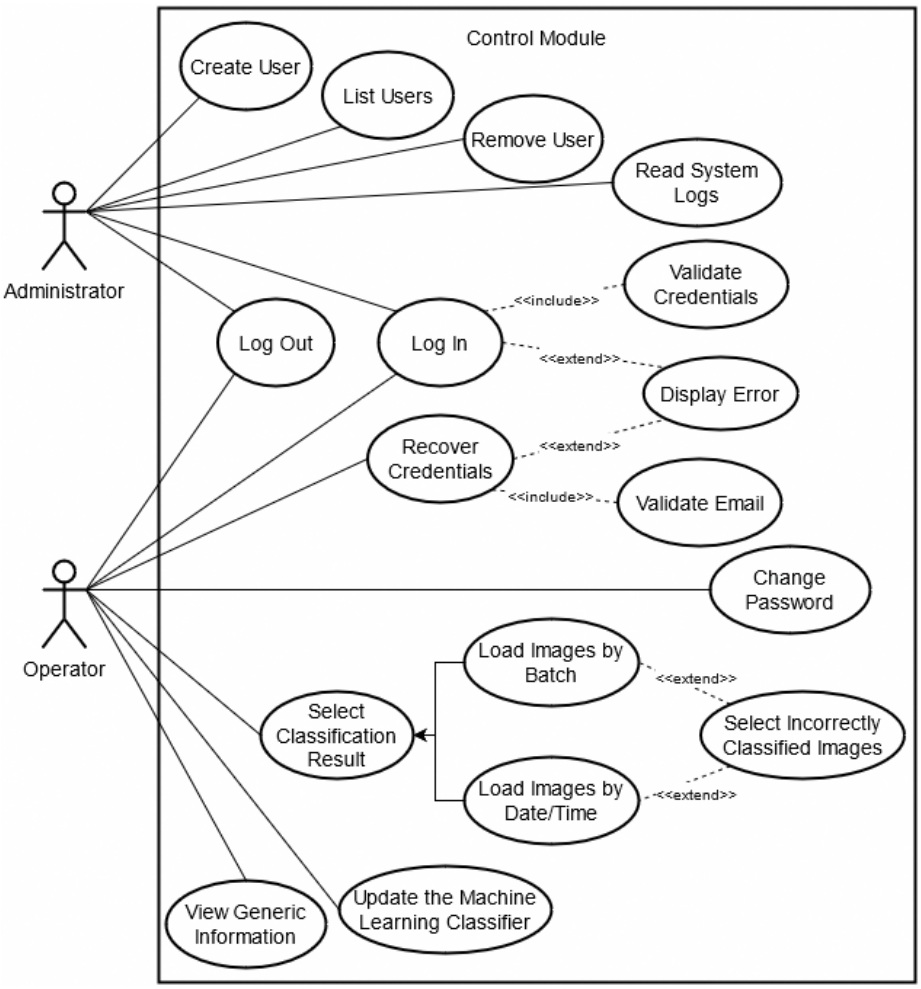


Figure 2.3: Diagram of the user interface functionalities.

Batch") or by specifying the day and hour ("Load Images by Date/Time"). The users then analyze the images displayed and select incorrect classifications ("Select Incorrectly Classified Images") which can then be used to correct and improve the machine learning model ("Update the Machine Learning Classifier") The users can view system information such as the total number of images in the database and when was the last image taken ("View Generic Information").

To conclude, both actor can log out ("Log Out") the web application once their job is completed. These are the functionalities the system developed in this project must perform.

2.5 Functional Requirements

According to the use cases presented in the previous section, a list of functional requirements constructed. The list of functional requirements is divided in two parts, one for each type of user.

The functional requirements for the administrator are presented below:

1. The administrator must be able to log into the web application using a username and a password. Only after login the following requirements can be accessed.

2. The administrator must be able to create accounts for persons who will use the web application. To create an account is necessary a unique username and the enterprise email. A temporary password is created by the administrator and then the account owner is asked to change the password.

3. The administrator must be able to view the details of the accounts (except the password) of every user of the application.

4. The administrator must be able to remove accounts of users who are no longer responsible for this activity.

5. The administrator must be able to read the system logs in order to troubleshoot problems that could happen such as images not being properly received.

6. The administrator must be able to log out the web application.

The functional requirements for the standard user are presented below:

1. The standard user must be able to log into the web application using a username and a password. Only after login the following requirements can be accessed.

2. The standard user must be able to recover the login credentials using the email linked to the account.

3. The standard user must be able to change the password.

4. The standard user must be able to choose the classification result of the images loaded into the web interface.

5. The standard user must be able to select the methodology used to load the images – either using the FIFO methodology or by selecting the day and hour when the images were classified.
6. The standard user must be able to select the classifier mistakes – incorrectly classified images.
7. The standard user must be able to improve the classifier by retraining the mistakes selected.
8. The standard user must be able to read generic information about the system in order to identify problems that could happen such as images not being properly received.
9. The standard user must be able to log out the web application.

The two lists previously presented will be a guide through the system development.

Chapter 3

Concepts and Related Work

In this chapter, a review of the state of the art is done in order to present concepts and methods related to this project. It includes a list of multiple infrared uses, machine learning models, monitoring systems and architectures, and sealing quality control systems.

3.1 Spectroscopy

Infrared spectroscopy and near-infrared spectroscopy are techniques that analyze the infrared radiation emitted by warm objects. Based on the energy emitted, the materials composing the object and its properties can be identified. This is possible because materials radiate energy differently accordingly to their specific physical properties.

Asaduzzaman *et al.* [1] used infrared technology to assess the quality of milk samples. The goal was to identify milk that was falsely labeled as having been produced in ecologically clean mountain areas. Raw milk samples from three different farms were used. These farms are located in the same geographical region but with different properties such as altitude and farming methods which causes differences in the characteristics of the samples. After the samples were warmed, their infrared emission wavelengths were analyzed and it was possible to identify the component characteristics and their concentration in the milk. A machine learning algorithm based on k-nearest neighbor was then used to classify the samples based on the characteristics identified.

M. Boiret & F. Chauchard [2] used near-infrared spectroscopy to control both the distribution and the content of active pharmaceutical ingredients within final drug products [2]. The goal was to create a real-time quality control system. Near-infrared spectroscopy is used because it supports high speed measures, it is free from pollution, it does not require reagents nor sample preparation, it is non-destructive and is capable of providing information about components existing in Active Pharmaceutical Ingredients (API) tablets[2]. The study used two batches manufactured in two different chemical manufacturing sites [2]. The measurements were done at high-speed, using a conveyor belt system which allowed the analysis of multiple tablets. The data obtained was an-

alyzed according to their quality (API distribution per tablet) and quantity (quantity of the API content within the final drug products). The data collected was displayed in graphics where the two batches can be distinguished proving the system effective.

This project does not require a detailed analysis of the glue composition which excludes the use of infrared spectroscopy. The analysis of the glue composition with this technology could be interesting if the goal was to study the seal condition in long term. For example, test different glue types in the sealing process, verify if the seal will have gaps in a long term and correlate that to the glue composition.

3.2 Condition Monitoring

In condition monitoring, the properties of an object such as a motor, a tool or even a building are analyzed regularly in order to detect changes as soon as they appear and prevent major problems.

Pontes *et al.* [3] used infrared technology to detect hot locations on a motor in order to identify situations of probable overheating. The goal was to increase the lifespan of the equipment by monitoring changes in the materials' properties, aiming to detect and fix problems as soon as possible. The images are split into the RGB plans and a combination of the red and green plans was done to get a grayscale image. The background was removed using a threshold which mapped all pixels below a certain value to 0 and the remaining to 255. Image signatures were created from the sum of the values in each line and column and the two resulting vectors were fed to the neural network. Through the comparison of signatures it was possible to evaluate the state of the motor – if the image presented higher values on the signature it meant the motor was overheating. The process requires low computing power, thus being possible to implement in a small industrial computer or a Raspberry Pi.

Elgargni *et al.* [4] proposed a system that assesses the condition of cutting tools before their use. The authors retrieved multiple visible wavelength and infrared images of the cutting tools and then, after using image processing functions and algorithms to isolate and extract the features, the images were classified using principal component analysis and an artificial neural network. They concluded that the infrared data, combined with suitable image processing and artificial intelligence, could be a very efficient method for on-line condition monitoring of cutting tools.

Haider *et al.* [5] presents another use of infrared to monitor the condition of electric components. The infrared technology is used to obtain images of electrical systems to identify problems in order to reduce maintenance costs, improve productivity and increase machine availability. The method is based on the fact that most components tend to have an increase in temperature while malfunctioning. Instead of using gray scale images, the infrared images are converted to HSV (Hue, Saturation, Value) and processed with various threshold methods: *Roberts Edge Thresholding*, *Prewitt Edge Thresholding* and *Otsu Thresholding*. Linear Discriminant Analysis (LDA) is then used for the classification and feature dimension reduction. In their experiments they used infrared images of switches, circuits, fuses, and other electrical equipment of their own institution building. A total of 40 sample images were captured, from which 60% were used for training and the remainder for testing. Approximately 55% of the images

corresponded to equipment in abnormal condition and 45% in healthy condition. From the tests, the authors concluded that 0.70 was the suitable threshold and then were able to get a model with 100% *accuracy*.

Al-Habaibeh & Parkin [10] use a low-cost infrared sensor to obtain images of several manufacturing processes such as drilling, grinding, welding and soldering. The images contain the distribution of heat in the working machine used in the process and are then evaluated using a novel detection algorithm which is a self-learning approach that distinguishes between normal and faulty conditions of the manufacturing processes. A faulty process will overheat the machine and it can be detected using infrared technology. The systems aims to distinguish between a machine working in normal conditions and overheating machines. This approach was then tested in the four processes mentioned above and the authors concluded that it is most successful with the grinding process and the welding process was the least successful.

This subsection presents interesting features that could be applied to this project. The image signatures [3] present an interesting method of reducing the number of inputs in the machine learning algorithm making it faster and simpler to train. Artificial neural networks are used in two of the three described works [4][3]. The HSV color model [5] simplifies the task of highlighting objects in the image.

3.3 Object Classification

Through the use of artificial intelligence the process of identifying objects in an image can be automatized and in this section some examples are presented.

Zheng *et al.* [6] propose the implementation of infrared technology to detect and identify battle targets such as cars, trucks and tanks. Despite methods already existing to process images, some army infrared sensors can only obtain a low-resolution image that, once used as input, causes small targets to disappear from the image. The solution to this problem was to design a new model based on the You Only Look Once (YOLO) Convolution Neural Network (CNN), which is a real-time object detection deep learning model [7]. YOLO [1], in its third version YOLOv3 uses a 53 layer neural network. It is designed for visible light image detection and recognition and requires a higher resolution, so the authors designed a model based on this structure. Three models were built: the first one was trained using only visible light images, the second one using infrared images and the last one was first trained with visible images and then fine-tuned with infrared images. The third model not only presented the best results from the group but also presented better results when comparing with the YOLOv3.

In Zheng *et al.* [6] the system goal is to detect objects in images of outdoor landscapes, streets and other similar environments. In these images there are a large quantity of objects to be analyzed because the camera captures a large range of action. In this project, the camera is specifically located above the conveyor belt after the sealing stage which limits the camera range and causes the bottle to occupy a large portion of the screen making it easily visible.

[1] YOLO official website (last access on 2021-05-22): `https://pjreddie.com/darknet/yolo/`

3.4 Non-Destructive Testing

In non-destructive testing, a sample of a product is taken from the production line to be the target of quality tests which will not destroy the sample meaning it can still be sold if its quality was assured.

Dua & Mulaveesala [8] used Infrared Thermal Wave Imaging (ITWI) to evaluate the condition of reinforced concrete. Reinforced concrete is commonly used in the building industry due to its low cost, high strength, robustness and sustainability, along with the ready availability of raw materials but has some drawbacks such as poor tensile strength and ductility [8]. Those drawbacks can lead to the formation of cracks where over time – the ingress of chloride ions and carbon dioxide in the steel surface originates corrosion and leads to the decrease of the cross-section in the material. When that happens, the safety and stability of the structure can be compromised. ITWI is used to measure the heat emitted by the surface of an object. In this work, the object is the surface of the reinforced concrete. It is analyzed using ITWI and the images obtained are then processed in order to generate an image where the faults in the material are clearly visible.

The technology used in this subsection is not essential to this project but, as explained in Section 3.1 it can be used to evaluate the long time performance of the seal but this is not part of the scope of the current project.

3.5 Sealing Checking

Sealing checking is the process of assuring the quality of a seal used to confine a product.

Al-Habaibeh *et al.* [9] developed a mechatronic approach to assure food is correctly sealed inside containers. The sealing is done using a laser and through the use of infrared thermography combined with image processing its authors try to verify the quality of the seal. The infrared camera obtains images of the seal during and after the sealing process. The heat pattern is clearly visible in the resulting images. The thermographical properties are analyzed and if any abnormal area is detected, then the seal is compared to previous faults/problems cases, so that the laser properties can be adjusted. The system is capable not only of detecting the localized heat pattern but also the contamination between the film and the container, as well as the cooling behavior of the materials.

D'huys *et al.* [11] present a pulsed-type active thermography experiment to detect solid contaminants in between seals of heat-sealed flexible food packages. In the study, the authors prepared 30 seal samples contaminated with ground coffee particles that were recorded shortly after sealing. The system obtains high resolution digital images using active IR thermography. These images are then processed using six methods:

1. A method based on a single frame;

2. A method based on a first order polynomial fit of the cooling profiles in the logarithmic domain;

3. A method based on thermal signal reconstruction;
4. A method based on pulsed phase thermography;
5. A method based on principal component thermography;
6. A method based on a 3D matched filter.

The method based on a fit of the cooling profiles presented the lower detection limit meaning that particles with an equivalent diameter of 0.60 mm were detected with a probability of 95%. Also, the detection capabilities of this method do not depend strongly on the time after sealing at which recording of the thermal images was started, making it a robust and generally applicable method.

Shuangyang [12] present a high-speed and high accuracy quality inspection of food packing seals. At the end of a correct sealing process, continuous sealing lines can be seen on the package. First, an image of the seal is taken and then the second step is to threshold the seal zone, making it a black and white image where the continuous seal lines should be highlighted. The authors then use a template to match the image and by comparing the number of pixels matched the system classifies the seal as faulty or acceptable. The system presents an average inspection *accuracy* of 93.6%.

This subsection presents a list of works which deal with problems similar to the study case of this project. Converting the retrieved images to black and white is an important aspect because it simplifies the task of highlighting the seal by removing a large portion of the background noise.

3.6 Machine Learning

Multi-layer perceptron (MLP) neural networks have been used before for image classification. Del Frate *et al.* [13] assess and optimize neural networks' performance for pixel-based classification of high resolution satellite images. The images were obtained from two sources: Quickbird data, characterized by very high spatial resolution, and the Landsat data characterized by high spatial resolution. The study aims to distinguish among areas made of artificial coverage (sealed surfaces), including asphalt or buildings, and open spaces such as bare soil or vegetation. The feature extraction and information discovery on urban areas can be used to monitor changes and urban growth over time, which is helpful to improve the environment and safety measures. The neural network obtained presented a satisfactory performance, with the overall *accuracy* of 87% for Quickbird and 82% for Landsat subareas.

Del Frate *et al.* [14] describe the application of m-arcsinh, a modified version of the inverse hyperbolic sine function in machine learning algorithms distributed in the Python library scikit-learn. The algorithms used were Support Vector Machines (SVM) and MLP Artificial Neural Networks. The study used six datasets from the scikit-learn and nine datasets from The University of California at Irvine (UCI) ML repository. The aim of the study was to develop a novel computationally efficient and reliable kernel and activation function and evaluate it against the standard functions available in the Python library scikit-learn. The *accuracy* and *F1 score* were the metrics used to assess

the classifiers' performance. The results showed that using the m-arcsinh kernel and activation function the MLP presented the best classification performance on 10 out of 15 datasets evaluated. The MLP performed better than the SVM, except for two out of 15 datasets. The models' reliability was higher and better than some standard functions.

Subhadip Basu *et al.* [15] present a Multi Layer Perceptron based classifier for recognition of handwritten Bangla alphabet. The Bangla alphabet is the second most popular written and spoken language in the Indian subcontinent. The MLP based classifier needs to distinguish between 76 elements. The variations in writing styles of individuals make recognition of handwritten characters difficult and demand the development of a classifier that has generalization abilities. 10,000 alphabetic characters were collected from 200 subjects with different sex and age from which 80% were used as training set and 20% as testing set. The images were scaled to 64×64 pixels and converted to binary through the use of a threshold. Only one hidden layer, with 60 neurons, was used, in order to keep a low computational power requirement. The classifier showed performances of 86.46% in the training set and 75.05% in the test set.

3.7 Monitoring System Architectures

Dimitris Mourtzis *et al.* [16] propose an approach for condition-based preventive maintenance of machine tools and cutting tools, based on real-time shop floor monitoring. The data used in the study come from two data sources:

- The multi-sensory system – hardware used to monitor the tools.
- The machine tool operator input – information collected from the operators through reports made on mobile devices.

The data are stored in a database in a cloud service and then the variables under analysis are calculated. The result can be accessed by the maintenance department through a web application. The communication between the operator and the maintenance department is facilitated through the cloud. The use of mobile technology leads to a reduction of the time it usually takes to solve maintenance problems on the factory floor. The authors tested the approach on five milling machines that over a month raised 20 reported problems, which were solved in 4 man hours instead of the predicted 10 man hours, according to the maintenance department.

Luc Bongaerts *et al.* [17] examine the role of hierarchy within the domain of factory floor control, in manufacturing plants. The paper analyses three distinct structures: hierarchical, heterarchical and holonic control systems. Hierarchical control involves a command—response structure between high level and low level entities [17]. Heterarchical control is achieved by allowing a high level of autonomy and decision making to be available to low level entities, independent from the overall plant operations [17]. Both of these systems have disadvantages, such as the hierarchical systems tend to have problems with reactivity to disturbances. The heterarchical control turned out to have problems with global optimisation and predictability. The authors then analyze the holonic control systems, which try to merge the good properties of the previous

mentioned systems. In the holonic control system, a holon has a degree of independence and handles contingencies without asking higher authorities for instructions, but can be also subjected to control from higher authorities [17].

Jun Sun & Wysk [18] propose a structure and architecture for automatic simulation model generation. They propose a framework that includes model generation and automated inputs applied to the manufacturing industry. The Shop Floor Control System (SFCS) receives production orders from the production database and then perform tasks such as selecting specific process routings, allocating resources, scheduling parts, etc. The simulation model was developed using ARENA Software ©. Through an Open Database Connectivity (ODBC), the simulation obtains part orders and part process plans using an SQL connection to an interactive database. These are then sent to the high-level task executor (also known as BigE) that performs the shop-level execution functions and keeps track of the status of each material processor, material handler, and the storage system resource. Once the task is completed, the BigE sends a message to the simulation and the task status in the database is updated. The database keeps track of part orders and how many parts in each order have been completed. They apply it to six manufacturing systems with material processors, material handlers and Automated Storage and Retrieval System (AS/RS).

P. Vithu & J.A. Moses [19] review methods of evaluating the quality of pre-processed food grains using a machine vision system that, using image processing and image analysis techniques, will perform a physical inspection that gives the most information about the condition of the grain. It reviews non-destructive, non-contact and non-invasive methods to evaluate the existence of foreign matters, insect infestations, microbial infections, or grain grain discolouration. Those systems identify and classify the grain according to type and variety. The paper also clarifies the limitations of the use of machine vision, such as:

- It is hard to understand the grain composition;
- It requires high quality images, so the grains and their details can be easily identified;
- The authors suggest the use of combined learning techniques and a variety of spectral ranges which will detect information that the visual range is unable reach.

Jiangfeng Cheng *et al.* [20] propose the architecture of 5G-based Industrial Internet of Things (IIoT) to develop cyber-physical manufacturing systems (CPMS) that perform a variety of tasks such as shop-floor layout optimization, smart planning in production processes, production factor configuration, inventory management, scheduling, etc. Multiple manufacturing scenarios are described alongside the implementation methods and manufacturing technologies under the circumstance of three typical application modes of 5G: enhance mobile broadband (eMBB), massive machine type communication (mMTC) and ultra-reliable and low latency communication (URLLC). IIoT data transmission architecture is similar to the traditional IoT and contains the three standard layers: sensing layer, network layer and application layer. The architecture includes both wired and wireless communication according to the scenario. If

high reachability is a requirement then wired communication is the method to follow and if an easy mobile access is a requirement then it is suggested the use of wireless communication.

Saez-Mas, A. *et al.* [21] detail a 4-layers architecture and adapt it to two specific case studies. The first case is moving car bodies from the paint plant to the assembly line and the second is analysing a layout of a section used to assemble the engine and transmission set. The 4-layers previously mentioned are:

1. Network Layer – includes the machines, buffers, paths and products to be transformed.

2. Logic Layer – includes decision-making processes that may activate other decision procedures, or even elements in the Network Layer.

3. Database Layer – feeds simulations with the data needed to perform activities and to make decisions. It also stores the results.

4. Visual Layer – eases communication with users and stakeholders.

The architecture promotes separation of the physical system and the logical system, thus allowing a better understanding of the problem, and the reuse of layers in future models that have similarities. It also facilitates exporting the information to be used in statistic tools, because of the Logic and Database layer separation.

This subsection presents a list of architectures used to solve problems in the shop floor. A database is essential to store the data collected. The use of a web application is relevant to this project. It enables the access to work from outside shop floor environment.

3.8 Quality Control Systems

The following enterprises develop proprietary quality control systems and are already available in the market. The problem solved in this project has similarities to the problems solved by these enterprises.

The European company Qipack [2] verifies the quality control in packing processes that are based on detailed real-time analysis of each heat sealed product. This system performs a continuous, real-time and non-destructive monitoring, using a high definition camera to record the seal immediately after its production. The system analyzes the pixels received from the camera and search for abnormalities in order to remove products with defect. It is used by companies such as Capri-Son, Danone, Heineken, Heinz, among others.

TheSealCheck™ [3] is a USA based company and develops solutions for thermal-imaging based automated heat seal inspection. It presents numerous applications for its software from caps, bags, packages, plastic trays, etc.

[2] Qipack Official Website (last access on 2021-04-10): `https://www.qipack.com`

[3] TheSealCheck Official Website (last access on 2021-04-10): `https://www.thesealcheck.com/`

Also based in the USA, the PECO-InspX [4] develops solutions for food inspection using mainly X-ray systems. The solution behaviour is similar but it is oriented to detect different defects. The X-ray are not suited to detect lower density objects. The company solutions focus on detecting metal, glass, stone, rubber, and many high-density plastics.

[4]PECO-InspX Official Website (last access on 2021-04-10): `https://www.peco-inspx.com/`

Chapter 4

System Architecture

In this chapter the system architecture is described. In the first section, the monolithic architecture is compared to the distributed in order to decide which to use and in the second section all modules of the architecture are detailed.

4.1 Architecture Types

The company has limited space available in the shop floor. Creating a distributed or decentralized system with multiple hardware components may present challenges during its implementation due to the lack of free space. A distributed system is harder to implement, maintain and troubleshoot. It is easier to lose packets in the communication between elements of distributed systems especially if the communication is established via Wi-Fi. When it comes to security, a distributed systems generally presents more targets to attack. The alternative is creating a monolithic or centralized system.

A monolithic system requires less physical space, it is easier to implement and simplifies the maintenance and troubleshooting tasks. Since the communications are all self-contained in a single computer, the data transmitted is less prone to be corrupted due to packet loss. These type of systems present a disadvantage which is if the computer breaks, the entire system stops and this can also be linked to the security of the system. In this case, there is only a single target but, if the target is compromised, the attacker disrupts the entire system.

The monolithic architecture was selected because it satisfies the company space requirement. Since a single computer is required, it is easier to create security rules to mitigate vulnerabilities such as restricted access to the system, video surveillance of the equipment, configuring the firewall with only the strictly necessary rules, keeping the system updated and others. Since all components are contained within a single computer, this machine processing power must be higher than required for each individual node if the architecture was distributed. The machine will not be located in a controlled environment but in the shop floor where accidents can happen and causing damages to the equipment. Robust hardware must be used to assure the system will not easily break.

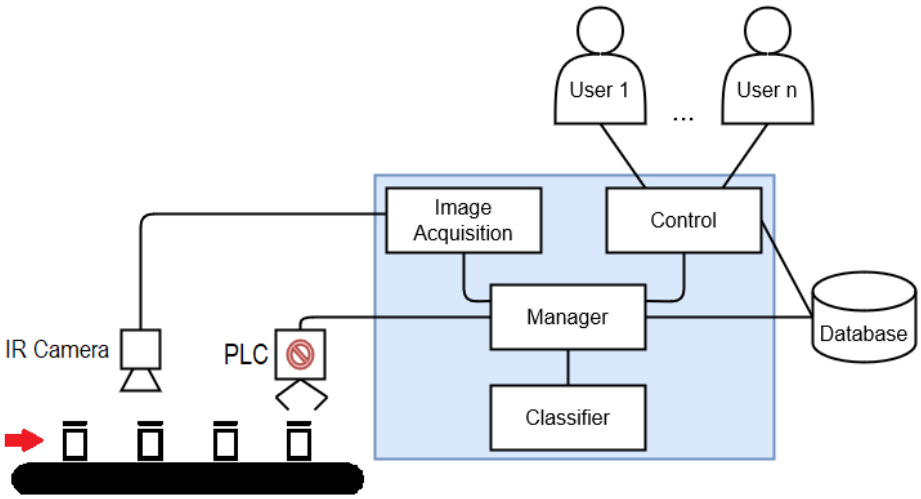


Figure 4.1: System architecture.

4.2 Detailed Architecture

Figure 4.1 shows a graphical representation of the modular architecture proposed for the system. It presents eight elements: i) infrared camera, ii) Image Acquisition module, iii) Manager module, iv) programmable logic controller, v) Classifier module, vi) Control module, vii) database and viii) the end users. In the next subsections each element will be detailed.

4.2.1 Infrared Camera

As the bottles are being filled and sealed, they advance one at time through a conveyor-belt, passing through the infrared camera visual range immediately after the caps are closed. Thus, when the images are captured the glue that fixes the seals is still hot and the ring of the bottle where the sealant is bound to the bottle opening is clearly visible in the thermal image.

The infrared camera captures images in video mode, and the video is sent to the Image Acquisition module.

4.2.2 Image Acquisition Module

In the Image Acquisition module, each frame from the video received is cleaned and converted to black and white in order to create a image where only the seal is visible by removing all other existing artifacts. A single frame from each bottle is selected - a frame where the seal is clearly visible in the center (region of interest) - and sent to the Manager module.

4.2.3 Manager Module

This module manages the connections between the elements of this architecture.

When the Manager module receives an Image from the Image Acquisition module, it sends the same image to the Classifier module and awaits for the classification result. If the result identifies a defectively sealed bottle, the Manager emits a order to the PLC to remove the bottle from the conveyor belt. Independently of the prediction result, it stores the image in the hard disk drive and adds a entry to the database containing information about the classified image.

Incorrectly classified images can be selected in the Control module web interface to be partial fitted. The Control module sends the image to the Manager which sends the same image to the Classifier that will partial fit the image. The goal of this action is to improve the machine learning model performance.

4.2.4 Programmable Logic Controller

The programmable logic controller (PLC) receives a order from the Manager to remove bottles classified with a defective seal from the production line. The conveyor belt moves at a constant speed allowing the estimation of the time the bottle takes to move from the infrared camera to the PLC.

Once the unsealed bottle arrives at the target location – PLC actuation range – the Control module send the signal to the PLC which will remove it.

4.2.5 Classifier Module

The Classifier module contains the machine learning model. Once it receives an image from the Manager module, either classifies it or partial fits the model to improve its performance. It sends the resulting information back to the Manager module.

4.2.6 Control Module

The Control module contains the web application used by users to interact with the system. The user can access the interface from anywhere as long as they have access to an Internet connection and a web browser that supports HTML and the HTTP protocol. The application has the standard basic functionalities such as log in, log out, add user, remove user and list users. In addition it has the project specific functionalities such as loading a batch images with a classification result defined by the user, selecting incorrectly predicted images and view generic system information such as if the system is active and when was the last image classified.

The standard operator responsible for loading images and selecting incorrectly predicted images should not be able to add or remove users, for example. For security reasons, an administrator account was created to perform tasks related to user management such as:

1. If a new worker is assigned to work with the system, the administrator must create a new account with a temporary password. The operator will then log in once the account is activated and define his definitive password.

2. If a operator is reassigned or leaves the company his account must be removed.

The administrator should also be the only only with access to the system logs.

This creates two types of users that can access the interface: the administrator and the standard operator.

4.2.7 Database

The database is only accessed by the Manager or the Control modules. The Manager modules accesses the database to add a new entry containing information about the classified image. The Control module accesses the database to load images in the interface. The data stored in the database is detailed in Section 4.3.

4.2.8 Security Discussion

The use of web-based interface brings the risk of excessive exposure to unauthorized users, in particular from the outside of the plant network environment, creating security concerns. The use of encryption on data transfers, the identification of clients through Secure Sockets Layer/Transport Layer Security (SSL/TLS) protocol used in parallel with a secure database authentication system, and Virtual Private Network (VPN) technology [16] are recommended countermeasures against threats and can be used with our proposed system. Although it is a subject outside the scope of this work, these security measures are proposed to regulate the access from outside of the company intranet to the system and its control interface, reducing the risk of unauthorized access to data and functionality.

4.3 Data Stored

The proposed system requires the persistence of two types of information. The first is the information related to the images classified and the second is the information related to the users of the web application.

For each images classified, the image ID, the file name, the classification result (whether the bottle is sealed or not), the date and time of classification and the images state must be stored. All images will be stored in a static location inside a folder in the machine where the system will be executed. By using the path to that folder and the name defined in the database entry, every single image can be accessed.

For each account created the account ID, the username, the password and the email (with the exception of the administrator which has no email) must be stored.

To conclude this chapter, the proposed architecture enables a remote access which discards the need to be inside the shop floor to perform the tasks (results verification and machine learning model improving). By reducing the need of a operator inside the in the shop floor, it increases safety and reduces the probability of work accidents, cross contamination or similar problems.

By analyzing the classification results, users can validate the operations and, in case the machine learning model made a wrong prediction, they can select the image to be

partial fitted. This discards the need of having to retrain using the entire dataset and allows the model to keep evolving and its performance will increase overtime.

Chapter 5

Data Acquisition and Preparation

This chapter cover every aspect related to the data used in this project. Its starts by analyzing the videos used to simulate the production line activity. From each video, a single frame from each recorded bottle is saved and used to generate the image signature that will be used as input in the machine learning algorithm.

5.1 Thermal Camera

The camera used to take images of the seals is an Optris Pi 400i and it is presented on Figure 5.1. According to the documentation, the optical resolution of the camera is 382×288, meanwhile the resolution of the videos obtained is 382×289 pixels. There is one additional row of pixels whose origin in unknown but most likely due to some type of data corruption. The camera supports a frequencies of 80 Hz and 27 Hz. The conveyor-belt does not move at a very fast speed which makes it possible to record all bottles using the 27 Hz frequency which will capture 27 frames per second. It can weight from 237 g to 251 g according to the lens used and is USB powered. More details are available in the official Optris Pi 400i web page [1].

Figure 5.2 shows the baseline image recorded by the camera and presents two images. Figure 5.2a is unchanged and presents the baseline image retrieved from the camera. It shows the conveyor belt and its surroundings according to the chosen infrared color scheme. Due to the lack of shadows in the infrared image, some detail such as depth is lost. The lack of depth makes it harder to visualize the objects presented in the image an understand the boundaries. For this reason, figure 5.2b was added. In this figure, details such as the conveyor belt boundaries (the black lines) and the direction it will move (arrow pointing upwards) were manually added.

In this infrared color scheme, hot locations are identified with a green color and in

[1] Optris Pi 400 official documentation (last access at 2021-04-10): `https://www.optris.com/thermal-imager-pi400i-pi450i`

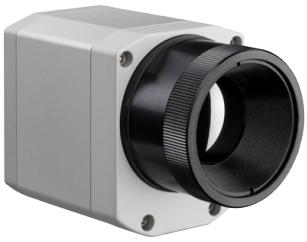
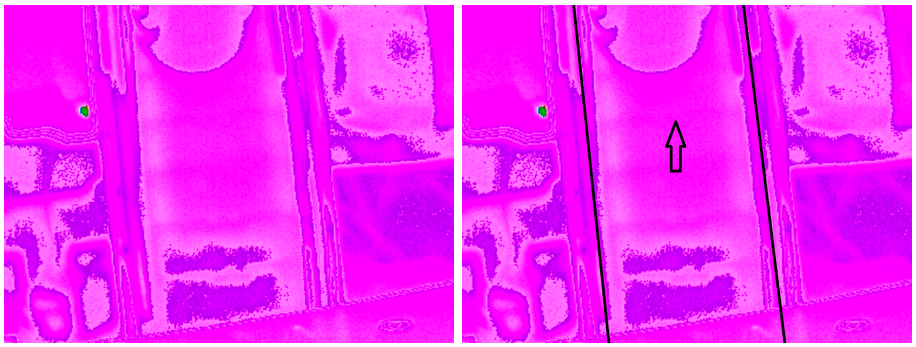


Figure 5.1: Thermal camera used on the process.



(a) Original (b) Edited

Figure 5.2: Baseline image of the conveyor belt presenting its boundaries and the direction it moves.

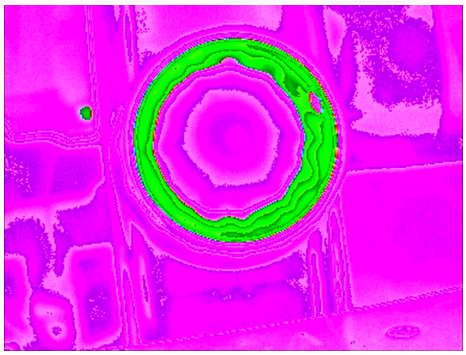
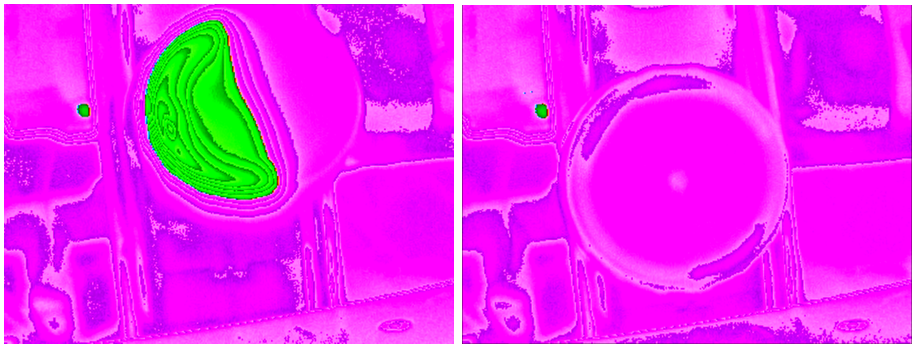


Figure 5.3: Example of the sealing process done right - fully sealed bottle.



(a) Bottle half-sealed (b) Bottle not sealed

Figure 5.4: Two images resulting from a poor sealing process.

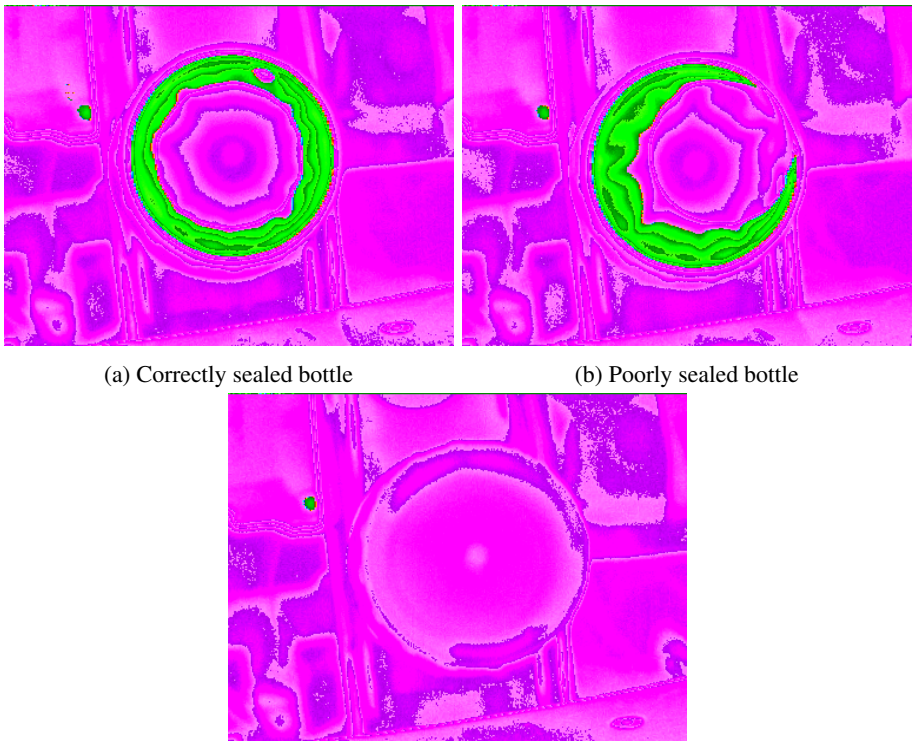
the baseline image presented in figure 5.2a a green dot can be seen. This is part of the conveyor belt motor that is situated in the infrared camera visual range.

Figure 5.3 presents a correctly sealed bottle. If the image of the seal obtained has a full circle similar to what is shown on figure 5.3, the process was done right and the bottle is ready to advance in the production line or be sold.

In figure 5.4 are presented two possible results of the sealing process done wrong. In figure 5.4a only half of the bottle is sealed. The camera still identifies a bottle and in the image obtained it is possible to verify which side is sealed (green color) and which side is not sealed. In figure 5.4b no glue was applied to the bottle. This situation is different from the first one because the bottle has the same color as the background and can only be identified because of the round shape presented in the image.

5.2 Raw Data Analyzed

A small conveyor-belt was used to simulate the bottle movement in the production line. Above the conveyor-belt, the camera presented in previous section was fixed.



(a) Correctly sealed bottle

(b) Poorly sealed bottle

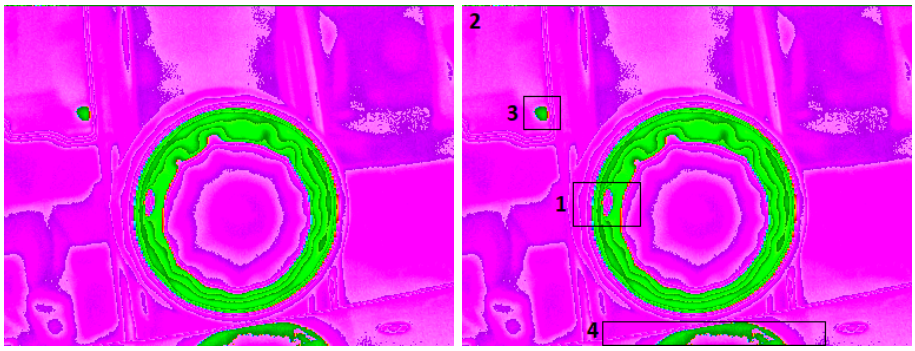
(c) Seal without glue

Figure 5.5: Three types of images of bottles recorded.

A group of bottles were removed from the production line after they were sealed and used to record these videos that were send to the Tula Labs. In these group, all bottles were correctly sealed so a trained employee manually damaged a few sealed bottles and placed them in the conveyor-belt that led the bottles through the infrared camera visual range. This generated the poorly sealed bottles seen in the video that were used simulate the errors that could occur on the shop floor. From this simulation, a total of four videos were obtained that included both sealed and unsealed bottles.

From this simulation, four different videos were recorded attaining a total of approximately eight minutes. During this time fifty-nine bottles were recorded. Figure 5.5 presents the three types of bottles that can be seen in the videos.

As explained in Section 5.1, in this infrared scheme, the hot locations are identified with green tones. The background is mostly at the environment temperature is and is identified with purple and pink tones. Figure 5.5a shows a thermal image of a correctly sealed bottle. The green circle represents the hot area where the seal adhered to the bottle opening. A correctly sealed bottle can be identified by a complete green circle all around the bottle opening. If the image of the seal obtained has a full circle, of an acceptable thickness, similar to what is shown in the Figure 5.5a, then the sealing is defect-free and the bottle can be moved forward for distribution. Figure 5.5b shows



(a) Original (b) Elements identified

Figure 5.6: Problems detected in the videos recorded. 1) Absence of glue in a specific area in the seal due to a pin; 2) Row of corrupted pixels; 3) Hot location caused by the exposed conveyor belt motor: 4) Bottle reflection on a metallic surface attached to the camera.

a thermal image of a unsealed sealed bottle where the glue was not spread properly in order to form the perfect circle. Finally, Figure 5.5c shows a thermal image of an unsealed bottle which contains no glue in the seal, therefore it does not present a hot location defining the seal.

5.3 Detailed Frame Analysis

The frames contain additional artifacts besides the seal which will be analyzed in this subsection. Figure 5.6 shows an image of a seal containing the four artifacts presented in each frame of the videos.

The first artifact is a small pink hole in the green circle. This hole is a colder area and is caused by a plastic pin that comes on all the bottle caps. This artifact is different from the following because it has no significant effect on the resulting image signature, therefore is does not need correction. This artifact only appears when a seal is being recorded.

The second artifact is the top border of the image which contains an entire horizontal line of pixels that, for some unknown reasons, was corrupted. This problem can be clearly seen in Figure 5.9a and appears in every frame recorded.

The third artifact is a hot location of the same color of the seal caused by part of the conveyor-belt motor that was left exposed to the field of view of the thermal camera. This artifact appears in every frame recorded.

Lastly, the fourth artifact is a reflection of the image bottle caused by the thermal sealing chamber stainless steel wall. Part of this reflection appears in the thermal camera visual range and is recorded resulting in what is shown in the fourth artifact of Figure 5.6. This artifact only appears when a seal is beginning to appear on the screen. After the bottle leaves the reflection zone, the background is reflected instead and the

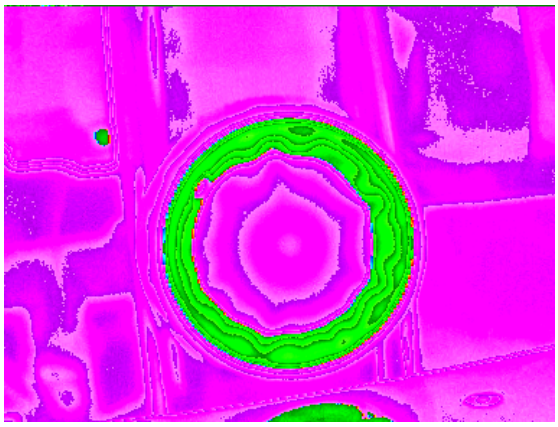


Figure 5.7: Original image that will be used during this section as an example.

noise created by this artifact can be avoided.

These last three artifacts differ from the first because they must be removed or avoid during image preprocessing because they add noise to the machine learning model.

5.4 Image Selection and Processing

The videos obtained are analyzed using image processing techniques in order to get a binary image containing only two colors – black and white. This binary image will be used to get the input for the machine learning model.

5.4.1 Color Scheme Analysis

The first step is to select the best color model to work with because it simplifies the noise removal stage. The color schemes tested are grayscale, RGB and HSV.

OpenCV by default uses the BGR color model which is similar to RGB color model – the difference between BGR and RGB is the order in which the colors are stored in memory. Before analyzing the three planes (blue, green and red), the original image was converted to grayscale. The results were poor and can be seen in the Figure 5.8. In this image, the seal acquires a similar color as the background making it harder to identify it.

Figure 5.7 was split into the three BGR planes and the results can be seen in Figure 5.9a, Figure 5.9b and Figure 5.9c). In all images the seal was highlighted but the blue plane presented the best result. Although there is still noise that must be removed using further processing, the image has less noise when comparing to the other two planes.

Finally, the HSV color model was tested. Even though the seal can be identified in Figure 5.10a, the HSV image does not present the best results when comparing to Figure 5.9a but, when combining this image with the OpenCV function inRange(), it is achieved the best result so far, as shown in Figure 5.10b where the seal is clearly

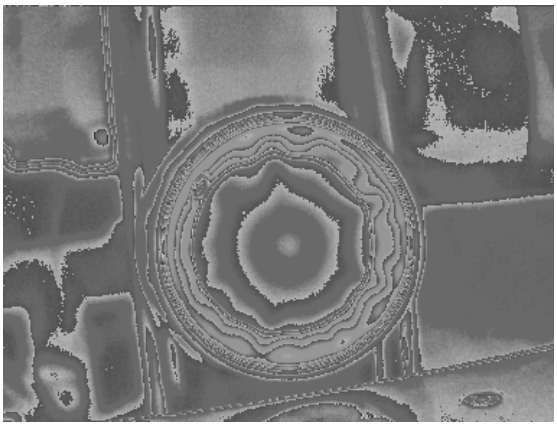
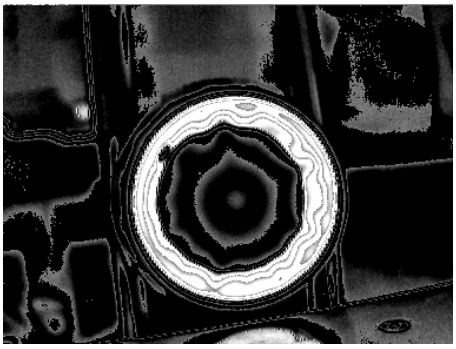
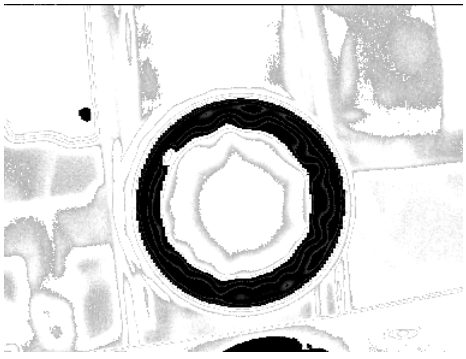


Figure 5.8: Figure 5.7 converted to grayscale.

(a) Figure 5.7 blue plane

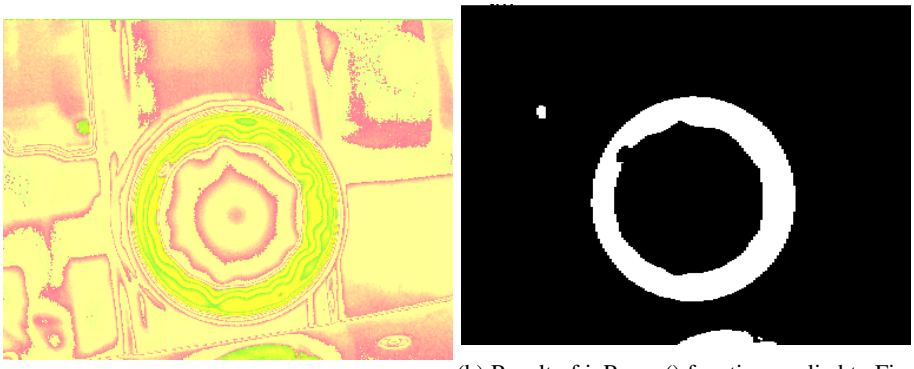


(b) Figure 5.7 green plane



(c) Figure 5.7 red plane

Figure 5.9: Figure 5.7 BGR planes.



(a) Figure 5.7 converted to HSV

(b) Result of inRange() function applied to Figure 5.10a

Figure 5.10: HSV color model of Figure 5.7 and inRange() function result.

Table 5.1: Range of hue, saturation and value used in the *inRange* function.

	Hue (H)	Saturation (S)	Value (V)
Maximum	60	255	255
Minimum	54	234	132

highlighted and the only noise presented in the image is the hot spot caused by the conveyor belt motor.

5.4.2 Function inRange()

The OpenCV function inRange() receives an HSV image and the lower and upper limits to be read. Then, it verifies if the pixels h, s and v values are contained within the interval defined by the upper and lower limits and converts every pixel inside that range to 1 (white) and the remaining to 0 (black). This could be applied to this problem in two different ways:

- Define an interval that contains the pixels in the seal.
- Define an interval that contains everything except the seal and invert the result.

The first option was selected to solve this problem. The next step is to identify the pixel range that the function must select and define lower and upper limits based on observation evidence.

As can be seen in Figure 5.7, the seal is identified by various green tones. In order to find the interval where the seal colors vary, a small portion of the seal was isolated and converted to an HSV image. The values of each pixel in the HSV image were stored in a text file and then loaded into an Excel sheet. In Excel, the maximum and minimum values of hue, saturation and value were found.

Usually the range of hue is between 0º and 360º and both the range of saturation and value are between 0% and 100% but the OpenCV converts these ranges to [0, 179]

hue and [0, 255] for both saturation and value. This means that the maximum value presented in Table 5.1 is (121º, 100%, 100%) and the minimum value is (107º, 92%, 52%). Note that the green defined as (0, 255, 0) in the RGB plane is defined as (120º, 100%, 100%) in the HSV color model [2]. Finally, to prevent small changes in the color of the seal the interval was expanded shortly beyond the limits identified. The interval used in the inRange() function was:

- Upper limit: (65, 255, 255) or (131º, 100%, 100%);
- Lower limit: (50, 210, 100) or (101º, 82%, 39%);

The results of this step are shown in Figure 5.10b. The only noise presented in this image is the dot caused by the conveyor-belt motor heat and the reflection at the bottom, therefore this method was chosen.

5.4.3 Noise Removal

The next step is to remove the remaining noise on the image such as the small black dot and the bottle reflection presented in Figure 5.10b.

At this point, there are only 1 (white) and 0 (black) pixels in the image. A group of white pixels is an agglomeration of white pixels neighboring white pixels. In Figure 5.10b there are four groups: the seal, the conveyor-belt motor hot spot, the reflection and the line at the top (for more information on this, see Subsection 5.3). The first two groups are noise and in both, the group of white pixels is smaller than the seal itself. By establishing a minimum number of white pixels in a group in order for the object to appear in the image this noise is removed.

The threshold limit is 3000 pixels which means that any group of white pixels constituted by less than 3000 white pixels will be removed from the final image. At first glance it may seem a high value due to the low resolution of the camera but the seal is a circle usually thick that easily surpasses this value without losing any important information. A higher value will also remove other noise that could appear unexpectedly. The results can be seen in Figure 5.11. From this image, the image signature will be created as explained in Section 5.7.

The white line at the top of the image would also disappear when the method described is applied. Nonetheless, it does not contain any relevant information either way, the was removed from the image which means the image dimensions are 382 x 288 instead of the previous 382 x 289.

5.5 Frame Selection

In this project, there is no need to use all video images as input in the neural network. That would make it heavier and possibly slower than needed so. A single image per bottle will be used and this method greatly reduces the number of predictions to make.

[2] Rapid Tables (last access on 2021-12-04): `https://www.rapidtables.com/convert/color/rgb-to-hsv.html`

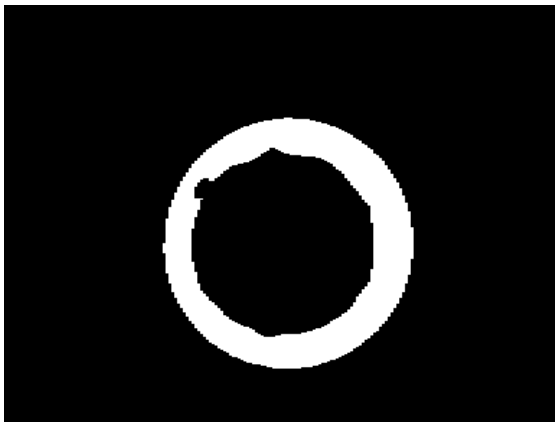


Figure 5.11: Figure 5.7 fully pre-processed.

The camera frame rate is approximately 27 frames per second. However, the production line serves no more than one bottle every two seconds, and most of the images can be safely ignored. Most of the video frames recorded do not contain any relevant information. The only important frames are those that show a clearly visible seal. Also only one image of each bottle is needed, therefore, all images that do not satisfy these requirements are discarded. The selected frame will then be used to generate the image signature that will be input to the neural network.

To determine if an image is relevant, a Region Of Interest (ROI) was defined using a constant named *DISTANCE_BORDER_TO_ROI*. Most of the time this ROI contains only black pixels but, once a seal appears, the pixels value change from 0 to 1 and this will trigger an action that will save this frame.

In order to define the ROI the circle radius was measured using the OpenCV HoughCircles() function. A single perfect circle was drawn above every seal in the outside layer as shown in Figure 5.12 and then the radius of this circle was stored in a text file. The data was then loaded into an Excel sheet where the biggest radius detected was identified. The value obtained is a radius of 99 pixels which means the diameter is 198 pixels. The seal must be fully visible in order to be valid and the images have 288 pixels of height so when subtracting the circle diameter, a vertical line with 90 pixels remains. From those 90 pixels, the bottom 10 pixels of the image were ignored to prevent problems caused by the reflection explained in Subsection 5.3. The distance from the top border of the image to the ROI is 80 pixels. Occasionally, information may be lost between the IR camera and the Image Acquisition module causing a frame freeze while the conveyor belt is still moving. A small *DISTANCE_BORDER_TO_ROI* constant is problematic in this situation because, the freeze does not have to be very long for the image to disappear from the ROI. This obliges that the distance from the top border to the ROI must the longest distance possible. The higher the value of the *DISTANCE_BORDER_TO_ROI* constant is, the longer is the distance from the image top border to the ROI and the longer the freeze has to be in order for the seal to disappear before being detected.

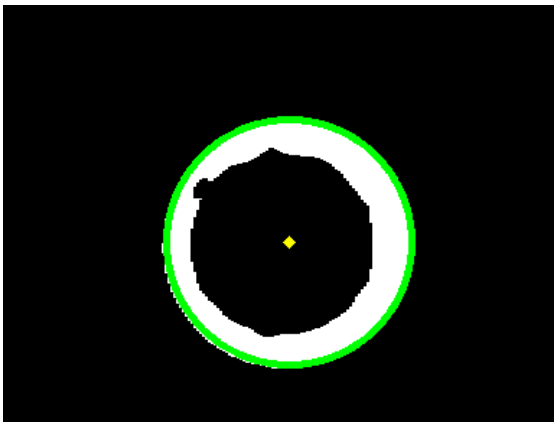


Figure 5.12: OpenCV HoughCircles() function applied to Figure 5.11.

The ROI can not be a single line in the middle of the image because, a seal without glue in the center would pass unnoticed if the vertical line was the only criteria to identify an object so, an horizontal line was added. The seals have on average 180 pixels of diameter so, 90 pixels were added to the vertical line and 90 pixels were subtracted to the vertical line.

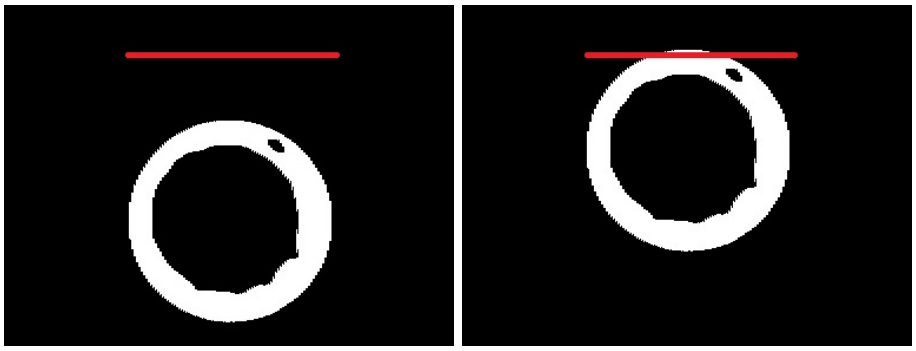
It was defined that the ROI is a 180 pixels horizontal line located 80 pixels from the image top border. When a pixel in that line is 1, it means there is an object and the frame should proceed for further processing. If no white pixels are found in that horizontal line then the frame is very unlikely to contain any relevant information and does not deserve further processing.

Figure 5.13 illustrates this process. The bottle moves upwards in the video according to the conveyor belt direction explained in the Section 5.1. Figure 5.13a shows a bottle appearing in the frame where the seal is already clearly visible but it has not reached the defined line. This frame is not used. Figure 5.13b presents the same bottle a few frames after Figure 5.13b where the seal reached the defined line. Here, the algorithm detects a change in the pixel value from 0 (black) to 1 (white) and it is assumed that there is a seal in the image. When the seal is detected in the image, an operation that stores this frame in the database is triggered.

5.6 Dataset Construction

From the prototype, four different videos were recorded and used attaining a total of approximately eight minutes. During this period fifty-nine bottles were recorded but only fifty-six are detected by the algorithm explained in the previous sections. Table 5.2 shows the number of bottles that were captured in each video.

Most bottles present no problem to the proposed algorithm, but there are some errors and special cases with those videos that must be addressed. There are three different situations in the videos:



(a) Frame ignored (b) Frame stored

Figure 5.13: Frame selection algorithm.

Table 5.2: General information about the videos recorded.

Video #	Sealed	Unsealed	Skipped	Total
1	15	0	0	15
2	23	1	1	25
3	5	1	0	6
4	11	2	0	13

- Normal Bottle
  This is the most common type of occurrence. These bottles have a cap with glue and are detected by the algorithm regardless of being correctly sealed or not.

- Uncapped Bottle
  This case represents bottles that have no cap, or have a cap without glue or seal (e.g. Figure 5.5c). Some of those bottles were introduced in the test videos, and they are not supposed to interfere with the present algorithm. Since this model depends on highlighting the green color of the hot glue to detect the bottle, the bottle passes unnoticed to the algorithm. This happens because the temperature of the environment is similar to the temperature of the plastic bottle making it impossible to identify the bottle by the color. It can only be identified because of the faint round shape presented in the image, but this is not sufficient for proper classification using the present method. These type of bottles require additional detection methods such as touch or motion sensors which are outside of the scope of this project to be detected. Also, the experienced operator assured that this situation rarely occurs and the main problem is to detect poorly sealed bottles.

- Unseen Bottle

  Throughout all videos, there are situations where the video acquisition freezes displaying the same frame for a period of time. This can be caused due to a poor connection between the infrared camera and the computer that is receiving the video. In most occurrences, this is not a problem because the freeze duration

is short and the movement of the bottle can be seen with just a slightly interference. However, there is one particular situation where the freeze duration is long enough to make the bottle disappear from the camera visual range before entering the region of interest, meaning that, the algorithm will not be able to detect the bottle.

Of the 56 images obtained, 53 represent correctly sealed bottles and only 3 are related to bottles with sealing problems that should be detected. Therefore, the dataset is small and also unbalanced. To improve the dataset size, a data augmentation technique is required.

The first option tested was to edit the images in order to make them different from each other and get a higher variety, but this method generates fake images that may or may not correspond to the reality. For this reason, this option was discarded.

The second option tested was the rotation of the already obtained images which creates real images of bottles. The angle of the rotation can also be adjusted to create a higher or lower number of new images. This methodology was selected.

Due to the fact that the dataset is not balanced, the number of rotations was adjusted according to the label type:

- Images representing a well sealed bottle were rotated 6 times, increasing 60 degrees per rotation.
- Images representing a poorly sealed bottle were rotated 24 times, increasing 15 degrees per rotation.

Applying smaller angle rotations to images of faulty seals makes sense, since those images are usually not symmetric. Images of well sealed bottles have a higher degree of symmetry, so less is gained when they are rotated.

Once the process was finished, the dataset contained 390 images, 72 of which represented poorly sealed bottles and 318 were well sealed bottles. The unsealed percentage increased from 5.35 % to 18.46 % of the entire dataset.

5.7 Input Data

There are several steps when preparing the images for input to the neural network. The first step is to convert the image to an appropriate size. The original images contain a total of 110016 pixels, so they are too large to use and can be down-sampled [22]. The higher the quality of the image the easier it is to identify flaws in the sealing process, but down-sampling the image to a size that still keeps most of the information allows for faster processing in real time. The present case uses thermal cameras which already have low resolution, so they were fast to process and no down-sample actions were performed.

Additionally, instead of using every pixel in each image as an input to the neural network, the sum of white pixels in each row and in each column was computed resulting in two arrays, one for the lines and one for the columns. These arrays are column and row signatures of the binary image. The second array is appended to the first and

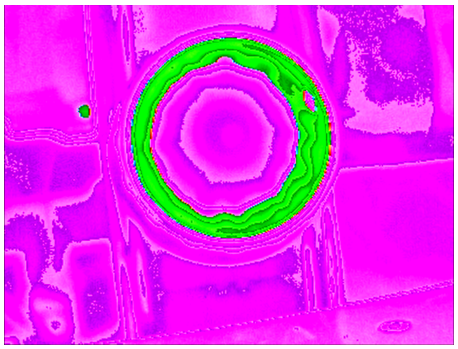
the resulting vector is called the **image signature** and it is used as input on the neural network.

Figure 5.14 shows side-by-side images related to the processing of two bottles. Figure 5.14a shows a thermal image of a well sealed bottle and Figure 5.14b shows a thermal image of a poorly sealed bottle. Figure 5.14c and Figure 5.14d are the binary representations of the images, obtained as result of the image processing steps previously explained. Finally, Figure 5.14e, Figure 5.14f, Figure 5.14g and Figure 5.14h are graphical representations of both components of the image signature. The charts show the vectors obtained by summing the images row-wise and column-wise. The row and column vectors are then concatenated and used as inputs of the neural network. A correctly sealed image will always present a distribution of pixels similar to the one presented in Figure 5.14e and Figure 5.14g and so, both the row and column signatures will resemble a parable on the middle of the chart. Defectively sealed bottles, however, will have different distributions of the pixels in the binary images and their signatures will differ from the parabolic shapes, as shown in the charts.

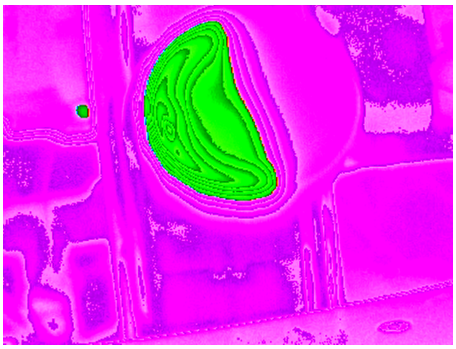
A Python script was created to read every single image from the dataset, generate the image signatures and and create two results:

- The row array is appended to the column array that is appended to the image ID.
- The image name is appended to the corresponding ID so each image can still be identified.

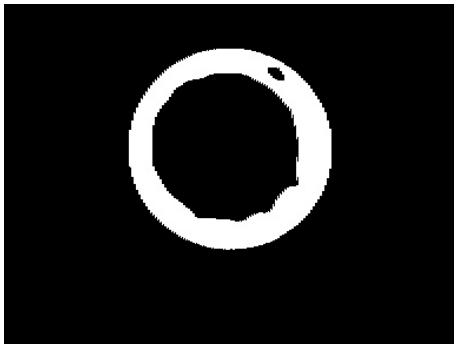
These results are then stacked vertically creating two matrices. The first contains 390 rows and only one column. It does not require a second column for the ID because it is appended in the beginning of the image signature array. The second contains 390 rows and 2 columns. The first column contains the ID and the second the image name. These matrices are then stored as a pickle file using the Python built-in library Pickle.



(a) Infrared image showing a well sealed bottle.

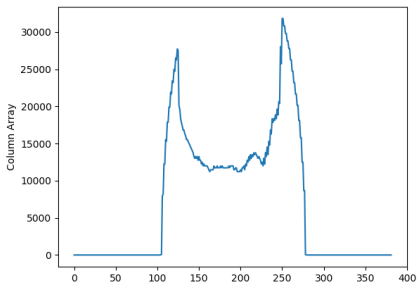


(b) Infrared image of ill sealed bottle.

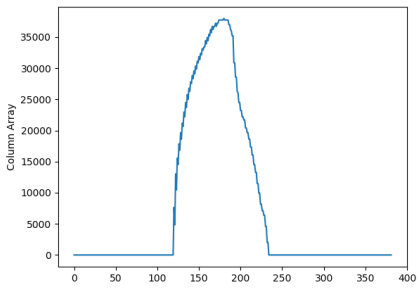


(c) Binary cleaned image of well sealed bottle

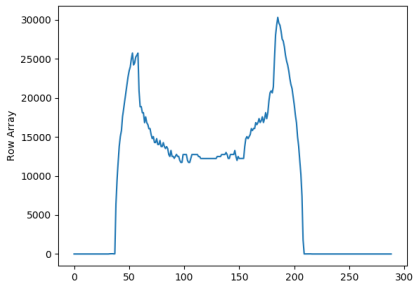
(d) Binary cleaned image of ill sealed bottle.



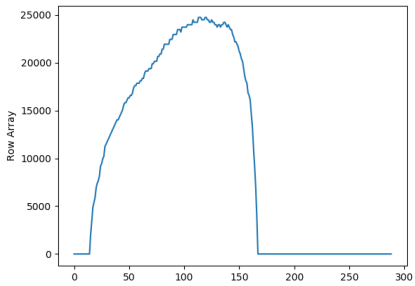
(e) Column signature of well sealed bottle.



(f) Column signature of ill sealed bottle.



(g) Row signature of well sealed bottle.



(h) Row signature of ill sealed bottle.

Figure 5.14: Comparison between the results obtained during the processing stages of a correctly sealed bottle and a defectively sealed bottle.

Chapter 6

ML Algorithm Selection and Model Training

In this chapter, the machine learning model is created. It includes the algorithm selection, the hyper-parameter optimization and the training and testing results.

6.1 Artificial Neural Networks

There are three types of machine learning techniques: supervised learning, unsupervised learning and reinforced learning [1]. Because the prediction result is easily identified from the image of the seal a supervised learning technique will be used.

There is a large variety of supervised machine learning algorithms available. Some examples are linear regression, naive bayes, support vector machines (SVMs) and artificial neural networks [2].

Artificial Neural Networks (ANN) were selected for the present system, considering, among others, the following reasons:

- ANN provide good performance as classifiers when the right parameters are used;
- They are well supported by the Python library Sklearn, which offers simple and efficient methods to implement ANN;
- ANN models are easy to train and adapt, so they can evolve over time if needed, or adapted if the bottle caps change.

The cons of this algorithm are:

[1] Microsoft Azure - Machine Learning Algorithms (last access on 2021-04-10): `https://azure.microsoft.com/en-us/overview/machine-learning-algorithms/`

[2] IBM Supervised Machine Learning Algorithms (last access on 2021-04-10): `https://www.ibm.com/cloud/learn/supervised-learning`

- When the problem has a high level of complexity, the algorithm also becomes complex and it can be nearly impossible to analyze and understand the decision - it is said that ANN can be black boxes due to this problem.
- They require the definition of a large number of parameters when comparing to the others supervised machine learning algorithms.

6.2 Multi-layer Perceptron

The model implemented in our system is the sklearn Multi-layer Perceptron (MLP) classifier. An MLP is a type of neural networks which contains multiple layers of perceptrons, normally connected with each other in a feed-forward way. It optimizes the log-loss function either using limited-memory BFGS (LBFGS) or stochastic gradient descent (SGD) algorithms [3].

The LBFGS was first described in Jorge Nocedal [23] where it was called SQN. It is part of the Quasi-Newton methods and optimizes using a Broyden-Fletcher-Goldfarb-Shanno (BFGS) but with limited amount of computer memory. In Dong C. Liu and Jorge Nocedal [24] the authors compare the LBFGS performance against two methods at the time and concluded that the LBGFS is faster.

The LBFGS has especially good results when the dataset is small which is the case in this project, however it cannot learn without training the whole dataset. For this project it is necessary a method that learns without having to train the whole dataset each time a mistake is found and for this reason, the SGD method was used, since it allows the use of the sklearn *partial_ fit()* function which is essential to solve this requirement. SGD is a simple way to train a model that fits linear classifiers and regressors under convex loss functions [4].

6.3 Hyper-Parameter Optimization

The MLP classifier includes multiple parameters such as hidden layer size, activation function, solver, learning rate and others. The full list of the parameters and functions of the MLP classifier can be seen in the sklearn official website link presented in Section 6.2. Those parameters were optimized through analysis and iterations and using the scikit-learn *GridSearch()* function algorithm. The *GridSearch()* function is a tuning technique that calculates the best values presented in list of hyperparameters used as input. Using the *GridSearchCV()* function alone could optimize the model to the training set generated which could cause over-fitting.

The first parameter studied was the number of perceptrons in the first hidden layer. Figure 6.1 presents results of this phase. Figure 6.1a shows the model best loss variation when increasing the number of perceptrons in the first hidden layer. Figure 6.1b

[3] sklearn MLP Classifier Official Documentation (last access on 2021-04-10): `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

[4] sklearn SGD Official Documentation (last access on 2021-04-10): `https://scikit-learn.org/stable/modules/sgd.html`

shows the performance changes when the number of perceptrons is incremented while using the test set. The figure shows that above 60 perceptrons the change in the model behavior is small.

When selecting the top values presented in Figure 6.1b, the *GridSearchCV()* would always select the higher value in range until it capped at 130 perceptrons. However, since the change in the model is small, only 100 perceptrons were used in the final model. The reason that led to this choice is to decrease the chance of overfitting and to reduce the computing power needed to test and train the models in the production line.

The second parameter tested was the number of perceptrons in the second hidden layer. Figure 6.2a presents the model best loss variation and Figure 6.2b presents the model performance when applied to the test set. Figure 6.2b shows that the model *accuracy* and *F1 score* are capped around 0.986 and 0.975 respectively. Based on Figure 6.2a, the value selected was 10 perceptrons since it marks the point where the model loss becomes almost constant.

The last parameter tested was the alpha regularization parameter value. Figure 6.3a shows the model best loss variation and Figure 6.3b shows the model performance when applied to the test set. Figure 6.3b shows the best results when alpha is greater or equal to 1,2. However, the best loss obtained with this value is high, as can be seen in Figure 6.3a. The *GridSearchCV()* function was also used to select the best value and the result was 1.2.

The MLP Classifier solver, activation function, hidden layer size and alpha are now defined. The final tests consist in finding the best learning rate via trial and error. The inverse scaling learning rate was selected and the grid search found the best learning rate.

Despite not being stochastic, LBFGS was also tested and it presented the best results. This happens probably due to the small dataset size that allows the solver to converge faster and perform better as mentioned in the sklearn MLP web page presented previously. If the continuous improvement of the ANN was not a requirement, this solver would be the one selected since it presented good results with only 30 perceptrons in a single hidden layer.

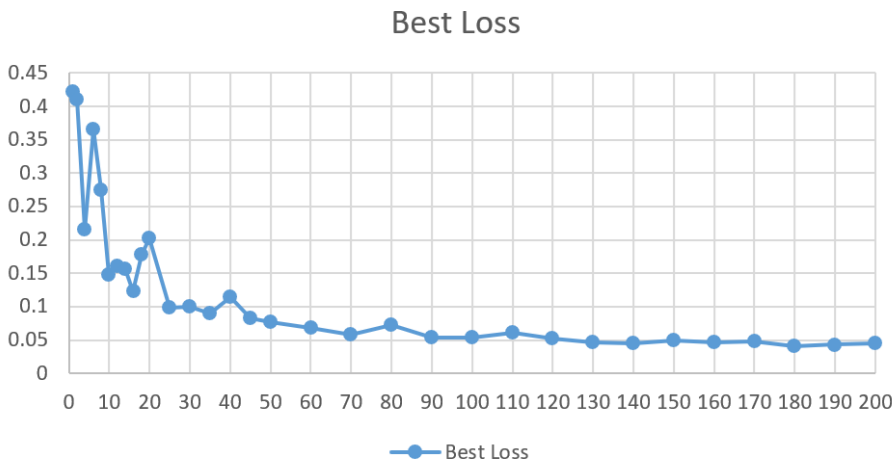
In summary, the neural net parameters are:

- Activation function: logistic sigmoid function;
- Solver: stochastic gradient descent;
- Hidden layer size: (100, 10)
- Alpha: 1.2;
- Learning rate: inverse scaling.

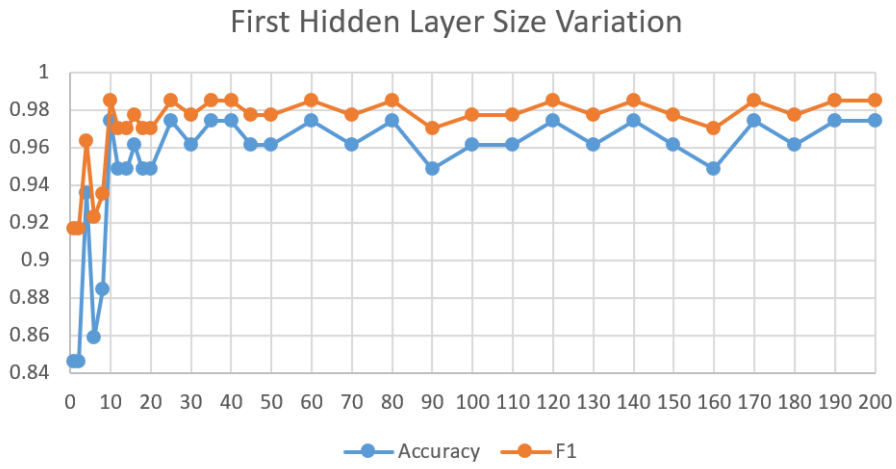
6.4 Training Methodology

The classifier was constructed through multiple iterations.

To construct the machine learning model, the pickle files generated in Section 5.7 containing the dataset (list of signatures) where first loaded. Due to the small size of

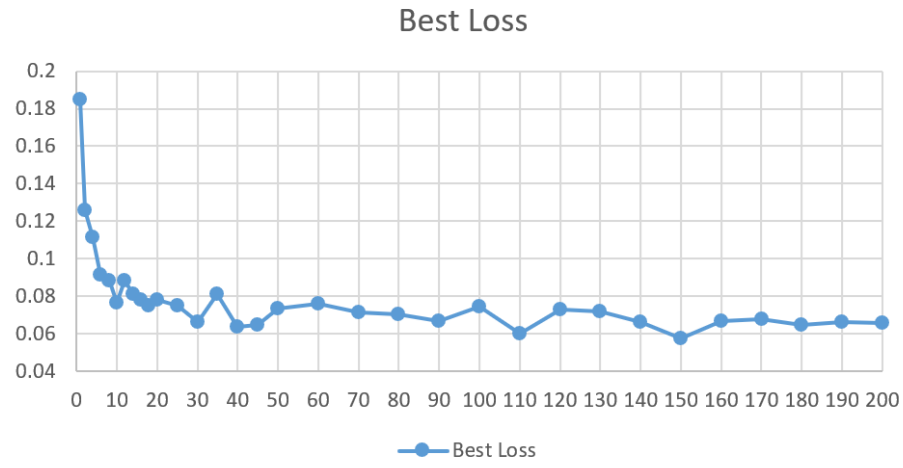


(a) Best loss variation

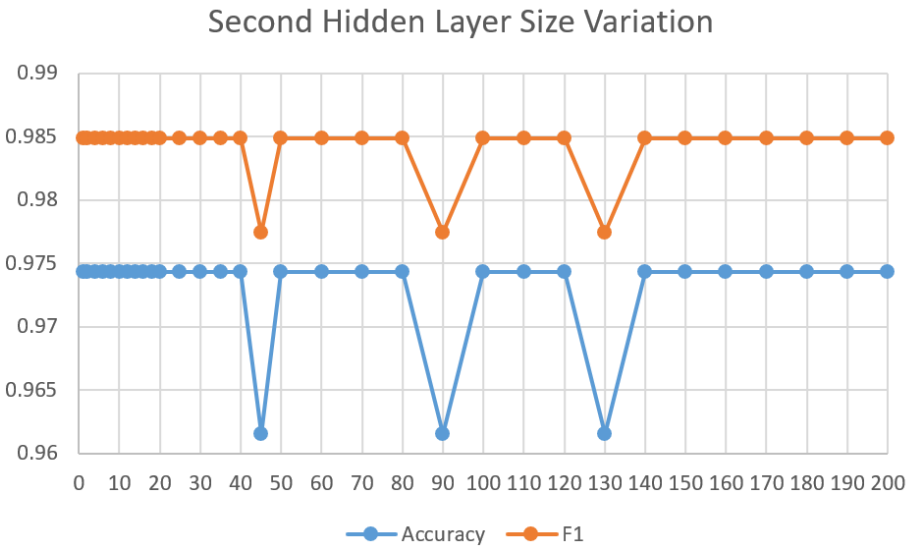


(b) *F1 score* and *accuracy* variation

Figure 6.1: Model behavior when changing the number of perceptrons in the first hidden layer.



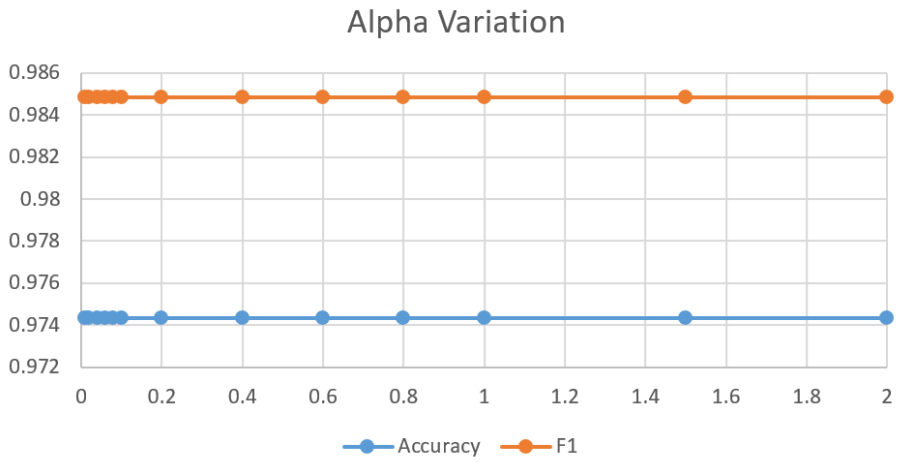
(a) Best loss variation



(b) *F1 score* and *accuracy* variation

Figure 6.2: Model behavior when changing the number of perceptrons in the second hidden layer.

(a) Best loss variation



(b) *F1 score* and accuracy variation

Figure 6.3: Model performance behavior when changing the value of alpha.

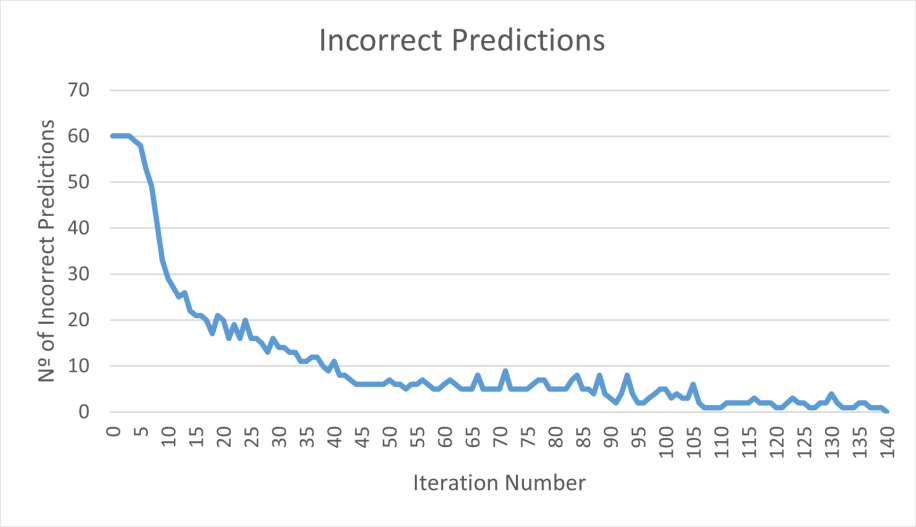


Figure 6.4: Variation of the number of incorrectly classified images during the training phase.

the dataset, 80% of the dataset is used in the training phase and only 20% in the testing phase. The training set is then used to fit the MLP classifier in the training phase. The training phase consists in a loop where in the first iteration the model is created and in each iteration (including the first one) the model is improved by partial fitting the signatures incorrectly classified.

The dangers of a pesticide leak create a necessity of have a model with no false negatives which present unsealed bottles that were classified as sealed. The goal of this process is to build a model without false positives and false negatives and the loop ends when either the goal is achieved or the maximum number of iterations (200) is reached. If there are false negatives the process should be repeated until the goal is reached. The result is stored in a pickle file for later uses.

Figure 6.4 presents the number of incorrect predictions counted in each iteration. In the first iteration, 60 images were incorrectly classified. This images are the unsealed bottles in the training set that were classified as sealed by the model on the first 5 iterations approximately. These incorrect classifications are then partial fitted and the model performance improves until the goal is reached on iteration number 140.

Figure 6.5 presents the evolution of the model *accuracy* and *F1 score* over the 140 iterations necessary to achieve the goal.

6.5 Classifier Tests and Results

The model was trained with 312 images of the augmented dataset. The model was then tested using the test set, which contains the remainder 78 images.

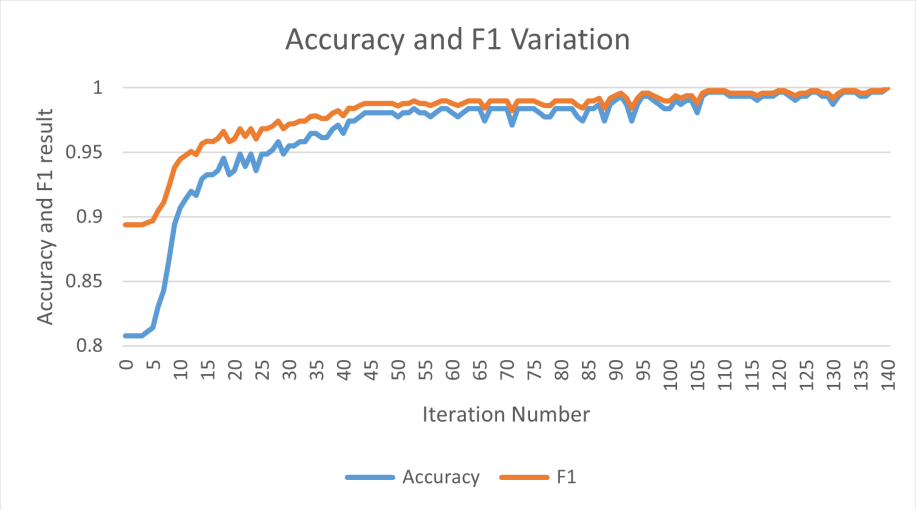


Figure 6.5: Variation of the model *accuracy* and *F1 score* during the training phase.

Table 6.1: Confusion matrix for the training set.

	Actual Positive	Actual Negative
Predicted Positive	60	0
Predicted Negative	0	252

There is one important aspect to consider when creating the neural net and that is, the number of false negatives must be as low as possible. Each time there is a false negative, an unsealed bottle goes to the distribution. In this case study, the bottles contain pesticide which makes it dangerous for the public to have defectively sealed bottles going to distribution. Therefore, the model was developed with the ability to be learn over time and receive additional train, in a way that false negatives are avoided.

The maximum training iterations was set to 200. However, the results shown in the present paper were obtained with a model that required only 57 iterations in the training phase.

Table 6.2: Confusion matrix for the test set.

	Actual Positive	Actual Negative
Predicted Positive	12	1
Predicted Negative	0	65

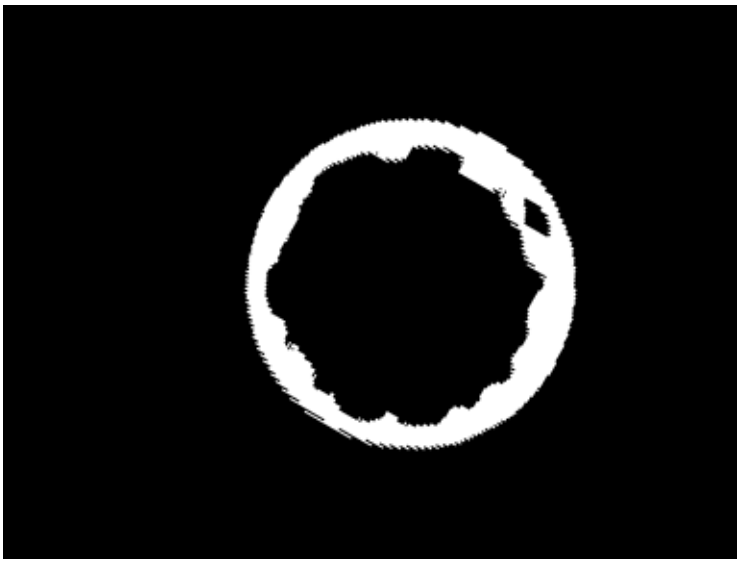
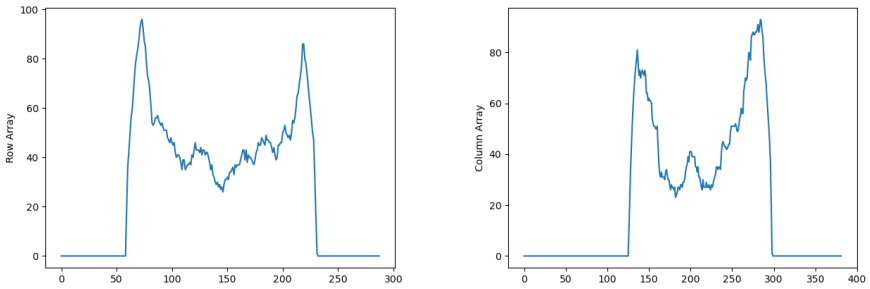


Figure 6.6: Incorrectly predicted image by this model in the testing phase.

Table 6.1 presents the resulting confusion matrix for the model created using the training set. Table 6.2 presents the results of the same model for the test set. The model performed no errors in the training set but it had a single false negative in the test set that can be seen in Figure 6.6.



(a) Row Array

(b) Column Array

Figure 6.7: Signature of the incorrectly predicted image.

Figure 6.6 shows a well sealed bottle that was classified as defectively sealed and would be removed from the production line. The glue layer in this bottle is thinner and more dented than the one in Figure 5.11 and this may be the cause of this result. The human operator classified it as acceptable, but this is a borderline situation considering

the thinness and irregularity of the seal. Figure 6.7a and Figure 6.7b show the image row and column signatures. They are very different from the signatures of better sealed bottles as shown in Figure 5.14g and in Figure 5.14e.

In the production line implementation, the model will be partial fitted to classify this image correctly and improve the performance. That can be achieved using the *partial_fit()* function provided by the scikit-learn library, so that the network is trained a number of times until it classifies that particular image correctly.

The model described above has a precision of 98.6 % and a recall of 100 % for this particular dataset.

$$Accuracy: \frac{12+65}{12+1+0+65} = 0.987$$

$$Specificity: \frac{65}{65+1} = 0.985$$

$$Sensivity/Recall: \frac{12}{12+0} = 1$$

$$Precision: \frac{12}{12+1} = 0.923$$

$$F1Score: \frac{2\times 12}{2\times 12+1+0} = 0.96$$

The model created is a good model not only because it presents good results in all metrics tested but because it satisfies the requirement of not having false negatives. The model can evolve over time in order to correct mistakes. The precision value is the lower value calculated and it is still above 0.92. It is a flexible system that can also be trained to deal with different types of bottles or caps, and can be adjusted when the position of the thermal camera needs to be changed.

Chapter 7

System Implementation

The project implementation is detailed in this chapter. The environment used to develop this system is described as well as the libraries used in each module of the architecture. Once described the development environment and the tools and technologies used, the application code is described.

7.1 Programming Language

This project requires a programming language that can be used to create real-time computer vision processes, build machine learning models, create a web server and connect to a database. Python, Java, R and Javascript are four possibilities of programming languages that can be used in this project.

According to the Stack Overflow survey from 2019 [1], "Python is the fastest-growing major programming language today". Python is a free and open-source language created in early 90's that has good support to all the requirements stated above. It has extensive libraries supporting Machine Learning and Artificial Intelligence, while, at the same time, providing a simple development environment and a short-learning curve.

R is used for statistical modeling and machine learning and can be used by people with no prior programming experience in these areas but it is not a suited to develop backend. Javascript is a language that performs exceptionally good in frontend but is not open-source. Like R, it less used in backend but, unlike R, it has frameworks such as Node.js that simplify the backend development using this language. Since the goal is to develop a complete prototype of the system including backend and frontend, Javascript and R are excluded because they are not suited to develop client-server applications when compared to Python or Java.

Java is the best alternative for Python in this project because it is a language that supports all the requirements but the learning curve it is longer when compared to Python. Python presents a large variety of machine learning and image processing

[1] Stack Overflow (last access on 2021-04-10): `https://insights.stackoverflow.com/survey/2019#technology`

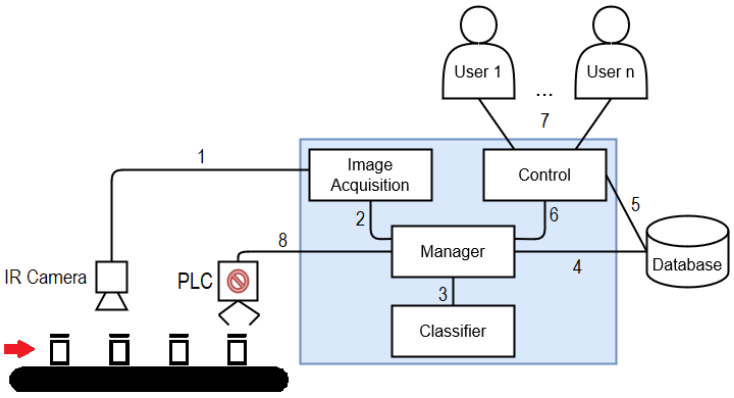


Figure 7.1: Architecture implementation.

libraries which Java does not include, therefore the programming language used in this project is Python.

7.2 Technologies Used

In this section, the technologies and libraries related to each module of the architecture presented in Chapter 4 are detailed.

Figure 7.1 presents the architecture with the connections between elements numerated. The figure contains eight connections which represent:

1. USB connection that links the infrared camera to the Image Acquisition module and is used to send the video captured in the camera to the module for further analysis;

2. TCP socket that connects the Image Acquisition module to the Manager module and is used to send the single frame from each bottle;

3. TCP socket that connects the Manager module to the Classifier module and is used to send image signatures to either be classified or partial fit the machine learning model;

4. Connection to the database using the Python library psycopg2 in order to add new entries for the classified images;

5. Connection to the database using the Python library SQLAlchemy in order to load images to the interface and update the image state value;

6. TCP socket that connects the Control module to the Manager module and is used to send image signatures to be partial fitted;

7. HTTP connection used by the clients to access the Control module interface;

8. Connection used to send a signal to the PLC to remove the unsealed bottles. This connection was not implemented since it is outside of the scope of this project.

Neither connection 1 nor connection 8 were part of the scope of the project meaning they were not implemented. Connection 1 was simulated using the list of videos captured.

7.2.1 Image Acquisition Module Technologies

The Image Acquisition module reads the video received from the infrared camera containing images of the seals, highlights the seal, converts the image to black and white, removes the existing noise, and selects the frame where the bottle is on the center. All these tasks are done using multiple functions from the following packages: Numpy, OpenCV and Skimage.

7.2.2 Manager Module Technologies

Similarly to the Image Acquisition module, the Manager module requires the OpenCV and NumPy packages.

In this module, the signature from the image received from the Image Acquisition must be calculated in order to send it to the Classifier module. The Manager module reads the image using OpenCV functions and the signature array is created using the NumPy.

The Manager module must also be connected to the database in order to add entries to the image table. This is achieved using the psycopg2 package.

7.2.3 Classifier Module Technologies

Python presents a wide variety of libraries for machine learning projects such as TensorFlow [2], scikit-learn [3] and Theano [4].

Scikit-learn is open source and mainly implemented in Python. It has algorithms already implemented, contains a good variety of metrics to evaluate the model performance and it has a vast documentation.

Theano is written in Python while Tensor Flow is written in C++ and Python and they are both open source. Unlike scikit-learn who only runs on CPU, Theano can both run on CPU or GPU. Theano requires the full in-depth implementation of the algorithms and are mostly used for deep learning and bring a higher complexity with them.

In this project, the scikit-learn library was selected because from the alternatives presented scikit-learn library is the easiest to learn and use. Despite Theano and Tensor Flow having faster compiling and execution speeds, the artificial neural network developed in this project is simple and the performance boost will most likely pass unnoticed. Theano or Tensor Flow would be better if the neural network input was the

[2] TensorFlow Official Website (last access on 2021-04-10): `https://www.tensorflow.org/`
[3] scikit-learn Official Website (last access on 2021-04-10): `https://sklearn.org/`
[4] Theano Official Website: `http://www.deeplearning.net/software/theano/`

complete image (288 *times* 382 = 110 016 inputs) instead of the image signature (288 + 382 = 670 inputs) for example.

7.2.4 Control Module Technologies

The web interface required in this project has limited access — it should only be allowed access to people working in the quality control stage of the production line. It does not require a complex interface, in fact, a simpler interface will simplify the task completion. It requires the core functionalities of a website — such as log in, log out, add users, and other — and the specific case of study functionalities - such as loading the classified images, verify the prediction result, identify the errors and retrain the classifier and check the system generic information.

There are multiple frameworks for web development in Python such as Django [5], Flask [6], Bottle [7] and many others. Django and Flask are the most used and have a big community of active users. A vast documentation is available for both which simplifies the process of solving problems. While Flask is a minimalist web-framework that depends on the addition of external libraries to create a complete application, Django comes with a large amount of inbuilt libraries.

Due to the built in libraries, Django is regarded as the most secure because by default provides protection against various threats such as XSS (cross-site scripting) protection and SQL injection for example [8].

Bottle is a micro web-framework created to be fast, simple and lightweight. Like Flask, it depends on external libraries that must be added to the project but, unlike Flask or Django, the community and the documentation available are small.

Flask was the framework selected because is minimalist (like Bottle) but well documented (like Django) which shortens the learning curve and problems are solved faster. It allows the development of an basic application using only the libraries that are essential to the system. This obliges the developer to know all used libraries and the reason why the system needs them which for educational purposes is essential.

7.2.5 Database Technologies

PostgreSQL[9], MySQL[10] and MongoDB[11] were the management systems analyzed due to their open source nature and popularity.

MongoDB is a NoSQL database management system which has high performance and offers the ability to scale horizontally if needed but does not follow the ACID model which leads to less structured data. It main uses are related to high volumes of data and real-time data.

[5] Django Official Website (last access on 2021-04-10): `https://www.djangoproject.com/`

[6] Flask Official Website (last access on 2021-04-10): `https://palletsprojects.com/p/flask/`

[7] Bottle Official Website (last access on 2021-04-10): `https://bottlepy.org/`

[8] Django security features (last access on 2021-04-10): `https://docs.djangoproject.com/en/3.2/topics/security/`

[9] PostgreSQL Official Website (last access on 2021-04-10): `https://www.postgresql.org/`

[10] MySQL Official Website (last access on 2021-04-10): `https://www.mysql.com/`

[11] MongoDB Official Website (last access on 2021-04-10): `https://www.mongodb.com/`

Both PostgreSQL and MySQL support the ACID model and are widely used in the industry. These are two options that could be used in this project due to the simplicity of the database which contains only two tables with no relation between them but PostgreSQL was the chosen option because it contains PL/Python procedural language which allows PostgreSQL functions to be written in the Python language as stated on the official website [12]. This feature is not used but, since the project is developed using Python only, it is interesting to have compatibility between the language and the database. PostgreSQL also supports encryption algorithms which can be used to add a new layer of security to the application and also the source code ccan be modified by the developer to easily satisfy his needs without the need to send back the changes since PostgreSQL is licensed under BSD.

7.3 Python Virtual Environment

In order to be able to run this application, the virtual environment needs to have the following packages installed alongside with its dependencies: opencv-python, scikit-image, imutils, psycopg2, scikit-learn, Flask, Flask-SQLAlchemy, Flask-Bcrypt, flask-login, Flask-WTF and email_validator.

The opencv-python and scikit-image are packages necessary to the image processing. The imutils package was used to rotate the 53 images obtained from the four videos in order to increase the number of images on the dataset. The package psycopg2 is a PostgreSQL adapter for Python. The scikit-learn is a package necessary for the classifier. The remaining are all used in the Control module interface to translate python code to HTML, connect to the database, encrypt passwords, and other functionalities.

These packages were installed in the command line via PIP which, unlike PyCharm, will install all the dependencies of each package. For example, opencv-python will also install NumPy and Flask will also install MarkupSafe, Jinja2, Werkzeug, click and itsdangerous.

7.4 Hardware and Software Used

This project is contained in two different machines to simulate a centralized database located outside the system machine most likely in a server running Linux. The use of a centralized database is a common practice in the industry. It keeps all data located in the same placed with easy access and removes the need of creating a database for the many applications used by the company. This centralized database stores data for or from many different applications. This project was developed using a personal laptop which limits the hardware available to simulate this environment. A virtual machine will be used to simulate this centralized database inside a host machine.

In the following sections the hardware and software used in each machine is detailed.

[12] PL/Python - Python Procedural Language (last access on 2021-04-10): `https://www.postgresql.org/docs/9.0/plpython.html`

7.4.1 Host Machine

This project was developed on a personal computer with an Intel Core i5 (8th Gen) and a Windows 10 operating system.

All Python development was done in the host machine using the IDE PyCharm. A single project was created but it was divided in four parts, one for each module described. DBeaver software was used to administrate the database which is located on a virtual machine described in the following subsection.

7.4.2 Database Host Machine

The use of a virtual machine enables the possibility of creating multiple customized guest machines inside a host machine through virtualization. VMware Workstation Player, Oracle VM VirtualBox and QEMU are three hypervisors currently used that can be used to create the virtual machine. VMware[13] is free for personal use but it is not open-source. QEMU[14] is open-source but optimized to be run on a host machine running Linux which is not the case. When compared to VMware and VirtualBox are more commonly used than QEMU. VirtualBox[15] is free and open-source. It supports snapshots which can be used to restore the machine to a previous state. The hypervisor used in this project is VirtualBox.

The database is located inside the organization network but outside the computer that is hosting this system as presented in Figure 7.1. Virtual Box allows the configuration of a internal network that enables the communication between the host and the guest machine to simulate this environment[16].

This could also be interpreted as a simulation of a database in the cloud which is a practice that is becoming more common because it discards the need of having an IT team responsible for maintaining the machine where the database is hosted and also increases the uptime of the database. Opening the organization network to cloud communication raises security issues because it implies the addition of configurations to the firewall in order to enable the communication. When working with cloud based applications is essential to have a good encryption. The management of public and private keys between the involved elements is also essential. A cloud based database can be seen as a downside because it increases the complexity of the system creating the necessity to have a good security oriented IT team to manage these aspects which can originate threats when placed in the hands of someone inexperienced.

The virtual machine is supposed to simulate a server and the most common operating system for servers is Linux. To simulate a realistic environment, the operating system installed in this virtual machine is Linux more specifically the Debian distribution. Since the machine acts as a server it only need a command line interface to allow interaction which makes 32 MB of video memory enough. The machine needs a connection to the internet in order to simplify the tasks of updating the operative system

[13] VMware Official Website (last access on 2021-07-07): `https://www.vmware.com/`

[14] QEMU Official Website (last access on 2021-07-07): `https://www.qemu.org/`

[15] VirtualBox Official Website (last access on 2021-07-07): `https://www.virtualbox.org/`

[16] Virtual Box documentation - Virtual Networking (last access on 2021-04-10): `https://www.virtualbox.org/manual/ch06.html`

and installing software such as PostgreSQL. The RAM, number of CPUs and storage of the virtual machine were defined based on the host machine properties - 6144 MB of RAM, 1 CPU and 20 GB of fixed disk storage.

The PosgreSQL service was installed and enabled. A static IP was defined in the virtual machine hosting the database which allowed the access to the database from other machines in the network configured in the VirtualBox settings. The database can be accessed from the host machine running Windows through the use of this static IP and the PostgreSQL standard port 5432.

7.5 Database

The PostgreSQL database is hosted at the virtual machine described in Section 7.4.2. It contains only two tables - images and users - with no relation with each other as seen in Figure 7.2.

images	
id	**serial**
name	varchar(20)
prediction	varchar(1)
date	varchar(20)
time	varchar(20)
state	varchar(1)

users	
id	**serial**
username	varchar(20)
password	varchar(20)
email	varchar(20)

Figure 7.2: Database structure.

7.5.1 Table Images

This table contains all information related to the images classified by the machine learning model in the Classifier module. The table is composed by six columns:

1. ID - it is a serie and represents the unique identifier for the image;
2. Name - it is a VARCHAR(20) and represents the name of the image which is something like '*img* < *number* > *.png*'.
3. Prediction - it is a VARCHAR(1) and represents the result of prediction done by the machine learning model. It is either 'y' (Yes) or 'n' (No) and this answers the question "Is the bottle sealed?".
4. Date - it is a VARCHAR(20) and represents the date when the registry was inserted in the format 'yyyy-mm-dd'.
5. Time - it is a VARCHAR(20) and represents the time when the registry was inserted in the format '<hour>h<minute>m<second>s'.

Table 7.1: Five states created to characterize an image.

ID	Name	Meaning
o	original	Images classified and stored in the database
s	seen	Images correctly predicted validated by the operator
r	retrain	Images incorrectly predicted identified by the operator
d	done	Retrained images that were initially incorrectly classified

6. State - it is a VARCHAR(1) and represents the status of the image according to what was detailed in the previous section(Section 7.5.2). It can be only one of the following letters: o (original), s (seen), r (retrain) or d (done).

In this table, none of the columns can be null and the 'name' should be unique to prevent the existence of two different images with the same name.

Four states were created and every image can only have a single state at a given time. The images can however change between states. The five states are:

1. Original ('o') - represents an image that was classified and stored in the database but the classification result was not yet verified by the user.

2. Seen ('s') - represents an image verified by the user whose classification result is correct.

3. Retrain ('r') - represents an image incorrectly classified that must be retrained (partial fitted) to improve the neural network performance.

4. Done ('d') - represents an image that was retrained.

Table 7.1 summarizes the five states previously explained.

7.5.2 State Changes

There are three paths that can describe the changes of state of an image:

- o $\rightarrow$ s;
- o $\rightarrow$ r $\rightarrow$ d;

When an entry is added to the database by the Manager module it contains the ID, name of the image, date, time and state. The value of the state at this point is 'o' (Original). The operator then accesses the web application to validate the predictions made by the neural network and two situations can occur:

1. The prediction is correct: the operator does not have to perform any action over this image and simply ignores it.

2. The prediction is wrong: the operator has to select the image check box in order for it to be retrained.

Once the operator verified all the loaded images he confirms the operation by clicking the 'Retrain' button. The system then searches for all selected check boxes and the state in the database entry of each selected image is changed to 'r' (Retrain). The remaining images correctly predicted have their state updated to 's' (Seen) and this is the last state they will have.

In the standard user homepage, the operator can then click on the 'Update Classifier' button which will search in the database for all images with status 'r' and then retrain them using the sklearn *partial_fit* function. The incorrectly sealed image will then get its status updated to 'd' and this is the final status they will have. This change of states creates a historic in the database where it is possible to see which images were incorrectly predicted because the poorly predicted images will have 'd' as its the final status and the correctly predicted images will have 's' as its the final status.

7.5.3 Table Users

This table contains information related to the users who access the web application. Each user account must have an ID, a username, a password and an email which means this table will have four columns.

The four columns are:

1. ID - represents the unique identifier for the user account. It is a serie because this value can not be repeated and should always increment by one.

2. Username - represents the username that will be used to log into the application. Usernames are usually short so it was decided they should have a maximum of 30 characters which is usually more than enough. It is a VARCHAR(30).

3. Password - represents the password that will be used to log into the application. The password is hashed before being stored which increases the size of the string that will be stored and, for this reason, 60 characters were defined as the maximum. It is a VARCHAR(60).

4. Email - represents the email the user associates to this account and where he will be receiving a new password in case he forgets the login details for example. Emails are usually longer than usernames so it was decided that 60 characters would be the maximum. It is a VARCHAR(60) and

In this table, the email column can be null because the administrator account does not have an email associated.

7.6 Detailed Implementation

In the following subsections, the logic behind the system behavior is detailed according to the modules presented in the proposed architecture. The database implementation details will also be described.

The communication between them must be described will be described first and then the implementation details will be presented.

7.6.1 Modules Interaction

To communicate between modules, TCP was selected over UDP because the data integrity must be assured. These connections are created using the socket library which is a Python built-in library. To create a TCP connection the sockets used must be from the SOCK_STREAM type which is used to declare a TCP socket. For practical reasons the family of the created sockets is always AF_INET which implies the use of IPv4. Each connection is composed by two and only two modules as described in Section 7.2 on behaving as a server and the other as a client. Three ports were used in this project:

- Port 6000: bond to the Manager module server socket to where the Image Acquisition module client socket will connect;
- Port 6001: bond to the Manager module server sock to where the Control module client socket will connect;
- Port 6002: bond to the Classifier module server socket to where the Manager module will connect.

The data send in each connection will be detailed in the following subsections.

7.6.2 Image Acquisition Module Implementation

The Image Acquisition module contains the entirety of the code explained in Section 5.

For the present project, the image receiving is simulated using the four videos used to compose the dataset.

For the present project, the four videos recorded were used to simulate the infrared camera behavior.

Each time this module starts, it tries to establish a TCP connection with the Manager module at port 6000. If the Manager module is listening, the connection is established.

First the Image Acquisition module asks the Manager module to query the database and retrieve the last ID value used to name a classified image. This ID is essential to start naming the next received images without reusing names which will cause conflict database. The server then sends the answer and the connection is closed.

The operations described in Section 5 are then performed - highlighting the seal, converting the image to black and white, identifying the object in the ROI and storing the frame locally until it is finally sent to the Manager module.

The frames stored are named as 'img<ID>.png' where the <ID> is a unique identifier number that will increment for each frame that is stored. In case, the system is configured to work with a single production line but, if in the future the company expands to two or more production lines, there could be images with the same name. This problem can be easily solved by changing the image name in order to follow the template 'line<ID>_img<ID>.png' allowing the system to be scaled up.

To conclude, a new connection with the Classifier module is created and the processed image is sent using the protocol shown in Figure 7.3.

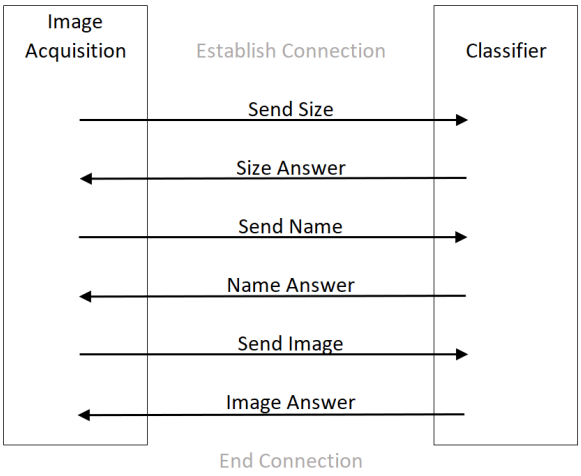


Figure 7.3: Communication protocol used to send the image from the Image Acquisition module to the Classifier module. This is implemented using TCP sockets.

Table 7.2: Types of errors handled.

Type	Message Logged
Error 1	ERROR: [date] > Invalid data received in the size request.
Error 2	ERROR: [date] > Invalid data received in the name request.
Error 3	ERROR: [date] > The created image size [new size] does not correspond to the received image size [size received].

7.6.3 Manager Module Implementation

When this module starts it connects to the database using the psycopg2 package and creates two TCP socket listening at port 6000 and 6001.

If a connection is established at port 6000, then there is a communication between the Image Acquisition module and the Manager module. The message received on the first received message is decoded and based on this first message, an operation will be performed. If the message is "*GET_LAST_ID*" it will query the database, get the ID of the last image and send it to the Image Acquisition module. Else, it means the an image will be sent according to the protocol presented in Figure 7.3. The protocol is explained in detail on the diagram presented in Figure 7.4.

If in any of the steps related to receiving the size, name and the image an error is found, it is written in the logs alongside the identification of the step of the protocol not completed as demonstrated in Table 7.2.

Once received, the image signature is calculated and sent to the Classifier module to be classified. Once the result is received, an entry is added to the database containing the information related to the image as described in Subsection 7.5.1.

The 'id' is the same ID used to name the image, the 'prediction' is the classification

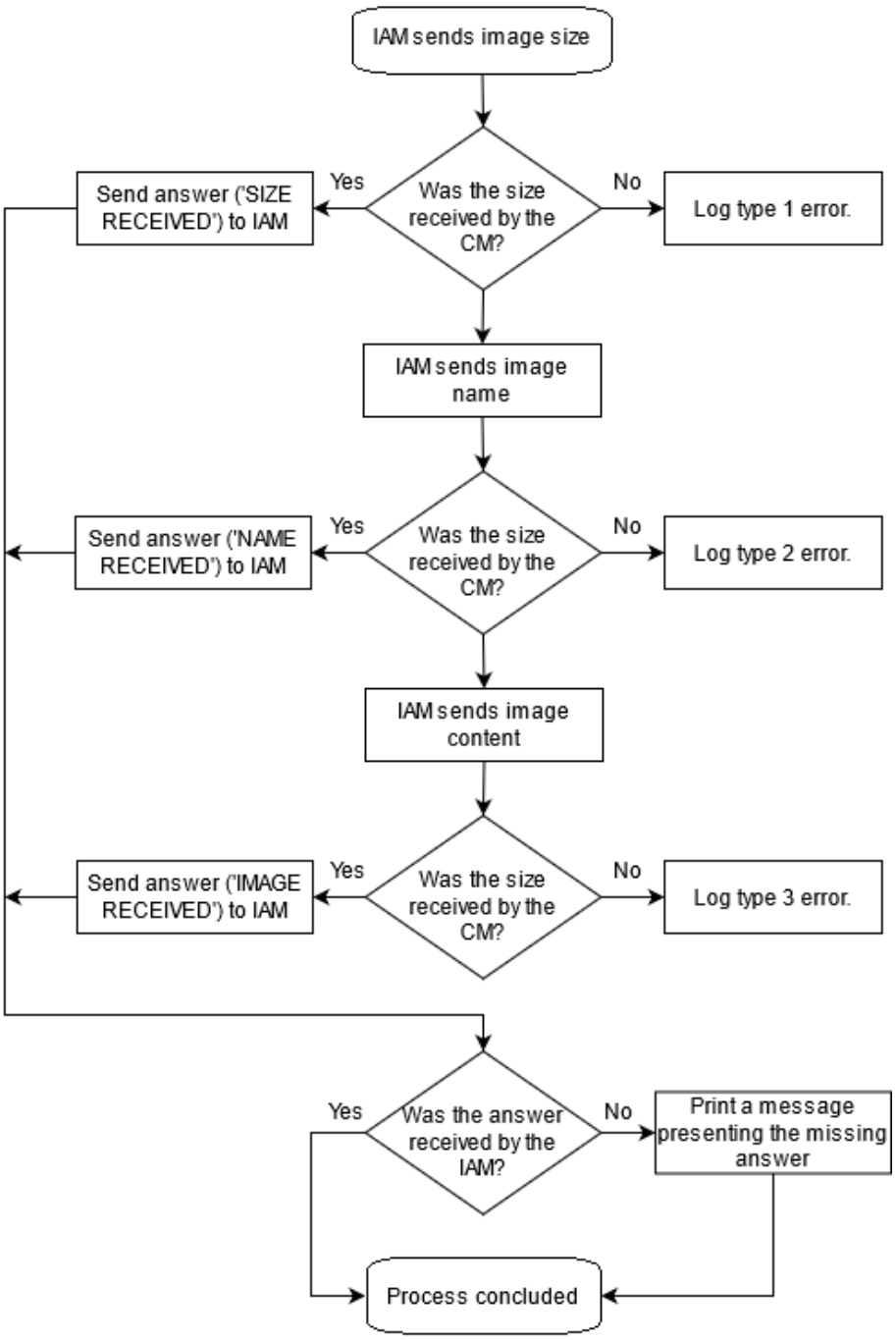


Figure 7.4: Diagram presenting the process of sending images from the Image Acquisition module (IAM) to the Classifier module (CM).

result - either sealed ('y') or unsealed ('n') - the 'date' and 'time' are the date and time when the image was stored in the database and the 'state' is an identifier of the state of the image.

If a connection is established in port 6001, then there is a communication between the Control module and the Manager module. The message received on the first received message is decoded and based on this first message, an operation will be performed. If the message is "*GET_STATUS*", the Control module is verifying if the Manager module is active so the answer sent is "ON". Else, an image signature is being received to be sent by the Manager module to the Classifier module in order to correct a mistake done by the machine learning model and partial fit the signature to improve the model performance.

Note that the image stored in the database is the final processed image, not the original colorized image. The average size of the processed image in the dataset is 6 KB. This value is 22 times smaller than the original image average size (133 KB). The processed image still contains all the relevant information without the background noise which simplifies its analysis.

7.6.4 Classifier Module Implementation

This module only acts as a server. At the start, it creates a socket listening at port 6002. To this port, only the Manager module connects to perform to actions: classify a new image or retrain a image classified incorrectly.

The first message received contains the operation identifier. If the message is 'CLASSIFY', the operation to be executed is the classification of a new image. The next message received by the Classifier module is the signature of a new image which is then classified and the result is sent back to the Manager module. Else the message is 'RETRAIN' and the image must be partial fitted. The next message received by the Classifier module contains the image signature and the incorrect classification result. The classification result is used to define is the image is a false positive or a false negative and image signature is partial fitted. To conclude, a new classifier is created an stored in the system replacing the old one and a generic message is sent to the Manager module indicating the process was completed. If an error occurs a different message will be sent and the Manager module will known when an error has occurred.

7.6.5 Control Module Implementation

This module contains the Flask web application that is accessed via web browser. Fifteen Flask routes were created to implement the system functionalities. These routes were created according to the use cases defined in Section 2.4. Each use case leads to a different route, and there are some additional routes which are not represented in the use cases. The routes are:

- login – displays the login form where a user can inserts his credentials and access the application;
- logout – logs out the user blocking his access to the application functionalities redirecting him to the log in page;

- recovery – displays the recovery form where a user can insert his email to retreive the login credentials.

The administrator routes are:

- admin – displays the administrator homepage where the following operations can be accessed: create a new user, list all users, remove a user and display server logs;
- register – displays the registration form where a new user can be added to the database using the company email, username and a temporary password;
- list_users – displays a table where all user accounts are listed and the table contains three columns: account ID, username and email;
- remove – displays the remove user form where a user can be deleted from the database using the account ID;
- log – display Control module logs in the screen.

The operator routes are:

- home – displays the operator homepage where the following operations can be accessed: change password, verify classified images by batch, verify classified images by date, update classifier, display generic information and logout;
- changepw – displays the change password form where a user is able to change the password;
- prediction – displays a form where the user selects a date, time and classification result in order to get images classified at a specific time period.
- gallery_by_date – this route requires the completion of the previous because it uses the inserted date, time and classification result to query the database and get a batch of images that loaded in the interface to be analyzed;
- gallery_by_batch – prompts the user to select a classification result and then queries the database to get a batch of images using the FIFO methodology and then loads it to the screen where the user is able to analyzed them.
- info – displays information about the system status, number of images in the database and the details of the last image classified;
- update – updates the machine learning classifier by retraining the signalized images.

Changes in the database are done through two python classes Users and Images linked to the SQLAlchemy. These classes map the tables from the database and contain the exact number of attributes each entry has (i.e. for users: username, email and password).

The starting screen of the application contains the log in functionality as it can be seen in Figure 7.5.

The credentials used in this page will be used to differentiate between the administrator and the operator.

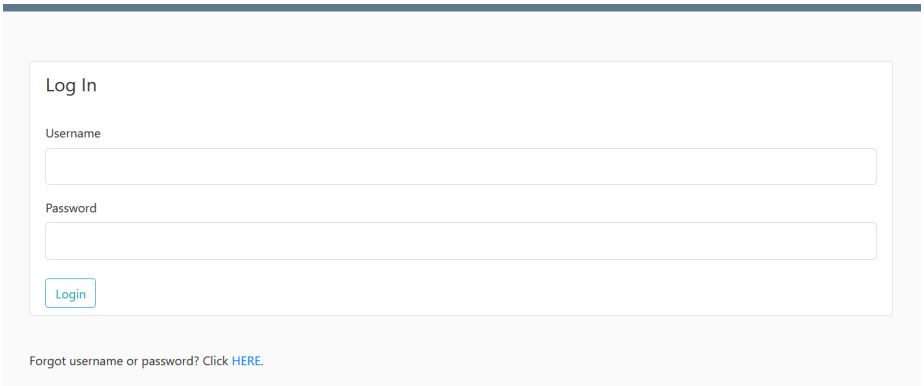


Figure 7.5: Log in page in the control interface.

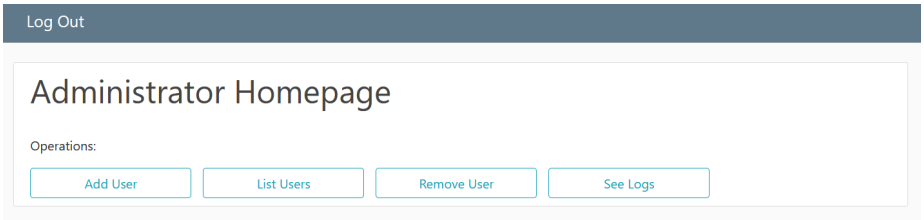


Figure 7.6: Administrator homepage in the control interface.

Administrator

Figure 7.6 presents the administrator homepage. The administrator can execute the following unique tasks: add user, list users, remove user and read server logs.

- Add User

  The administrator will use the organization email and username of the operator and a random password to create his account. The account is added to the database using the SQLAlchemy. Once the account is created, the operator must log in and change his temporary password.

- List Users

  Displays a table containing all accounts available that have access to the application. The table contains details such as the account ID, the username and the email. The administrator uses this functionality to acquire and account ID in order to delete it.

- Remove User

  Using the account ID retrieved from the 'List Users' operation, the administrator can delete the user account. This sends an email to the user through the built-in

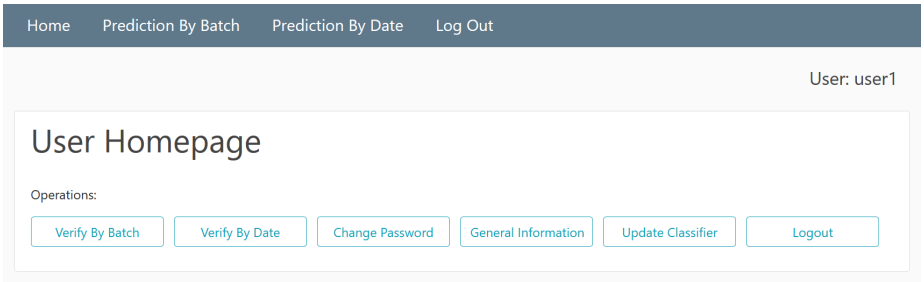


Figure 7.7: Operator homepage in the control interface.

library smtplib warning him that the account was deleted and he no longer has access to the application.

- See Logs

  This operation displays the Control module logs in the screen. This functionality is intended to help with troubleshoot tasks.

Standard User

The operator homepage is presented in Figure 7.7.

The operators accounts contain the following operations:

- Recover user credentials

  In case the operator forgets either username or password, they can be retrieved through email. The recovering process consists of sending an email using the *sendmail* function from the smtplib library to the operator containing the defined username and a new randomly generated password. The operator can then access the application with these credentials and should be advised to change the password.

- Change password

  Upon creating an account or forgetting the password, the operator is recommended to change the password for security reasons. Login is required to perform this operation. The user must insert the old password and then write the new password two times. The Users class is then used to query the database and find the entry related to the logged user and the password value is hashed using the *generate_password_hash* from the bcrypt library and stored.

- See general system information

  This operation allows the operator to verify if the system is active. It creates a TCP socket that attempts connection with the host in port 6001 and if the connection is successful, it means the Manager module is active. On the other hand, if the connection can not be established it means the system is shut down. Besides

this, the operator can see the details of the last image stored which are retrieved through the Images class. This is essential because because the Manager module can be active but the Image Acquisition module can be inactive (not recording images).

- Verify images by batch

  This operation will prompt the operator to select the classification label to analyze and then it loads a constant number of images to the screen. Since this project is based on only four videos with 53 detected images the constant value is small - the batch size is 8 bottles. The batch size is a small number because this is a prototype and the system will not be classifying images 8 hours a day. Note that when implementing the system to a production line that records a bottle each 3 seconds approximately, at the end of the working day (lets consider 8 hours) the number of bottles recorded is 9 600 (20 bottles per minute $\times$ 60 minutes $\times$ 8 hours).

  Each loaded image has a checkbox that the operator checks if the image was incorrectly predicted. Once he declares the batch as seen, he presses the 'Retrain' button at the end of the page which will load the next batch using the FIFO methodology and change the state of the checked images. The checkbox is named after the image ID and, when the 'Retrain' button is pressed, the system gets the name of the checkboxes activated and uses it to search for entries in the database using the class Images with the same name. In these entries, the state is changed.

- Verify images by date (and time)

  This operation differs from the last only in the image loading process which allows the operator to verify images recorded at a certain date and time without having to see the previous images in which he is not interested. The operation will display a form where the operator inserts the day, hour and the classification label and this information is then used alongside the Images class to load all images recorded within that hour to the screen.

  The operator selects the incorrectly predicted images similarly to explained in the previous operation. Once he finishes the selection, he confirms the operation and the images status are updated in the database.

- Update Classifier

  This operation queries the table images in the database searching for entries where the state i 'r'. The image signature of each image identified is obtained and sent to the Manager module which redirects it to the Classifier module. In the Classifier module, these images are partial fitted and the updated classifier replaces the old one.

In the ideal scenario, all images should be verified by a operator but, as previously mentioned, there could be a total of 9 600 images a day or even more depending on the company work schedule. The verify by batch operation was implemented in the application but is treated as a theoretical scenario. Figure 7.8 presents the database

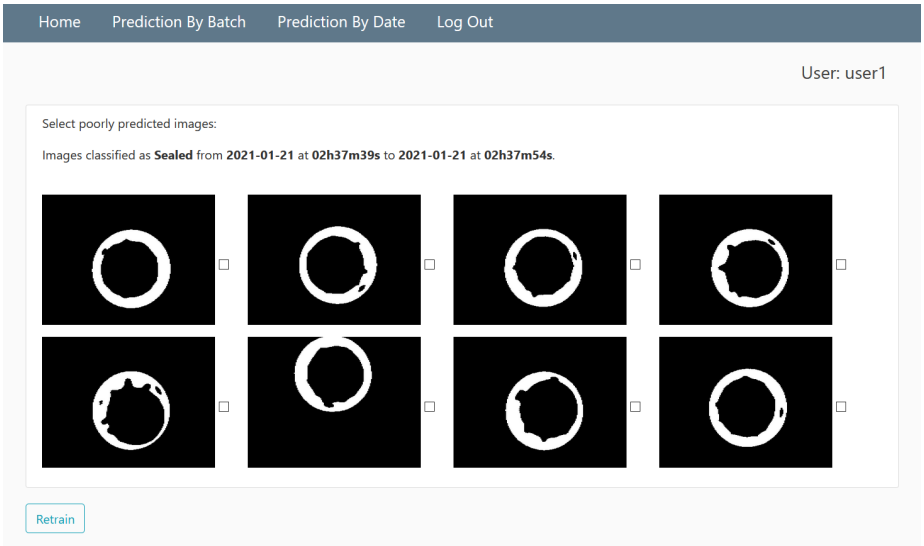


Figure 7.8: Operator's interface to select incorrectly classified seal images to be re-trained in order to improve the machine learning model performance.

seals analysis. Here the operator will select the checkbox of all incorrectly predicted images and then press the 'Retrain' button.

Later, if a problem — unsealed bottle not removed by this system — is identified, using a production number, the operator will be able to track the bottle and estimate the period when the bottle was recorded. The operator will then insert the estimated date and time in the control interface and the system searches for the incorrect prediction.

Chapter 8

Discussion

This chapter contains the discussion about the work developed. It emphasizes the necessity of the system, identifies and analyses the result obtained and presents the contributes of this work.

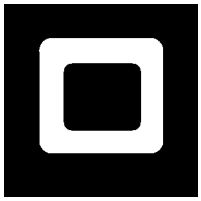
8.1 Work Relevance

Nowadays, assuring the quality of the product sold to the client is an essential element of any business. The same problem generally has multiple solutions on the market developed by different enterprises with different cultures and goals which creates something slightly unique but nonetheless solve the same problem. This is a healthy environment which fuels the advancement of technology and improves the customer experience. Since the buyer is able to choose between multiple products, selling a defective product will not only make the company lose a client but also create bad publicity which can lead to the loss of many more clients in a competitive market. This is specially important when the defective product can harm the user's health and the environment. The quality control of the sealing process of bottles which contain toxic substances is a clear example.

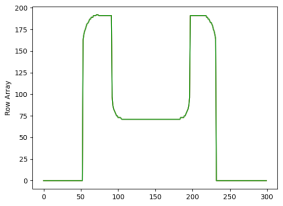
8.2 Advantages of the Model Proposed

The quality control system presented in this dissertation satisfies the requirements defined. It can be executed in a personal laptop which satisfies the requirements related to physical space and hardware. Nowadays, there are Raspberry Pi's with hardware characteristics good enough to support this system which would require even less physical space in the shop floor.

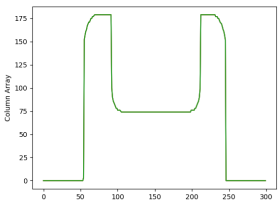
In this project, having a false negative represents a defectively sealed bottle containing a toxic substance which can be distributed to the market, if not detected by other means. The model achieved the goal of not having false negatives occurrences obtaining an *accuracy* of 98.7% and an *F1 score* of 96%.



(a) Seal

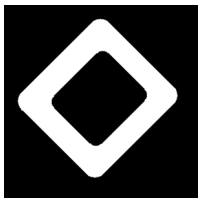


(b) Row Signature

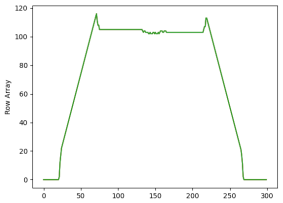


(c) Column Signature

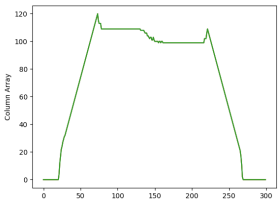
Figure 8.1: Image signature of a squared seal recorded horizontally.



(a) Seal



(b) Row Signature



(c) Column Signature

Figure 8.2: Image signature of a squared seal recorded vertically.

The systems works for seals of any shape and size but mixing multiple shapes of seals in the same conveyor belt can not be as effective as it will be explained in Section 8.3.

The system also gives the operator the possibility to work from outside the shop floor by accessing the web application. The web application requires no installation and can be accessed using a browser that supports HTML and the HTTP protocol.

The advantage is the machine learning model will be continuously learning and correcting its errors and it will most likely get better accuracy.

8.3 Limitations of the Model Proposed

In this particular case, the seals are circles which means the bottle position is irrelevant but, for example, if the seals were squared all bottles should be in the same position to prevent the image signature range from being too broad. Figure 8.1 and Figure 8.2 show the impact of the object position in the image signature in case the seal presents a squared geometry. Both images were manually drawn just to simulate the seal.

Using this method to predict the quality of different types of seals in the same production line is less viable because the larger the image signature range is, the higher the probability is for an unsealed object to have a signature similar to the type of signatures that should be accepted.

Have operators responsible for improving the machine learning model by selecting

images to be retrained can be a disadvantage. If the worker selects correctly classified images to be retrained the model, its performance will decrease.

8.4 Main Contributions

In this project an evolutionary quality control system was developed. This system collects the images, classifies and stores them but it still requires a operator to identify mistakes in order to improve.

This project used methods described in multiple works on the Chapter 3. An infrared camera and the HSV color model [5] were used to retrieve images of the hidden seal and clean the image in order to highlight the seal as detailed in Chapter 5. The image signature [3] is calculated to reduce the number of inputs of the artificial neural network [4][3] without losing performance. In this project, the artificial neural network consists of a multi-layer perceptron [13][14][15] implemented using the Python scikit-learn library. The system contains a database [16][18][21] to store an entry identifying each image classified. The classification can be verified from outside the shop floor environment [16] using a web application.

This project adapts this technologies to the case study which can be included in the group of works presented in Section 3.5. It creates an architecture to solve the detailed problem, described a machine learning model that can be used to classify seals independently of their shape and adds the possibility of correcting the system mistakes by partial fitting the incorrectly classified images causing an improvement on the model performance.

Our method proposed for solving the present problem was published in Journal of Imaging (`https://www.mdpi.com/journal/jimaging`) [25]. It presents in detail the image selection algorithm (Section 5) and the construction of the machine learning model (Section 6). The despite the small dataset, the model is effective and contains no false negatives. In our paper an early scheme of the architecture is also presented.

Chapter 9

Conclusion

This chapter contains a summary of the work done in this project, an analysis of the objectives and goals achieved and what could be done in the future to improve the system.

9.1 Summary

A quality control system to assess the sealing process of bottles which contains a toxic substance is developed in this project. Assigning an operator to visually verify the quality of the seal is not a solution because the seal is located below an opaque lid and can not be accessed without removing it. The seal is fixed using hot glue which raises the possibility of using infrared technology to obtain images of the seal.

When obtaining images of the seal with a thermal camera after the sealing process is completed, the glue distribution over the bottle opening is a good indicator to assess if the seal is correctly placed. An image containing a complete green circle corresponds to a perfectly sealed bottle and only these bottles should be sold.

During an 8 hours work-day, the infrared camera records up to 9 600 bottles. Assigning an operator to verifying each individual bottle can be done but due to the repetitiveness, machine learning tools are used to automate the task.

The infrared image obtained is converted to black and white and then cleaned – existing noise is removed – creating an image that only contains the seal. An array is created containing the sum of white pixels in each row. A second array is created by repeating this process to the columns. The two arrays obtained have a very similar and unique shape when it comes to perfectly sealed bottles – it assumes a U shape. When the seal is defective, the array shape changes and it is possible to note the imperfection. These arrays are fed to an artificial neural network which will predict the seal condition.

Despite the good machine learning model created, errors can still occur and, to assure they are identified, all predicted images are stored in the system and an entry linked to them is inserted in the database containing properties like the path to the image, the prediction result and others. An operator can remotely access a web application where the stored images are loaded. In this control interface the wrong classifications can be

identified and can be retrained to assure the model keeps learning if needed.

9.2 Final Considerations

The goals proposed for this project were achieved. A system capable of selecting images, identifying correctly sealed bottles with the possibility of validating the classifications via web browser and selecting wrong results to be corrected in order to improve the machine learning model was created. It respects the requirements defined in Chapter 2.

The prototype developed is not ready to be used in the industry because it lacks functionalities that will be detailed in the following section.

The system developed in this project despite solving the problem of the case study, should not be used in the industry because it is a prototype and does not contain the functionalities that will be detailed in the following section.

9.3 Future Work

Some functionalities could be added to simplify the use of the prototype. These functionalities are:

- Remotely starting the system from the web interface or at the very least create a Python script which will start all modules when executed.
- Restoring the images states to the original to correct mistakes such as correctly classified image identified as incorrectly classified by a user on the control interface.
- Develop a motion-based algorithm to detect bottles without glue. The current algorithm does not detect these type of bottles because it requires green color – caused by the existence of hot glue – in the infrared image. Despite the bottle without glue being the same color as the background, the circle can still be seen as presented in Section 5.2 and by using an edge detection algorithm such as OpenCV Canny Edge Detection it may be possibly to identify the bottle.

The image selection method described in this dissertation and the motion-based algorithm could be use to count the bottles recorded. When only the motion based method detects a bottle, the counters would have different values which means a bottle without glue passed through the camera visual range.

The following improvements are not included in the scope of the current project but would also be interesting to implement. First, the mechanical actuator responsible for removing unsealed bottles could also be programmed using a device that would link the Python software to the hardware such as Arduino for example.

Finally, it would be interesting to implement the system in a shop floor and evaluate its evolution after a period of time with daily use but this will require a security focused analysis to improve the system security before its implementation. Currently, a few routes can be accessed without permission when typing directly the URL. By exploiting

this bug, it is possible to retrieve sensitive information from the users table through a SQL injection.

Bibliography

[1] M. Asaduzzaman, M. Kerschbaumer, M. Bodner, N. Haman, and M. Scampicchio, "Short-wave near infrared spectroscopy for the quality control of milk," *Journal of Near Infrared Spectroscopy*, vol. 28, no. 1, pp. 3–9, 2020.

[2] M. Boiret and F. Chauchard, "Use of near-infrared spectroscopy and multipoint measurements for quality control of pharmaceutical drug products," *Analytical and Bioanalytical Chemistry*, vol. 409, p. 683–691, January 2017.

[3] R. Pontes, M. Mendes, J. T. Farinha, and J. Almeida, "Motor overheating monitoring using thermal images and artificial neural networks," in *18th International Symposium on Ambient Intelligence and Embedded Systems*, Coimbra Engineering Academy, September 2019.

[4] M. Elgargni, A. Al-Habaibeh, and A. Lotfi, "Cutting tool tracking and recognition based on infrared and visual imaging systems using principal component analysis (pca) and discrete wavelet transform (dwt) combined with neural networks," *The International Journal of Advanced Manufacturing Technology*, vol. 77, p. 1965–1978, April 2015.

[5] M. Haider, A. Doegar, and R. K. Verma, "Fault identification in electrical equipment using thermal image processing," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pp. 853–858, IEEE, September 2018.

[6] G. Zheng, X. Wu, Y. Hu, and X. Liu, "Object detection for low-resolution infrared image in land battlefield based on deep learning," in *2019 Chinese Control Conference (CCC)*, pp. 8649–8652, July 2019.

[7] J. Redmon and A. Farhadi, "Yolo: Real-time object detection."

[8] G. Dua and R. Mulaveesala, "Thermal wave imaging for non-destructive testing and evaluation of reinforced concrete structures," *Insight - Non-Destructive Testing and Condition Monitoring*, vol. 60, pp. 252–256, May 2018.

[9] A. Al-Habaibeh, F. Shi, N. Brown, D. Kerr, M. Jackson, and R. M. Parkin, "A novel approach for quality control system using sensor fusion of infrared and visual image processing for laser sealing of food containers," *Measurement Science and Technology*, vol. 15, pp. 1995–2000, August 2004.

[10] A. Al-Habaibeh and R. Parkin, "An autonomous low-cost infrared system for the on-line monitoring of manufacturing processes using novelty detection," *The International Journal of Advanced Manufacturing Technology*, vol. 22, p. 249–258, June 2003.

[11] K. D'huys, W. Saeys, and B. D. Ketelaere, "Active infrared thermography for seal contamination detection in heat-sealed food packaging," *Journal of Imaging*, vol. 2(4), p. 33, November 2016.

[12] Z. Shuangyang, "Fast inspection of food packing seals using machine vision," in *2010 International Conference on Digital Manufacturing Automation*, vol. 1, pp. 724–726, Dec 2010.

[13] F. Del Frate, F. Pacifici, G. Schiavon, and C. Solimini, "Use of neural networks for automatic classification from high-resolution images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 4, pp. 800–809, 2007.

[14] L. Parisi, "m-arcsinh: An efficient and reliable function for svm and mlp in scikit-learn," *arXiv preprint arXiv:2009.07530*, 2020.

[15] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu, "Handwritten bangla alphabet recognition using an MLP based classifier," *CoRR*, vol. abs/1203.0882, 2012.

[16] N. M. Dimitris Mourtzis, Ekaterini Vlachou and N. Xanthopoulos, "A cloud-based approach for maintenance of machine tools and equipment based on shop-floor monitoring," *Procedia CIRP*, vol. 41, pp. 655–660, 2016.

[17] D. M. Luc Bongaerts, László Monostori and B. Kádár, "Hierarchy in distributed shop floor control," *Computers in Industry*, vol. 43, pp. 123–137, October 2000.

[18] Y. J. Son and R. A. Wysk, "Automatic simulation model generation for simulation-based, real-time shop floor control," *Computers in Industry*, vol. 45, pp. 291–308, July 2001.

[19] P. Vithu and J. Moses, "Machine vision system for food grain quality evaluation: A review," *Trends in Food Science & Technology*, vol. 56, pp. 13–20, October 2016.

[20] J. Cheng, L. D. Xu, W. Chen, F. Tao, and C.-L. Lin, "Industrial iot in 5g environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, pp. 10–19, June 2018.

[21] A. Saez-Mas, J. P. G. Sabater, and J. M. Llorca, "Using 4-layer architecture to simulate product and information flows in manufacturing systems," *International Journal of Simulation Modelling*, May 2018.

[22] Y. Xu and Z. Jin, "Down-sampling face images and low-resolution face recognition," in *2008 3rd International Conference on Innovative Computing Information and Control*, pp. 392–392, 2008.

[23] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.

[24] D. C. L. . J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, p. 503–528, 1989.

[25] S. Cruz, A. Paulino, J. Duraes, and M. Mendes, "Real-time quality control of heat sealed bottles using thermal images and artificial neural network," *Journal of Imaging*, vol. 7, no. 2, 2021.