INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Detection of dish manufacturing defects using a deep learning-based approach

Afonso Luís Costa Barbosa da Silva

Master in Telecommunications and Computer Engineering

Supervisor:

Prof. Doctor Tomás Gomes da Silva Serpa Brandão, Assistant Professor,

ISCTE-IUL

Co-Supervisor:

Prof. Doctor João Carlos Amaro Ferreira, Assistant Professor,

ISCTE-IUL

October, 2020

TECNOLOGIAS
E ARQUITETURA

Detection of dish manufacturing defects using a deep learning-based approach

Afonso Luís Costa Barbosa da Silva

Master in Telecommunications and Computer Engineering

Supervisor:

Prof. Doctor Tomás Gomes da Silva Serpa Brandão, Assistant Professor,

ISCTE-IUL

Co-Supervisor:

Prof. Doctor João Carlos Amaro Ferreira, Assistant Professor,

ISCTE-IUL

October, 2020

Resumo

O controlo de qualidade é fundamental para assegurar o bom funcionamento de um processo industrial. Este trabalho propõe a utilização e adaptação de um algoritmo, baseado em aprendizagem profunda, como parte integrante de um sistema automático de controlo de qualidade numa fábrica de pratos de porcelana. Este sistema receberá imagens adquiridas em tempo real por câmaras fotográficas colocadas diretamente sobre a linha de produção. O algoritmo utilizado classificará os pratos presentes nas imagens como "defeituoso" ou "sem defeito". O objetivo do sistema será, portanto, a deteção de pratos defeituosos, fazendo com que menos pratos com defeito cheguem ao mercado, contribuindo assim para uma melhor reputação da fábrica.

Este sistema é baseado na aplicação de uma rede neuronal convolucional. Este tipo de redes requer um elevado número de dados para ser treinado de modo a conseguir realizar a classificação de imagens. Uma vez que a pandemia de COVID-19 se fez sentir em maior escala em Portugal na altura do desenvolvimento deste trabalho, foi impossível a obtenção de imagens provenientes da fábrica. Devido a este contratempo, os dados utilizados neste trabalho foram gerados artificialmente. Ao fornecer imagens completas de pratos ao algoritmo, o mesmo atingiu uma taxa de acerto da deteção de defeitos de 92,7% com o primeiro conjunto de dados e 91,9% com o segundo. Ao fornecer ao algoritmo segmentos de 100x100 pixéis da imagem original, o mesmo atingiu 91,6% de taxa de acerto, o que se traduziu numa taxa de acerto de 52,0% na classificação das imagens completas de pratos.

Palavras-chave: Controlo de qualidade, aprendizagem profunda, rede neuronal convolucional, classificação de imagem.

Abstract

Quality control is essential to ensure the smooth running of an industrial process. This work proposes to use and adapt a deep learning-based algorithm that will integrate an automatic quality control system at a porcelain dish factory. This system will receive images acquired in real time by high resolution cameras directly placed on production line. The algorithm proposed in this research work will classify the dishes presented in the images as "defective" or "without defect". Therefore, the objective of the system will be the detection of defective dishes, causing fewer defective dishes to reach the market, thus contributing to a better reputation of the factory.

This system is based on the application of an algorithm called Convolutional Neural Network. This algorithm requires a large amount of data to be trained and to perform the image classification. Since the COVID-19 pandemic was felt on a larger scale in Portugal at the time of the development of this research work, it was impossible to obtain data directly from the factory. Due to this setback, the data used in this work was artificially generated. By providing the complete images of dishes to the algorithm, it achieved a defect detection accuracy of 92.7% with the first dataset and 91.9%. with the second. When providing the algorithm 100x100 pixel segments of the original images, using the second created dataset, it reached 91.6% accuracy in the classification of these segments, which translated into a 52.0% accuracy rate in the classification of the complete dish images.

Keywords: Quality control, deep learning, convolutional neural network, image classification.

Acknowledgements

I would like to acknowledge the person who supported me the most in all stages of the development of this thesis. Renata, without your support and motivation, I would never have succeeded. I love you the most.

I would also like to thank my professors Tomás Brandão and João Ferreira for all the availability in times of pandemic, never making me feel alone or lost with this research work. To INOV INESC Inovação, for developing the script that generated realistic examples.

Finally, I would like to thank my family for all their support.

Contents

List of Figures

List of Tables

List of Acronyms

ML – Machine learning

AI – Artificial Intelligence

ANN – Artificial Neural Network

CNN – Convolutional Neural Network

ReLU – Rectified Linear Unit

GPU – Graphics Processing Unit

GB – Gigabyte

PCA – Principal Component Analysis

ILSVRC – ImageNet Large Scale Visual Recognition Challenge

Chapter 1. Introduction

1.1 Motivation

Quality control is manually performed in many factories around the world. Most industries allocate human resources whose function is to examine each produced piece in order to detect manufacturing defects. This procedure is expensive for a company and is susceptible to human error.

In factories that produce thousands of products per day, it is almost impossible for the personnel to keep up the rhythm without getting eyesight fatigue. That may result in misjudgments for some of the produced pieces, and therefore defective products may end up in the market, causing damage to the company reputation. Additionally, manual inspections may become a limiting factor in terms of speed of production if the company policy is to inspect all the products. A standard alternative policy is to inspect a small subset of the produced products in order to maintain the production rhythm. In such cases, the company will obtain statistical information about the percentage of defects that occur along their mass production lines, but many uninspected defective products will end up in the market.

Therefore, manual quality control may lead to degraded product shipping and consequent loss of potential profit. In order to minimize these losses, or at least get a better defect detection rate in mass production lines, companies are starting to rely on automated defect detection procedures.

1.2 Context

Quality assurance in industrial production is essential for the long-term success of the producing entity. Current automation levels in industrial production require quality inspection procedures with little or even without human intervention, in order to keep the production pace. For an automatic quality inspection to be considered effective, it should achieve the accuracy of the human level or more. To maintain competitiveness, entities with mostly automatic production lines seek to achieve quantity and quality with automation without compromising each other. For this purpose, confidence in automated quality control processes is of great relevance. It is

important to know *a priori* what types of defects can appear in products during their production. In addition, trust in one or more operators is required in order to ensure the effectiveness of automatic quality control. A high number of false positives can drastically reduce this confidence, dissipating any advantage in the use of automatic processes for these inspections.

Methodologies based on computer vision and deep learning are a real alternative to overcome these challenges, given their recent state-of-the-art results in image classification and detection tasks. Quality control based on machine learning is potentially cheaper to implement on a production line compared to traditional approaches, such as paying some employees to perform this control. Considering the availability of online open source environments for machine/deep learning algorithm development as well as cheap hardware in terms of cameras and computers, it is expectable that more and more companies start relying on these approaches [1].

The use of traditional machine learning approaches would require the identification and extraction of features that, in some way, are relevant for the detection of possible defects. Adapting a traditional machine learning system to other cases, such as detecting additional types of defects for which it had not been initially developed or applying it to the inspection of different products, would also require the identification and extraction of additional features. These feature identification and extraction processes could prove to be more difficult than using deep learning-based approaches, where feature identification and extraction processes are embedded in the system's training. Algorithms based on deep learning acquire a higher rate of effectiveness the more data is provided to them, which is a situation like a production line it is a win-win situation since it is possible to obtain a large dataset quickly and therefore a significant increase in the accuracy of the trained model.

Given the vast number of produced dishes by the factory in question, this work proposes the use of a deep learning-based algorithm capable of classifying dishes as defective or non-defective. This work was made possible thanks to the collaboration between ISCTE-IUL and INOV INESC Inovação.

1.3 Objectives

The main objective of this work is to adapt a deep learning-based algorithm capable of performing an automated classification process of dish images into two classes: “defective” or “non-defective”, allowing automated quality control inspections in a porcelain dish factory.

The algorithm performs this classification based on images that are collected along the production line. The porcelain dish factory managers are aware of two possible types of defects in the dishes produced on the mass production line. These are *stitches* and *cracks*, which will be detailed in Chapter 3. It is also possible that, at the end of the production line, a dish will have more than one defect. Nevertheless, and according to the factory managers themselves, manufactured dishes containing more than four defects are extremely rare.

Deep learning-based algorithms that are gaining popularity in image classification and recognition tasks are those based on convolutional neural networks (CNN). CNN's may present different architectures, which makes its computational cost differ from case to case. Architectures previously explored in other works showed great results in image classification and detection and are available in the form of online libraries. It is important to mention that training a CNN typically requires a larger amount of labelled data when compared with the use of traditional machine learning algorithms. In the context of this work, a significant amount of dish images, with and without defects, must be provided for the algorithm’s training, in order to allow adequate learning of the relevant characteristics of these two classes of dishes for correct classification. It is also necessary to identify the defects generated in the production line in order to label the images of the dataset.

Since CNN’s are widely used in image classification and recognition applications, and they obtain excellent accuracy results in these types of problems, it was decided to explore this class of machine learning algorithms in this thesis.

1.4 Work Contributions

The COVID-19 pandemic conditioned the operation of several companies during 2020. At the time of this project, the factory that would supply the image dataset shut down its production lines, invalidating the use of image data acquired in real conditions. Since the main contribution

of this project was to use deep learning to detect defects in dishes, it was necessary to setup an image dataset that would allow training and testing an image classifier. This setup avoided an indefinite project stall, allowing to make progress even with this setback.

Since a dataset containing images of defective industrial dishes was not available online, the adopted solution was to create an artificial dataset by downloading a set of images of dishes that were found online and manually inserting the defects into those images. Posteriorly, data augmentation techniques were used to increase the number of samples and thus, a relatively large dataset was created.

This dataset had certain limitations since the defects were manually inserted and did not effectively represent the actual defects produced at the factory in question. Several experiments were performed since it was the only source of data available until mid-June. Based on these experiments, the architecture of the adapted algorithm that obtained the best results in terms of accuracy and processing time was chosen for further developments.

During the month of June, the factory provided a few dishes directly from the production line. It was decided in INOV that, with these few dishes available, a possible solution to progress with the project would be to develop an image generator that uses the texture of these dishes in order to generate a large quantity of dish images identical to real ones. Not only would the texture be similar, but the defects inserted into these images would be more realistic since at that point in time, the INOV developers had already been able to observe defects in real dishes.

When developing and subsequently running this image generator, a new dataset with a large number of realistic images was created. However, one of the factors that made difficult to pre-process the data related to this dataset was the resolution of the generated images. This image generator aimed to create a more realistic dataset, and the images generated by it had a similar size to the high-resolution images that would be captured by the cameras placed at the production line. Generally, the input layer of CNN admits images that have much smaller dimensions. This new dataset was used in the training and validation of the adapted algorithm. Once the results of accuracy, precision, recall and F1-score were obtained, it was possible to perform experiments in order to try to improve the previous mention metrics and reduce false positive results, since these are the ones that most harm the firm's profits.

Two possibilities of classifying these images were tested. The first was to classify the entire image of the dish resized to a low resolution. These low-resolution images result from a

subsampling of that acquired on the production line. The second was to individually classify pieces of the image and then classify the entire image of the dish based on the results obtained in classifying these pieces.

In the first possibility, resizing the entire image of the dish causes loss of image quality, which can harm the classification performed by the algorithm. To check whether it would be possible to obtain higher results in accuracy, precision, recall and F1-score, the second possibility consisted in dividing the initial image into smaller segments, using image crops without any resizing operation (and therefore keeping the original image quality). Subsequently, these segments of the image are individually classified and based on their classifications, the entire dish image is classified, in order to compare the two possibilities tested.

Regarding the algorithm, simple CNN architectures with low computational cost were tested, as well as transfer learning architectures. A simple CNN architecture is more adaptable to the specific problem to be solved since all parameters are trainable. Transfer learning architectures are more generic, which does not allow a full adaptation to the problem in question, since only the last layers of the classifier (fully connected layers) are trained. In addition, these architectures are computationally heavier.

In these comparisons, the key factors are the data processing time – which is the time it takes to train the algorithm with the dataset – as well as the accuracy of the algorithm.

1.5 Document Structure

This thesis is organized according to six chapters, whose contents are as follows:

- In *Chapter 1* is where the context and motivation for carrying out this research work are exposed. In addition, the objectives are also defined in this chapter.
- *Chapter 2* presents an introduction to the main machine learning concepts and algorithms used in the scope of the thesis. Additionally, this chapter also includes a brief state-of-the-art that presents previous works related to the use of machine learning for image-based industrial defects detection;
- *Chapter 3* contains the proposed solution for the automatic quality control system. The difficulties and setbacks felt in carrying out this research work are also explained in

detail. Due to delays caused by the COVID-19 pandemic, two artificial datasets used to obtain results in this work were created. As stated earlier, during a large part of this dissertation the only source of data was the dataset where the defects were manually inserted. The second dataset was created since there was a need to represent the data in a more realistic way. As soon as the samples of the production line were received, the creation of this dataset started with the goal of obtaining results that were closer to the possible values obtained when using data from the factory;

- *Chapter 4* illustrates the experiments and comparisons performed on the dataset with the manually inserted defects in order to find out which architecture leads to the best results for this binary classification. These tests and comparisons were carried out during the lockdown time, in order to allow the research work to progress;
- In *Chapter 5*, the results obtained using the realistic dataset are displayed. To create this realistic dataset, a script developed in INOV was used. This script works as an image generator and generates images of dishes with or without defect. These images are based on the real samples received from the factory. This chapter also explores and discusses the advantages/disadvantages of classifying segments of the original instead of classifying the image as a whole;
- *Chapter 6* presents the main conclusions of this work, and considerations for future work are also described.

Chapter 2. Literature Review

This chapter aims to provide a better understanding of the technologies used for the development of an automated defect detection system based on image classification. Later, it also provides a literature overview of some work related to the main goal of this thesis.

2.1 Basic Concepts and Definitions

2.1.1 Machine learning

The ability to learn is one of the main characteristics of intelligence, making it crucial for both human cognitive development and artificial intelligence (AI) [13].

Machine learning (ML) in the field of AI that can be considered as the science that allows computers to act without being explicitly programmed to do so. In machine learning, the algorithms are used to provide and select the necessary information for the machine to detect certain patterns or similarities on data in order to perform correct predictions. This whole process typically requires a large amount of data for training.

2.1.1.1 Types of learning

According to Stephen Marsland [17], machine learning algorithms can be classified in the following categories:

- *Supervised Learning* – A large dataset with the corresponding correct answers, called a training set, is provided. Based on this set, the algorithm generalizes a response in order to get the correct results from predictions performed on new data. Generally, there is also the testing set, where new data is provided in order to evaluate the prediction. This method is also called *Learning from exemplars*;

- *Unsupervised Learning* – Also known as Density Estimation, in unsupervised learning, the correct answers are not provided with the training set. It is up to the algorithm to identify similarities in the data and group them according to those;

- *Reinforcement learning* – T his learning method trains algorithms based on a penalty and reward system. A reinforcement learning algorithm learns based on the rewards and penalties obtained through the actions it takes. For instance, if the agent (who uses the algorithm) gets a penalty, he realizes that that action should not be taken in the future;

- *Evolutionary learning* – Biological evolution can be portrayed as a learning process. Biological organisms adapt from generation to generation in order to obtain more and better chances of survival. This is what these evolutionary learning methods try to do. Each attempt can be considered as a generation. From generation to generation, there is a progression towards the objective. Typically, later generations tend to get better results than previous generations, whether in image classification or game completion.

2.1.1.2 Artificial Neural Networks

Artificial neural networks (ANN) are computing systems that were inspired and structured based on biological neuronal networks found in the brains of humans and animals [14]. Besides, these ANN's are one of the most popular methods of supervised learning.

The brain consists of a highly connected network of neurons that communicate with each other through electrical pulses. These electrical pulses are transmitted from neuron to neuron through the dendrites and captured by the axon terminal. Between these two elements is the axon, responsible for retransmitting (or not) the electric pulses that the neuron receives, and for selecting the next neuron where the pulse should go afterwards. Therefore, the millions of neurons present in the human brain are all connected. This is called a neural network [15].

An artificial neural network consists of several units grouped in layers. The input layer is the initial layer where data enters the network. The output layer is the last layer of the ANN that returns the results computed by the system. Between these two are the hidden layers. Depending on the complexity of the problem to be solved, there is no definite number indicating how many hidden layers should exist per ANN.

Each unit in these layers is called a neuron. Each neuron in a certain layer is typically connected to all the neurons in the next layer. The links between layers have a specific weight associated. This weight is simply a number (positive or negative, integer or decimal) that represents the value of the information passing through the link. Each neuron performs a weighted sum of the information arriving at all input links and adds a term called bias (an offset value) to it. The result of this sum is received by an activation function which, depending on the result, activates or not the neuron. These activation functions are intended to make the network more robust, allowing it not to be susceptible to slight variations.

A widely used activation function is the sigmoid function, which transforms the values it receives into values between 0 and 1 through the following function: $\theta(x) = \frac{1}{1+e^{-x}}$ [16]. This function is often used because it has limits, it is differentiable and, for all real input values, has a non-negative derivative, which allows an easier and more efficient computation both for training and for testing. Figure 2.1 shows an example of a simple Artificial Neural Network architecture.

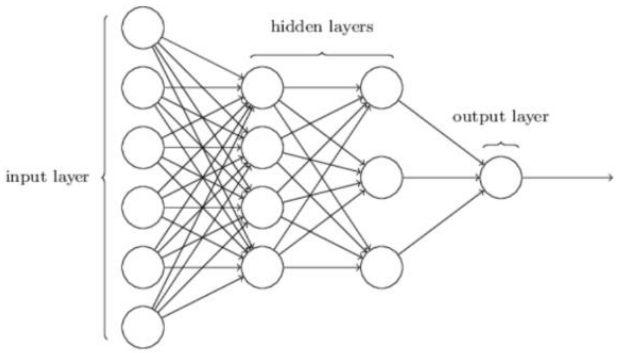

Figure 2.1 – Artificial Neural Network [2].

2.1.1.3 Back propagation

Back Propagation is a training method that was first published in the 1970s [18]. This method was later used to optimize artificial neural networks and, more recently, used in Deep Learning. Back Propagation is a process that propagates the total loss of the system back onto the neuronal network. This allows evaluating the neural link contributions to the overall loss, updating its weights in order to reduce its value.

2.1.2 Deep learning

Deep learning is a subset of machine learning. This subset can be interpreted as a set of machine learning techniques that aim to transmit knowledge to computers so that they reach conclusions that would be normal for humans but on a large scale.

2.1.2.1 Convolutional Neural Networks

There is a type of neural network called convolutional neuronal network (CNN) that stands out for its efficiency in solving image detection and classification problems, adaptability, as well as its ease of developing.

CNN's are inspired by the visual cortex of the human brain and have been widely applied in image recognition and object detection [10]. A CNN used to classify images is intended to accept an image as input and return the class to which the image belongs as output. This process is inherent in humans, and it even becomes difficult to explain how we know we are looking at a dog when we see a picture of a dog we have never seen before in our life. We can say that it is through empirical knowledge, but a CNN would "look" at this image in a very different way.

CNN "sees" the displayed image as a matrix of pixels. Similarly to a regular neural network, CNN is also organized into layers, neurons and weights. What diverges in a CNN structure is that this network seeks to make the input image increasingly abstract as it progresses through the network. Each neuron at each layer receives a fraction of the image and aims to treat it according to the type of layer it is in and according to the parameters applied to that specific layer. Not always a neuron has a direct link to all the next layer's neurons. Thanks to open source platforms such as TensorFlow (Google open source platform) or Pytorch (Facebook open source platform), CNN's are relatively easy to program or modify.

CNN's have different types of layers, which can be organized according to the following categories:

- *Convolutional Layer* – This type of layer applies a convolution kernel (usually with quadratic graduation) to its input data matrix. The kernel slides along with the entire matrix and, at each point of it, produces an internal product between its values and the

values presented in the image, returning a single output, as explained Figure 2.2. The first layer in CNN is always a convolutional layer.

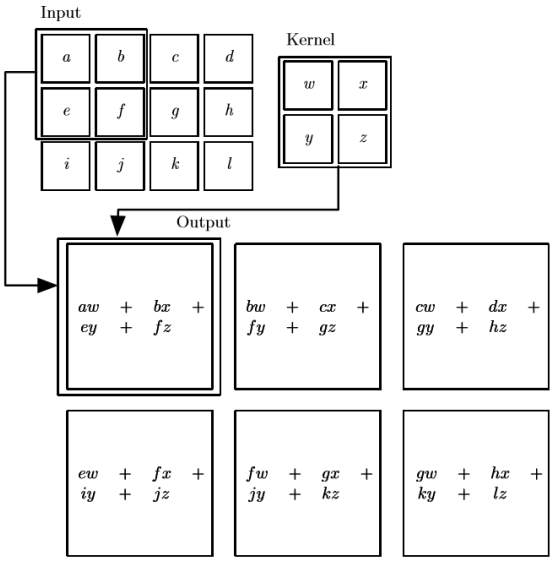

Figure 2.2 – Kernel application [3].

When computing this inner product, it is necessary to use padding. Padding consists of adding 0's around the image so that the filter can effectively apply the inner product to all pixels in the matrix. An example is shown in Figure 2.3.

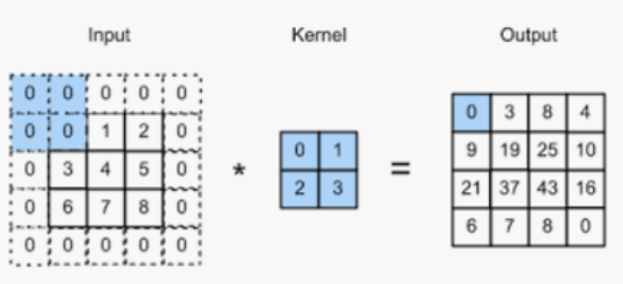

Figure 2.3 – Padding example [4].

It is also important to mention stride. Stride is a kernel parameter that indicates the number of units in which the kernel moves through the matrix. For the value (1, 1), the kernel moves one pixel horizontally and one pixel vertically each time. Changing this field influences how the filter is applied as well as the resulting image size after this layer.

- *Activation Layer* – Activation layer (or nonlinear layer) is placed after each convolutional layer, having an activation function that assigns nonlinear properties to the input matrix resulting from the previous convolutional layer. The choice of the activation function to be used is usually configurable when developing a CNN.

  One of the most used activation functions is the Rectified Linear Unit (ReLU). This function changes the value of the negative pixels to 0 and keeps positive values. Being a nonlinear function, but with many properties of a linear function, this function makes training models easier to optimize [3].

- *Pooling Layer* – It aims to reduce the number of pixels in the matrix, making it more abstract. This reduces the computational time to process it. Like the convolutional layer, it uses a kernel that slides along the matrix. To shrink the image, maximum pooling, average pooling or even minimum pooling can be used. Maximum pooling selects the largest of the values covered by the filter dimension in each portion of the matrix. Minimum pooling makes the same selection but with the smallest of the covered values. The average pooling returns the average of the values covered by the filter. A brief example of these three types of pooling is demonstrated in Figure 2.4.

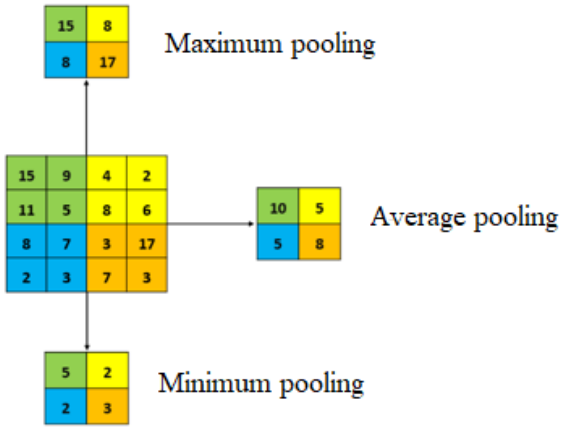

Figure 2.4 – Pooling demonstration, adapted [5].

- *Fully Connected Layer* – This type of layers is associated with the final decision about the class to which the initial image presented as input belongs. It has this designation because all the neurons in this layer are linked to all the neurons in the previous layer,

as is the case in all the layers of regular artificial neural networks mentioned above. This layer is represented by a vector with the number of positions equal to the number of neurons in the layer. The resulting vector contains at each position a percentage, which indicates the probability of the input image belongs to the class represented at that same position. The class with the highest percentage is the class predicted for the input image. The SoftMax activation function is often used for this distribution of values since it expresses the probability of an image belonging to a given class, which is a clearer concept both for the network developer and generic people unrelated to the algorithm development.

2.1.2.2 Overfitting

Overfitting occurs when an algorithm fits the training data too well, performing poorly on the classification of new samples. The network learns specific details of the training set images and is unable to correctly classify new images that do not contain those specific details. Figure 2.5 shows an overfitting illustration.

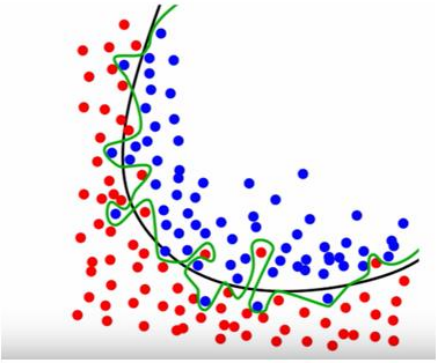

Figure 2.5 – Overfitting illustration [6].

The black line represents the ideal boundary for separation of two distinct classes. The green line represents an example of overfitting, where the classes are separated to the smallest detail.

2.1.2.3 Data Augmentation

Data augmentation is a method that aims to increase the amount of data used in CNN training and avoid overfitting. This method consists of handling the data before applying it to the network in order to increase the size of the training set since new examples are created based on the existing ones. Additionally, this method allows to highlight important details and normalize secondary details. Cropping, shifting or rotating are some of the techniques that are commonly used.

2.1.2.4 Dropout

Dropout consists of "turning off" some neurons from the network during training. The neurons that are turned off do not contribute to the training of the network and the consequent back propagation. Thus, whenever an image enters the network as input, CNN has a different architecture, but all these architectures share the same weights in the links. This technique reduces the complex adaptations that neurons create with each other since one neuron cannot rely on the presence of another neuron, as it may or may not be turned off. Therefore, each neuron is forced to learn more robust features that will be useful for classifying the image with another set of neurons other than the set that was previously required. Figure 2.6 shows a dropout example in a neural network.

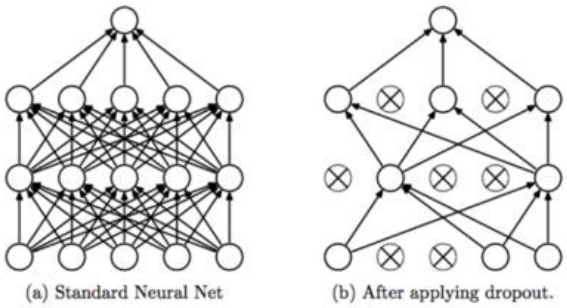

Figure 2.6 – Dropout example, crossed units have been dropped [7].

2.1.2.5 Skip Connections

The skip connections method consists of dividing the network into modules, called residual blocks. Instead of entering one of these blocks, the data is added to the output of the module, which allows greater data flow within the network and a decrease in overfitting and in execution time. As demonstrated in Figure 2.7, this method can assume different schemes.

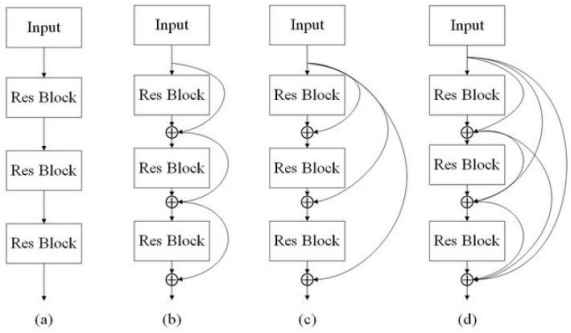

Figure 2.7 – Different skip connection schemes [8].

2.1.2.6 Transfer Learning

Building a CNN from scratch is a time-consuming process due to the complexity of the training procedures, as well as quite expensive in terms of computational memory. Given the fact that there are several open sources platforms with available CNN's with practical examples and architectures, researchers try to explore the hypothesis of reuse the knowledge acquired by some networks and modifying the final output in order to match their own classification objectives. Transfer Learning is motivated by these practices and attempts to improve the traditional method of machine learning by accessing knowledge of one or more tasks from the original network and using it to improve the learning of the new network [19].

Knowledge transfer techniques represent the progress of making machine learning as efficient as human learning [20].

The procedure is to use the same initial layers with the same weights. Then, replace the last layers (fully connected layers) with others that have the desired classification method. There are three comparison measures that allow evaluating if transfer learning is really being effective.

The first comparison measure is based on comparing the initial performance achieved using only the transferred knowledge with the initial performance of the network without training. The second measure is done by comparing the time the network takes to learn the task for which it was designed using only the transferred knowledge with the time it would take to train it from scratch. Finally, the third measure consists in comparing the results of the network classification with the transferred knowledge and the trained knowledge. If any of these measures indicates a better network performance that was not learned by transfer, a negative knowledge transfer has been performed. One of the main challenges of transfer learning is to develop transfer methods that do not lead to these negative transfers, producing positive transfers between appropriately related tasks.

2.2 Related Work

This section is divided into two parts. The first one contains the state of the art in convolutional neural networks architectures. The second part addresses to the application of machine learning to manufacturing defect identification and classification problems.

2.2.1 Main CNN Architectures

Recently, some CNN architectures have been developed with exceptional effectiveness in image classification. These architectures have a certain number of layers. However, it is possible to create a CNN combining as many layers as we like. It is important to remember that networks with more layers are deeper and more complex, and logically, more time, memory and computational performance will be required.

LeNet5 is the first relevant model for CNN's technological progress, given its complexity and effectiveness. The model was suggested in [9]. This model is characterized by seven layers. The first six layers use the Tanh activation function. The Tanh activation function maps pre-activation to itself by generating values from -∞ to ∞. This function is used because, besides being easy to implement, it implements a wide form of nonlinear regression. The last layer is a fully connected layer with ten neurons, which uses the SoftMax function to return a vector with ten positions as output in order to classify the initial image. The purpose of this model was to

recognize which digit was drawn by hand in the initial image, with each position of the output vector corresponding to a digit from 0 to 9.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been around since 2010 and aims to evaluate algorithms for object detection and large-scale image classification. The ImageNet database has millions of images from various categories, and every year a portion of this database is used for this challenge. The winner of the challenge is determined based on the top 5 error percentages of the algorithm used. The one with the lowest error percentages wins. In 2010 and 2011, having the lowest error rate, around 25% was considered a great rating, but in 2012 everything changed due to the AlexNet model, which achieved a 15% error rate.

The AlexNet model [10] is a CNN composed of eight layers, five convolutional layers and three fully connected layers. The first convolutional layer receives 224x224x3 RGB images. The second layer performs max pooling. The third, fourth and fifth layers are linked together without any kind of pooling. The first two fully connected layers have 4096 neurons each and the last one outputs using SoftMax in a 1000 position vector relative to the 1000 image classes that were being used in the classification challenge. This model is quite innovative, not so much because of the choices of layers, but for three other reasons. The first is the use of the ReLU activation function, which has substantially improved the computational speed, which caused the disuse of the Tanh function. The use of multiple GPUs was also crucial to the success of this model [21]. In 2012, top GPUs contained 3GB of memory (currently this value would be relatively low). This model allowed one GPU to be used for one half of the training set and another GPU for the other half, which not only allows the training model to be larger but also allows the network to train faster. The other significant reason for the decrease of the error percentage was overlapping pooling, which grants the stride value of a kernel that applies to pool a larger value than kernel's height and width. This allowed improvements between 0.3 and 0.4%. This architecture has about 60 million parameters to train and, at the point of publication, the authors pointed out that their architecture was “one of the largest convolutional neural networks to date on the subsets of ImageNet” [10].

Due to the complexity of the network, to avoid overfitting, it was necessary to use data augmentation techniques such as horizontal flips and Principal Component Analysis (PCA) in RGB images, in order to increase their relevance. Dropout was also used to train the network.

In 2014, both the first and second challenging approaches of the ILSVRC achieved significant improvements in comparison with the previous years, so it is important to mention both.

The runner up was the VGG architecture proposed in [11], which achieved an error percentage of 7.8%. There are two versions of this architecture, VGG16 and VGG19 where 16 and 19 represent the number of layers in the network. Similarly to the AlexNet model, VGG uses several convolutional layers with 3x3 kernels, as shown in Figure 2.8. The main difference between these two is that this network stacked more layers onto AlexNet but used smaller size filters. To reduce the initial image size, max pooling is again used, and the last 3 layers are also fully connected layers, the first two having 4096 neurons each, and the last one having a thousand neurons and using SoftMax to organize and return the prediction. This architecture was trained on four different GPUs for two to three weeks. This last sentence demonstrates one of the main disadvantages of this architecture, which is the speed to train, due to the 138 million parameters. However, it is one of the most widely used architectures for image classification nowadays.

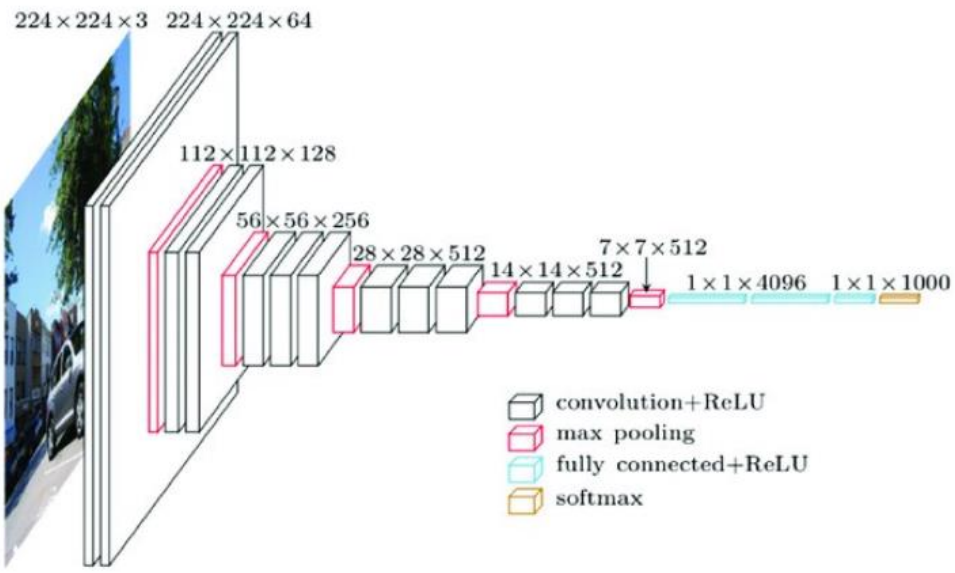

Figure 2.8 – VGG Architecture [11].

Despite the astonishing result obtained by the VGG architecture, the first place in the 2014 ILSVRC was won by GoogleNet architecture, also known as InceptionV1 [12].

Later, more advanced versions of this architecture would be developed. GoogleNet was developed by Google. The percentage error obtained by this architecture was 6.7%, which is very close to human effectiveness in an image classification challenge. Some of the methods used to achieve these results were the various convolutional layers with 1x1, 3x3 and 5x5 kernels where it was up to the network to decide which kernel combination was most efficient for handling a specific input. This method dramatically decreases the number of operations performed and increases the depth and width of the images. Another method that had an impact on the error percentage of this architecture was the use of global average pooling at the end of the network instead of using Fully Connected Layers. This change has improved the error percentage by 0.6%. This architecture has twenty-two layers and 4 million architecture parameters, which reduces the computational resources required compared to the VGG's architectures.

The following year, 2015, the ILSVRC was won by the ResNet architecture [22]. This was surprising due to the percentage of error obtained by this architecture, 3.6%. The key to success is a 152-layer architecture, thus, a much deeper network. Thanks to the skip connections method it was possible to train this network with even less complexity compared to the VGG architecture. This architecture has 26 million parameters, and it was innovative because it made it possible to use more layers without compromising model's generalization power.

2.2.2 Product Defect Detection

In this thesis, image classification is relevant because the main goal of the thesis is to classify dishes as "with defect" or "without defect". The detection and localization of defects are important in most of the industrial and non-industrial areas.

The work developed in [23] had the goal of detecting scratches on car paints based on images. The authors used two CNN's. The first CNN used was meant to detect and extract only the car region from the original image. This CNN obtained an accuracy of 98.3% in the extraction of the vehicle. The provided database for training this architecture consisted of 380 images with a resolution of 2340x4160 pixels. Three hundred twenty images were used for training, and 60 were used for testing. Figure 2.9 shows the architecture and some details of this CNN.

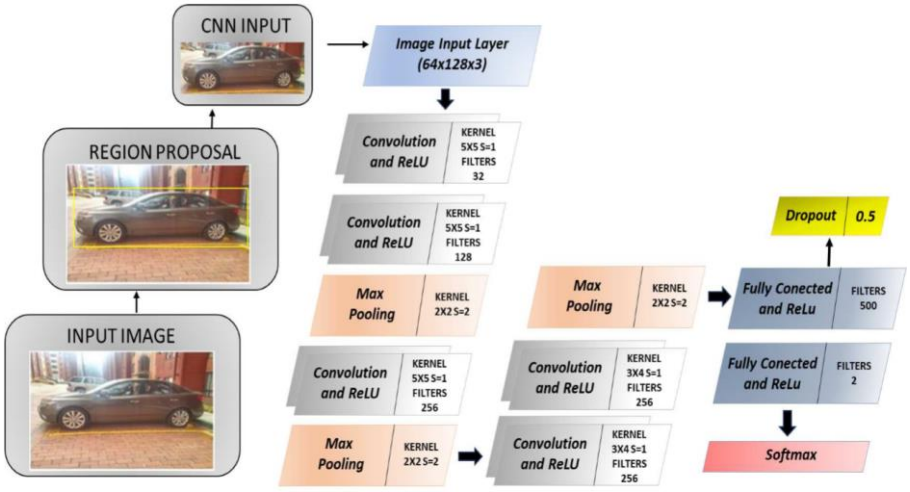

Figure 2.9 – Architecture of the first CNN used, where S refers to the Stride [23].

Then, the authors used the output of the first CNN as an input of the second CNN architecture. This CNN divides the resulting image from the previous one into multiple sections in order to detect the smaller details. The dataset provided to this architecture consisted of 15510 images, where 6190 of them presented painting scratches. On both categories (with and without scratches), 80% of the images were used for training, and the remaining 20% were used for testing. The result was accuracy of 96.89% in the detection of scratches in the input car region image.

In other work, the authors of [24] propose the use of a 10-layer CNN to detect and classify different types of coffee beans defects. Data acquisition was performed by placing the coffee beans on a white paper sheet before the photo was taken. Figure 2.10 shows examples of defected beans and an example of a normal bean.

Coffee beans had five different types of defects that were manually labelled by experts. The ratio between train, validation and test was about 10:2:1. The highest classification accuracy obtained was 98.8% for black beans. The lowest classification accuracy obtained was 67.5% for broken beans.

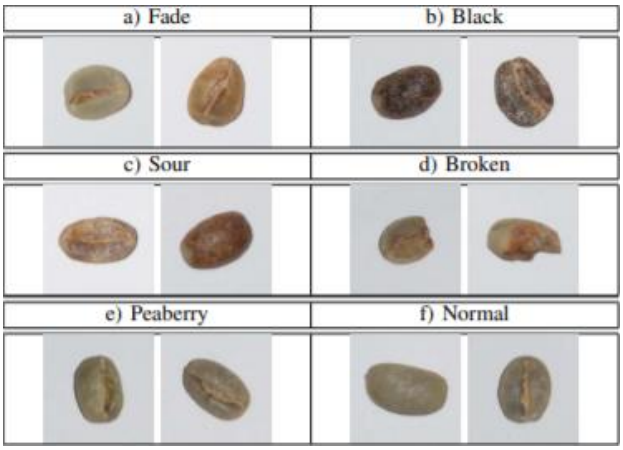

Figure 2.10 – Examples of defected beans, [24].

In [25], a pre-trained CNN architecture was adapted in order to detect two types of defects in LED chips. The adapted architecture was proposed in [26]. The authors used two datasets containing normal chips and defective chips with two types of defects, including line blemishes and scratch marks. The dataset size was increased by data augmentation techniques such as rotation, flipping, shifting, noising and blurring the original images collected. The proportion of training and testing images in both pieces of training were 4:1. The dataset consisted of 30000 images.

To detect and classify defects on metallic surfaces, another CNN architecture was suggested in [27]. The original images were captured by an industrial microscope. In order to train a suitable network, the authors used data augmentation techniques such as rotation, translation, zoom, etc. These operations increased the size of the training and testing sets.

In [28] and [29], the authors used CNN's to detect pavement cracks, both using TensorFlow. While [28] developed a 10-layer CNN architecture, [29] tested different structures in order to find the most accurate. The work presented in [29] combines CNN to detect the cracks and PCA to classify them as longitudinal, transverse and alligator cracks. The authors obtained correct classified rate percentages with and without the use of PCA and compared the results. This work obtained an accuracy of around 97% in detecting longitudinal and transverse cracks.

The authors of [30] and [31] reported high accuracy results on detecting defects in train railways. In [30], an architecture based on the VGG16 model was used, while in [31] a different approach was followed by using a three-stage architecture in which each stage had a different CNN model. [30] tried to apply different techniques and parameters to their architecture by generating seven different models. The results obtained show that these techniques can affect

the performance of the algorithm. The work proposed in [31] is more focused on the catenary support of the railways. The 3-stage system can be summarized in Figure 2.11.

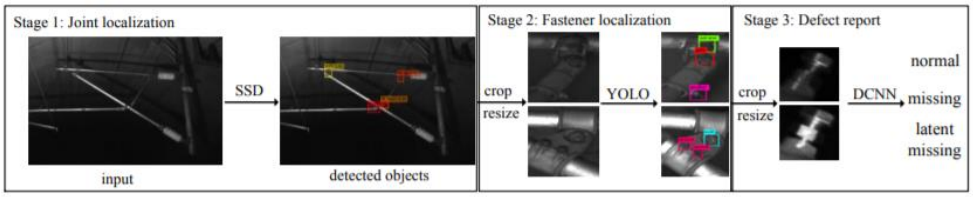

Figure 2.11 – System summary [31].

Stage 1 CNN architecture is based on a VGG model. Stage 2 CNN is based on a ResNet model. Stage 3 CNN has seven layers and classifies the input images has normal, missing, or latent missing. SSD (which stands for *single-shot multi-box detector*) and YOLO (which stands for *you only look once*) are deep learning-based image object detection methods.

In [32], different machine learning algorithms such as Logistic Regression, Linear Discriminant Analysis, k-Nearest Neighbors, CART Decision trees, Naïve Bayes, Support Vector Machines, Random Forest and CNN were used in order to detect four types of defects in porcelain products. Their results show that CNN was the algorithm with the best accuracy, reaching a mean of 89% defect detection accuracy. The CNN architecture used was quite simple, with only 3 layers. An overview of the proposed system can be observed in Figure 2.12.

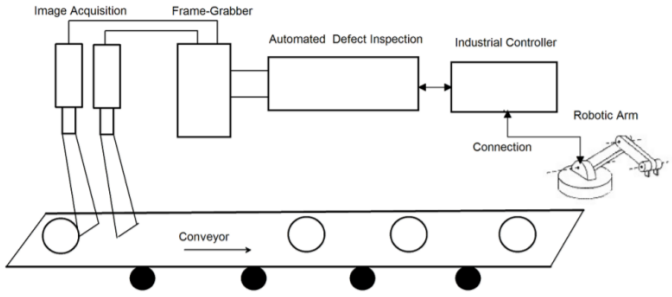

Figure 2.12 – Porcelain dish defect detection system proposed in [32].

Chapter 3. Proposed System

This chapter starts by presenting a brief description of the factory's current quality control procedures. Subsequently, the defined requirements of the project are presented. The adapted deep learning-based image classification algorithm is also detailed in this Chapter, as well as the built image datasets for training and testing it. The aim of the present work was to be used in the production lines of a Vista Alegre group's porcelain factory in Aveiro, which is the leading supplier of IKEA brand dishes in Portugal.

3.1 Quality Control Scheme

Currently, Vista Alegre group's porcelain factory in Aveiro daily produces four to five thousand dishes. Two workers are assigned to quality control posts. Their function is to detect and to remove the dishes that present one or more defects at the end of the production line. This quality control scheme cannot be done throughout its production due to limiting factors such as the temperature of the dishes or the manufacturing line speed. The workers assigned to the quality control post do not have the capacity to inspect all produced products, mainly due to the following reasons:

- *High number of produced dishes* – each employee would have to inspect more than two thousand dishes per day, and even by doubling the number of employees assigned to this post, the number of dishes inspected by each worker would still be high;

- *Eyesight fatigue and lack of focus* – the defects that are sometimes found in the dishes are minimal and can easily go unnoticed. It is also very hard to keep the required level of concentration for inspecting hundreds of dishes during long periods of time. Misjudgments can, therefore occur frequently.

The current quality control at the factory is carried out using an estimate for the defective dish production ratio. Form the 4-5 thousand dishes produced per day, only 200 samples are inspected for quality control. Based on the manual quality control inspections, the factory managers estimate that about 10% of the produced dishes contain some type of defect. Most of the defective dishes that are sent to the stores are not sold, which may potentially harm the

company’s reputation and, consequently, its profit. From the few possible types of defects that may occur during the production of a dish, the main types of defects identified by the factory managers are the *stitches* and the *cracks*, illustrated in Figure 3.1.

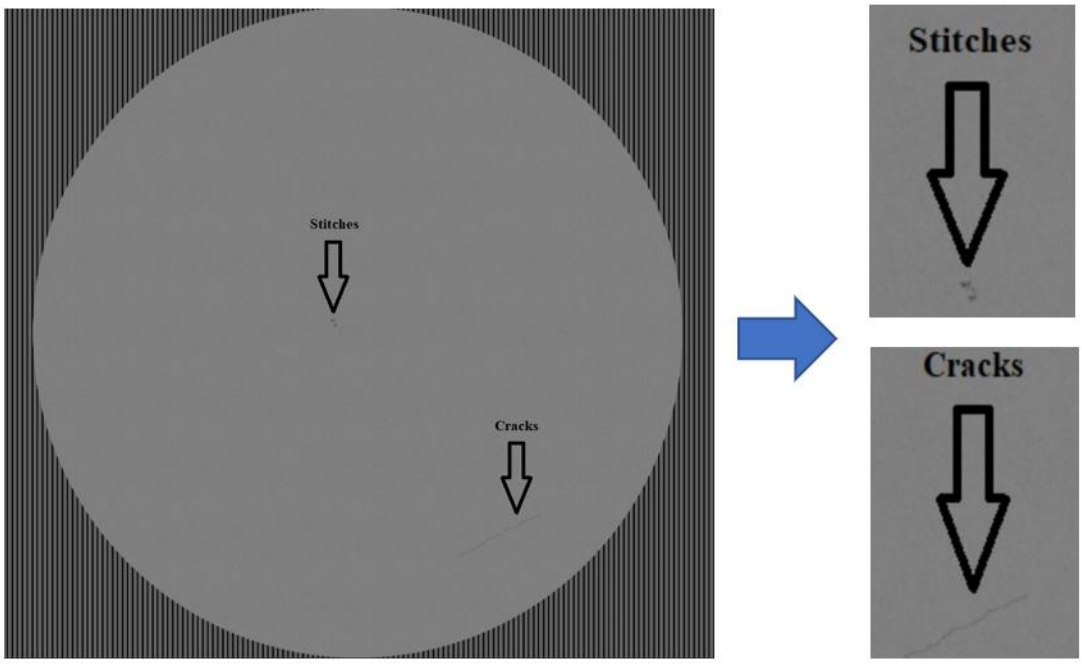

Figure 3.1 – Most Common Dish Production Defects (Simulated).

Based on conversations with the factory managers, the set of requirements necessary for this project’s realization are as follows:

- Define a set of high-resolution cameras capable of capturing the dish defects with enough detail. The cameras should also be sufficiently fast in order to capture sharp images of the dishes moving in the manufacturing line;

- Installation of cameras in different parts of the production line, in cooperation with the factory's workers. After installation, the cameras should acquire a large set of images, which will be subsequently labelled in order to setup a dataset for training and testing the machine learning algorithms;

- Development/Adaptation of a deep learning-based algorithm with the ability to perform binary classification (defective / non-defective) on dish images acquired on the production line;

- Evaluation of the algorithm's classification results and comparing these with the results of the manual inspection in the quality control.

The proposed automated quality control system for the detection of dish defects will follow the diagram represented in Figure 3.2.

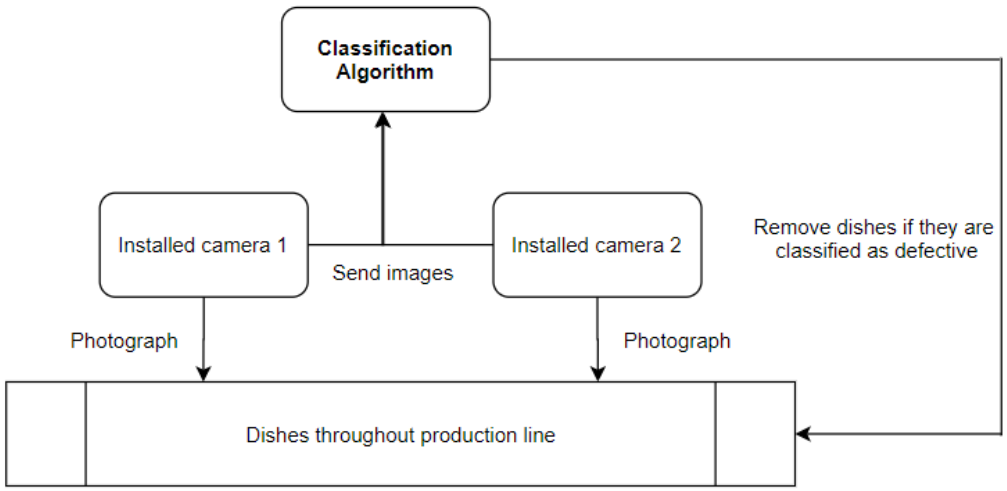

Figure 3.2 – Automated Quality Control System.

The proposed system implies placing the cameras at critical points along the production line and capturing images of all the dishes passing through their field of view. Once these images are acquired, they are then submitted to the classifier algorithm that should classify the dish as defective or not. Depending on the result, the dish may be removed from the production line by the factory workers or by an automatic system which can be implemented in the future. This mechanism would enable quality control in real-time as well as more efficient resource usage. In addition, it will be possible to identify a much larger number of defective dishes when compared with the current manual sampling quality control procedure. Therefore, it is necessary to optimize the developed classifier with the goal of minimizing the number of false positives, while not compromising the detection of defective dishes.

In the first phase of the system's implementation, the dishes classified as defective will be tagged by a small "invisible" ink stain. At the end of the production line, factory employees would be checking to see if the dishes with the paint stains were defective or not. When there are enough verifications made by the factory workers, an evaluation of the performance of the algorithm will be made under real test conditions. Based on these evaluations, some adjustments might be necessary for the classifier.

After this analysis, the second phase of the system's implementation would include mechanisms for automatic production line removal of the dishes classified as defective. This would increase production efficiency and thus fulfil the ultimate objective of the project.

The realization of this work involves the collaboration between ISCTE-IUL and INOV INESC Inovação. This dissertation is focused on the development of the binary classification algorithm for the dish images. The choice of cameras, as well as their placement directly on the production line, are not in the scope of this dissertation. The complexity of the problem will also be simplified by considering the frontal side of the dishes only.

In the following section is detailed the first CNN architecture used in this research work.

3.2 Classification Algorithm

Since no deep learning-based algorithms for detecting defects in manufactured dishes were found in the literature (namely those based on CNN's), it was decided to start from a simpler CNN architecture as a trade-off solution between the implementation complexity, the time required for training and achieved results. Therefore, a possibility would be to investigate compromise solutions in other problem domains.

One possible CNN architecture that fulfills the above criteria is the one proposed by student Carolina Gonçalves in her dissertation [33]. This algorithm aims to identify image regions containing the presence of the invasive plant species *Acacia Longifolia*. It starts by splitting a high-resolution aerial image acquired by a drone into smaller squared images which are then submitted to an image classifier. The classification algorithm is based on a CNN architecture and classifies the images int two possible classes, "with" and "without" invasive plant species. Since the CNN architecture used in this work was relatively simple and achieved high classification accuracy in a binary classification case, it was decided to use it as the starting point for the present work development. Nonetheless, the original architecture described in [33]

was subsequently adapted and modified by considering the results obtained in the experiments for the specific case of this dissertation.

The starting CNN architecture used in the scope of this dissertation is depicted in Figure 3.3.

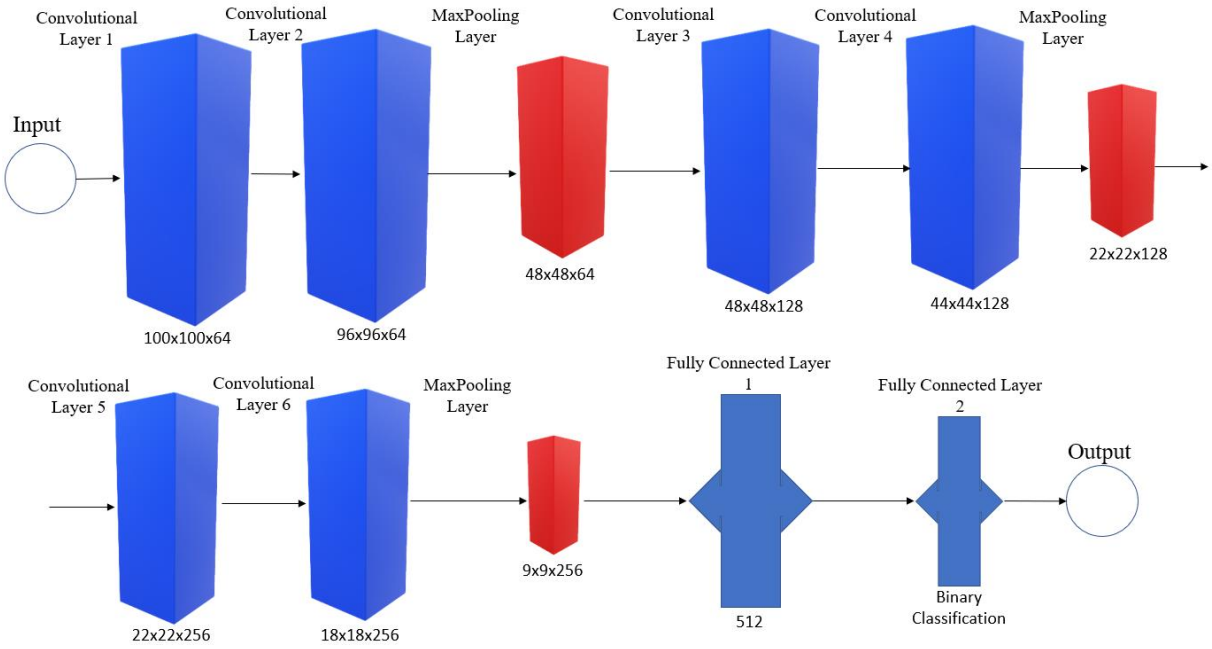

Figure 3.3 – First architecture setup.

The inputs of the CNN are 100x100 pixel images. The kernel size in all Convolutional Layers and MaxPooling Layers is five, and the Stride is 1 in all these layers as well.

This architecture is very simple, containing a MaxPooling Layer for every two Convolutional Layers. The final stages of the CNN are two Fully Connected Layers that are basically a "classic" neural network-based classifier. The last layer generates an output that allows classifies the dish as "with defect" or "without defect". The experiences related to adaptations of this architecture are described in Chapter 4.

3.3 Data Acquisition

This section describes the original procedures for obtaining the dataset, the problems caused by the COVID-19 pandemic that led to the failure of obtaining this dataset in a timely manner, and the workaround solutions that were implemented.

It was previously defined that the dataset used for CNN training and validation would be obtained directly from the production line. Figure 3.4 illustrates the intended data acquisition.

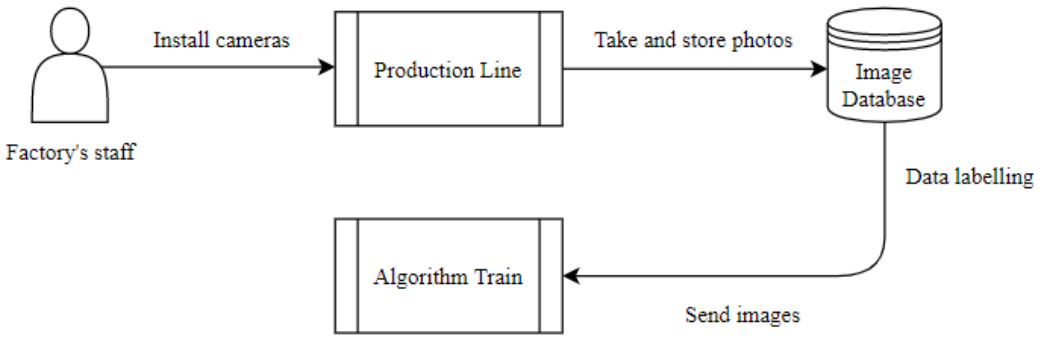

Figure 3.4 – Data acquisition.

The first step, as stated earlier, would be to install cameras at different locations of the factory's production line. That would allow to quickly obtain a dataset with a large number of images. Then, the images would be stored in a database. Access to the database would allow data labelling, performed by the factory employees. Finally, it would be possible to perform the training and testing in different classification models.

The project was scheduled to start on March 1st, 2020. However, the global health crisis due to the Covid-19 pandemic hit Europe in large numbers around that time, and the factory that should provide the images stopped its production. The project was set to standby, as it was not possible to obtain a dataset under real conditions. In order to overcome this situation, alternative solutions for getting image data have been carried out.

The following two sections explain what was done to generate the data that was used for training the CNN models proposed in the scope of this thesis. Section 3.4 describes the process of creating a dataset only with images of dishes found online, introducing the defects in those dishes manually. In Section 3.5, it is explained how a script developed by INOV Inovação, which functioned as a generator of dish images, was used to obtain a more realistic dataset.

3.4 Preliminary Dataset Development

A possible solution that could eventually provide this work with sufficient image data would be to use an online dataset containing manufactured dishes with and without defects. However, after an internet search, no datasets with the desired data have been found. Therefore, it was decided to create an artificial dataset from dish images found on online stores that resembled the images provided by the factory. At this time, since access to the factory was not allowed and the project was still at a very early stage, we had not yet been supplied with actual examples of produced dishes on the mass production line.

Since training a CNN requires a large amount of data, after downloading a set of online dish images, defects were manually inserted onto those images, and data augmentation techniques were used in order to generate a large amount of images that would allow training of the algorithm properly. By creating a dataset that allows the training of a deep learning algorithm, it is possible to generate models. A dataset properly labeled by the factory workers can be later be provided to these models, which would allow obtaining results faster. If these models did not generate the desired results, adjustments in the models could be performed.

3.4.1 Original Images

A mass production line guarantees the same environment and similar illumination conditions on the acquisition of images for all produced products. Assuming similar image acquisition conditions, 10 images of dishes, all with a clear background, were downloaded from online stores. A few dish image examples are shown in Figure 3.5.

Figure 3.5 – Example of some of the original dishes.

As previously mentioned, after an internet search, it was not possible to find images of dishes with manufacturing defects. The adopted solution was to manually insert defects in those 10 original images. An effort to diversify the inserted defects was made, not only in terms of shape and number but also with respect to their location on the dish. The result was 10 images of dishes with defects and 10 images of dishes without defects. Examples of images with manually inserted defects are shown in Figure 3.6.

Figure 3.6 – Example of some of the original dishes with defects.

3.4.2 Data Augmentation Techniques

In order to train a CNN, a large set of images is required. When the dataset does not have enough images to train the CNN, it is possible to apply data augmentation techniques to generate additional images. In this work, since there were only 10 images of dishes without defect and 10 images of dishes with defect, many data augmentation techniques were used. To perform these augmentations, a few MATLAB scripts were developed. The first developed script had the function of generating images based on the rotation data augmentation technique. An input image is rotated from 0 to 330 degree, with steps of 30 degrees. For each rotation step, the output image is stored. This script was applied to the 20 original images, from each resulted 220 additional images. A rotation example is illustrated in Figure 3.7.

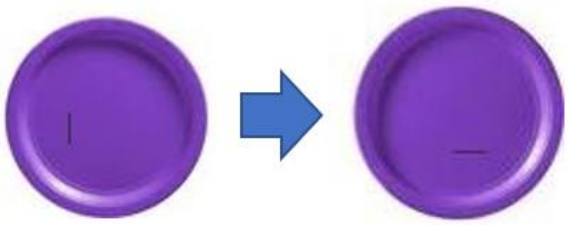

Figure 3.7 – Example of a 90-degree rotation.

Afterwards, a second script was developed. This time, the script had the goal of shifting the previous 220 images 1 to 5 pixels in all directions (right/left and up/down, individually) and added them in the previous folder. Figure 3.8 shows an example of the result of this script.

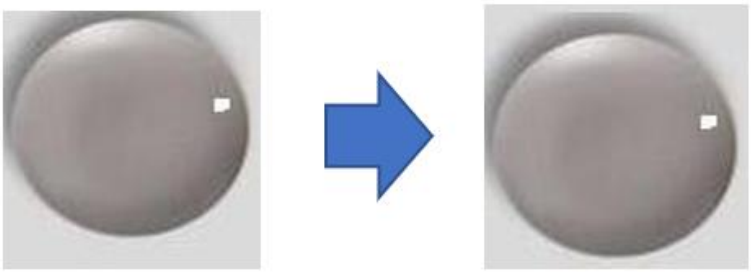

Figure 3.8 – Example of shift.

The last developed script had the goal of changing the intensity of the pixels. To do that, the values of the pixels were multiplied by a value in the range from 0.9 to 1.1 with steps of 0.02. The values below 1.0 turned the images darker, while the values above 1.0 turned the images brighter. These augmentations were performed to all the images obtained with the earlier techniques used. With all these augmentations, it was possible to create a dataset with thousands of dish images with and without defect. Figure 3.9 shows an example of this last procedure.

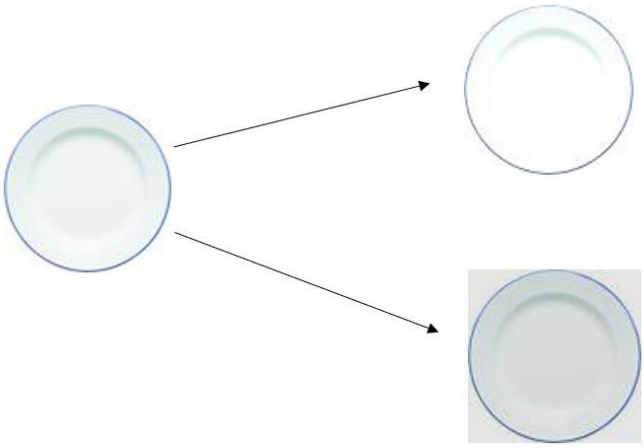

Figure 3.9 – 1.1x original image is on top and 0.9x original image is down.

3.4.3 Preliminary Dataset

After applying the data augmentations techniques described in the last subsection, a dataset with 61911 dish images was generated. The images of this dataset are organized according to 2 classes (dishes with and without defects). The total number of images depicting dishes without defects is 30938, and the remaining 30973 images are of dishes with one defect. The final dataset distribution is synthesized in Table 3.1.

The images were saved in two different folders according to their class. This is an important practice in deep learning because each class of the model must be properly labelled and separating the classes assigning them to different local folders makes this task a lot easier.

All the images were resized to 100x100 pixels using bilinear interpolation, in order to normalize image size in the dataset and to allow them to be submitted to the CNN input. On the other hand, the resizing procedure reduced the initial quality of the images.

Image distribution of the preliminary dataset		
Total number of images	**With defect**	**Without defect**
61 911	30 973	30 938

Table 3.1 – Distribution of the preliminary dataset.

The dataset described in Table 3.1 was used in order to be able to implement the CNN architecture and subsequent adjustments. As at the time there was no access to real data, and there was no forecast of when this would happen since there was confinement due to Covid-19. Thus, this was the solution found for the work not to freeze.

It was known at this time that the dataset in realistic terms it was far from the conditions under which the system is supposed to work. Consequently, the results obtained in the tests performed could not represent reliable approximations to the results obtained when evaluating the classifier with real data. The two key factors that conditioned this dataset were:

- *Generated defects* - The defects generated in this dataset are easily visible to the naked eye, which might not be the case in real defects from the factory's production line;

- *Resolution* - The cameras to be installed on the production line will acquire high resolution images (5184x3888 pixels). Therefore, the resolution of the preliminary dataset images is too low when compared to the resolution of the images to be acquired in the production line.

Due to the reasons presented above, a more realistic dataset was subsequently generated. This dataset generation is described in the next section.

3.5 Realistic Dataset Development

This section describes the generation of the images of dishes that represent the more realistic dataset. The creation of this dataset was possible since the factory's employees sent some samples of dishes directly manufactured on the production line to INOV Inovação. This way, it was possible to provide the script textures from the dishes and recreate two types of known defects mentioned in Section 3.1 (*cracks* and *stitches*).

3.5.1 Dish Image Generator

A dish image generator was created at INOV Inovação. This dish image generator aims to create a simulated environment as similar as possible to the factory's environment. This way, the results obtained will be more reliable for implementation in a real scenario. In the following

two subsections it is explained how the script backend (what the script does at the image processing level) and frontend (the user interface) works.

3.5.2 Script Backend

The scripting backend was developed to allow the generation of defects in a random position on the dish surface. This script also allows adding (or not) noise, either into the dishes, into the background, or on both.

It is also possible to insert shadows on the generated images of dishes. For this purpose, a developed function multiplies the value of the outermost pixels of the dish's surface by a static variable, in order to dusk them and simulate a shadow.

As for the inserted defects, it is possible to increase or decrease their length, as well as their intensity (increasing or decreasing the intensity of the pixels). In addition, the script backend also provides the support for the generation of the defect's contours, which allows the user to quickly check the defect's location in the image since sometimes they are difficult to discern by observing the image.

The output images of the script have a default size of 5184x3888 pixels, in order to simulate the resolution of the images captured by the cameras that will be placed on the production line. These images are therefore too large to be immediately provided to a CNN, so additional image preprocessing procedures are required. Further details can be found in Chapter 5. of this dissertation.

3.5.3 Script Frontend

The script frontend allows quick manipulation of the parameters used to generate defects on the surface of the dish, using checkboxes and sliding windows. By clicking on the *preview* button, it is possible to find a sketch of the first generated image by the script. It is also possible to edit the parameters and perform a new preview if the first results are not desirable. Figure 3.10 shows a frontend example of the script.

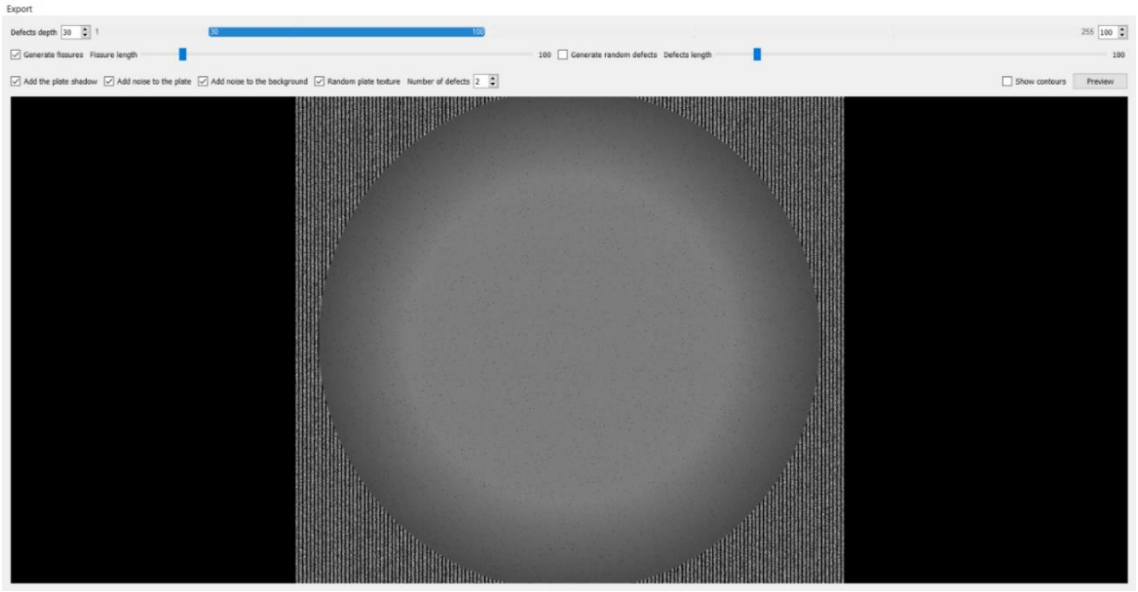

Figure 3.10 – Frontend example with default values.

When generating the dishes, by clicking on the *export* button, the script offers an option to choose the number of dishes that will be generated. If the number is greater than 1, all the dishes generated will have the same parameters (*e.g.*, types, number of defects and length of the defects), but the inserted defects will be randomly distributed across the dish surface.

Finally, for a faster location of the defects generated by the script on the dishes, by clicking on the button *show contours*, the defects are overlapped by a reddish color that allows an easier visual inspection, since this color stands out from the rest the surface of the dish. Figure 3.11 illustrates an example of the defect contour.

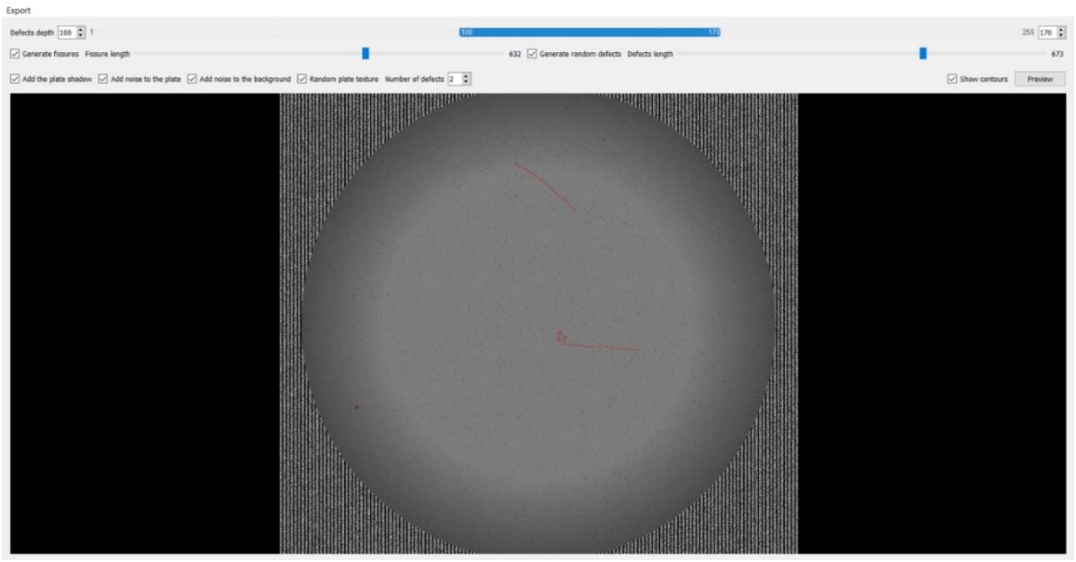

Figure 3.11 – Example of defect contour (in red).

3.5.4 Realistic Dataset

Using this script, 25200 images of dishes were generated. The images of this dataset are once again organized according to 2 classes (dishes with and without defects). The total number of images depicting dishes without defects is 12600, and the remaining half are images of dishes with one or more defects. This final dataset distribution is shown in Table 3.2.

As these images had large dimensions, a substantially larger memory space was needed in order to store them compared to the space needed to store the preliminary dataset.

Image distribution of the realistic dataset		
Total number of images	**With defect**	**Without defect**
25 200	12 600	12 600

Table 3.2 – Distribution of realistic dataset.

The dataset described in table 3.2 was provided to the CNN architecture that obtained the best results (these results are shown in Chapter 4.) when trained and validated with the preliminary dataset. The results obtained with this realistic dataset are described in Chapter 5.

Chapter 4. Preliminary CNN Architecture Experiments

This chapter describes tests and experiments performed using the CNN architecture presented in Section 3.2 of the previous chapter. These tests include modifications on the dataset structure and on the architecture of the CNN. Some of these tests were performed due to errors related to the lack of memory. Others were only performed for fine tuning the CNN.

As previously said in Chapter 3, since this dataset was the only one available for several months, many tests with different architectures were performed using its data. Comparisons were also performed in order to evaluate which parameters best suited the problem in question. The use of pre-trained architectures (Transfer Learning) was also evaluated.

The main goal of these tests was to understand if the models could generalize details of the dishes when classifying these. With these experiments, it would be possible to ascertain whether the models had the capacity to acquire knowledge from images of dishes and whether it is possible to distinguish defective dishes from non-defective ones.

To understand the performance of the model, the values of the accuracy and loss of the models were compared. The loss is a measure used in the training of the model that is calculated throughout the chosen loss function. The accuracy is calculated by dividing the correct number of associations between the images and the respective labels by the total number of images in the dataset.

Another relevant variable in the performance of the model is the time it required for training, considering the available resources.

All the experiments were carried out using a Nvidia GeForce GTX 1050 4 GB, with 8 GB of RAM and an Intel ® Core™ i5-7300HQ CPU @ 2.50GHz.

4.1 Hyperparameter Settings

In order to train the CNN architecture described in Section 3.2, it was necessary to define the optimization and the loss functions. The chosen optimization function was Adam function [34] while chosen the loss function was the MSELoss (Mean Squared Error Loss). This function creates a criterion that measures the mean squared error between each element in the input x and label y. The activation function used between the Convolutional and MaxPooling layers and for

the first Fully Connected Layer was the Rectified Linear (ReLU). The last Fully Connected Layer uses a SoftMax activation function since it is the output layer. A dropout rate of 0.5 was used after every Convolutional Layer.

80% of the dataset images described in Section 3.4 of the previous chapter were used for training the CNN, and the remaining 20% were used for testing the validation loss and accuracy.

Additionally, the batch size as set to 64. It was decided to train repeatedly this CNN architecture with variations on the learning rate with the goal of realizing which was the best value. Once the best result was obtained, the number of training epochs could be increased.

The next section illustrates the evolution of the training and validation accuracy and loss in the experiments performed.

4.2 First CNN Architecture Experiment

To be able to obtain a good visualization of the information on the value of accuracy and loss over the training period, it is not only better to represent this evolution graphically, but also increase the number of epochs. The following tests were started by changing the learning rate values to check if there would be some significant impact. The number of training epochs was set to 25. The *xx* axis represents the number of epochs while the *yy* axis represents accuracy and loss, respectively. Figure 4.1, Figure 4.2 and Figure 4.3 show the generated graphics.

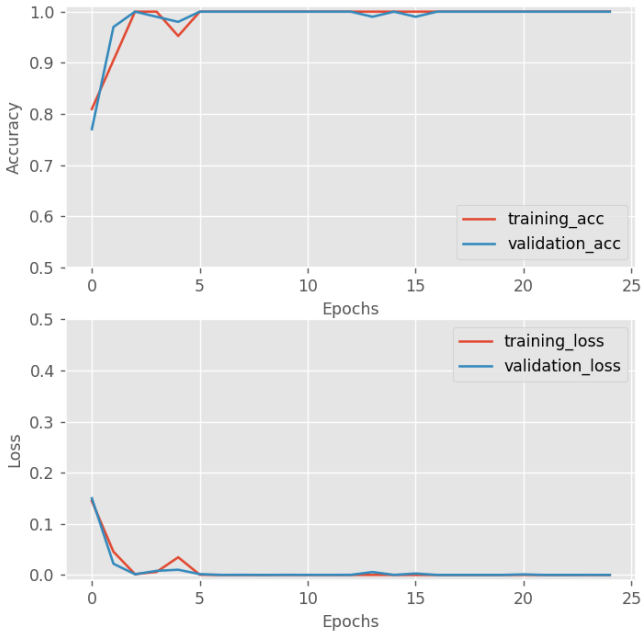

Figure 4.1 – Evaluation with learning rate = 0.00005.

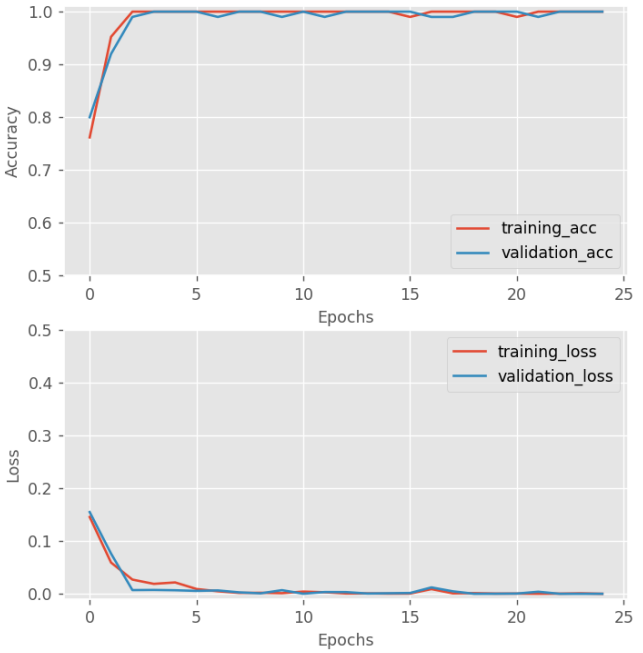

Figure 4.2 – Evaluation with learning rate = 0.00010.

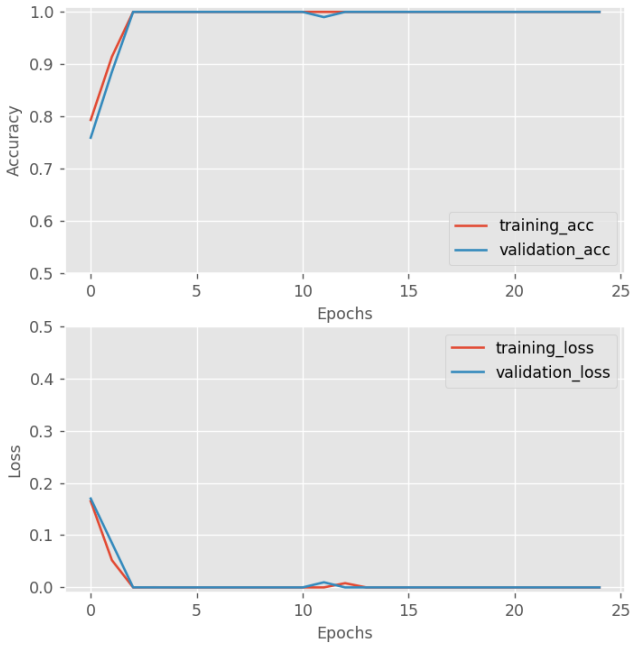

Figure 4.3 – Evaluation with learning rate = 0.00015.

The difference between a learning rate of 0.00015, 0.00010 and 0.00005 does not seem to be relevant. In total, the CNN took about 179 minutes, or two hour and fifty-nine minutes to train in all these experiments.

By analyzing these two plots, it is also possible to conclude that the values of training accuracy and validation accuracy are very similar, as well as the values of training loss and validation loss.

Although the validation images were not provided to CNN during training, they are very similar to the ones that were used for that purpose. Remember that the 60000 images dataset under use was generated by applying data augmentation techniques to 10 initial dish images. Therefore, since the images on training and validation sets are very similar, their learning curves overlap, and all the dish images end up correctly classified.

This classification performed by the algorithm seemed too easy, as it correctly hit all the results. To verify how the CNN is able to learn to identify defects in different dish models from those used in training, the proposed solution was to change the dataset structure. All the images of two of the original dishes were removed from the training set and saved in another location. The removed images were used on the validation set, while the training set was composed of

the remaining ones. Thus, after training, CNN no longer considers any previous knowledge of the dish models in the validation set. This solution is illustrated in Figure 4.4.

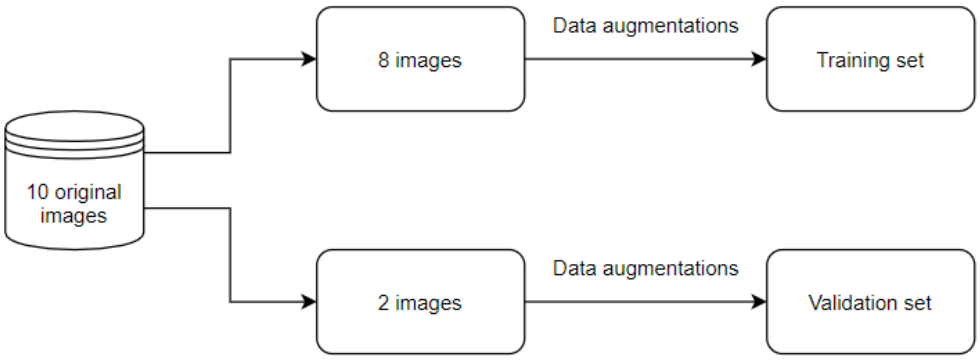

Figure 4.4 – Solution to diverge training and validations values.

Based on the previous tests, the number of training epochs was increased in order to check the behavior of the training and validation values over more time. The learning rate was set to 0.00010 as this value is a middle term of the ones tested previously. The batch size was again set to 64. The hyperparameters setup was as follows:

- Learning rate = 0.0001;
- Batch size = 64;
- Epochs = 100.

Figure 4.5 shows the evolution of the accuracy and loss using this setup along 100 training epochs.

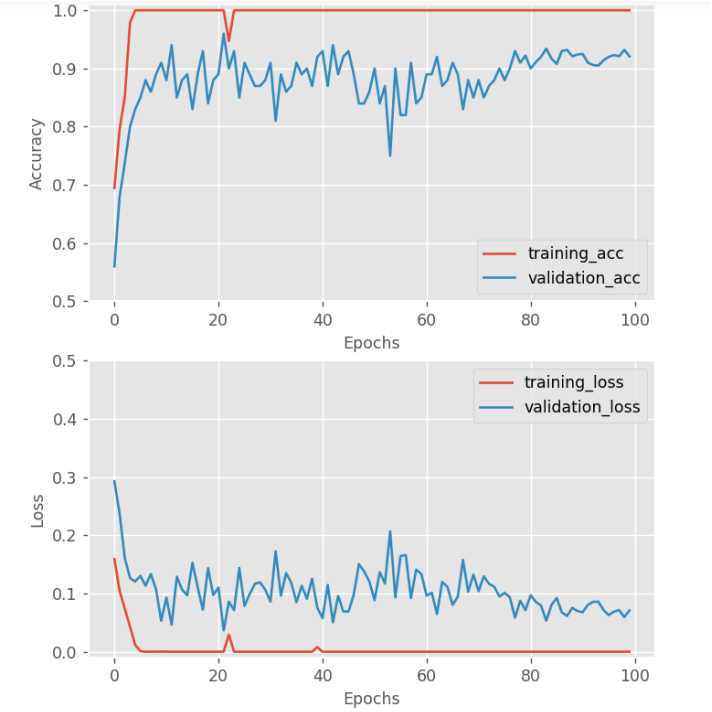

Figure 4.5 – Results for the dataset modification.

.

Each epoch took about 7 minutes and 10 seconds, which ended up in total training time of approximately 11 hours and 56 minutes.

The values of accuracy and loss are no longer the same for the training and validation sets. By observing the learning curves, the CNN overfits to the dish model images used for training, since not only the training accuracy reaches the maximum value of 1 and the training loss reaches the minimum value of 0, but also the validation accuracy, as well as the validation loss, cannot reach the same values, which makes the training and validation curve very distant. Considering the process of generating the dataset, it can be concluded that the dataset is not sufficiently representative for generalizing the defects detection for different dish models. For the validation curves to approach the training curves, it would be necessary for the dataset to contain more models of dish images. Although the validation accuracy and loss values are very unstable throughout most of the epochs, they begin to being less unstable around epoch 75.

The following section describes experiments performed on the CNN architecture and their respective results.

4.3 Architecture Modification Experiments

Additional tests regarding the architecture of the CNN were also carried out with the goal of understanding the impact of the MaxPooling layers on the performance of the CNN. The first architecture modification done was to remove the Convolutional Layer 4 and the respectively MaxPooling Layer, as shown in Figure 4.6.

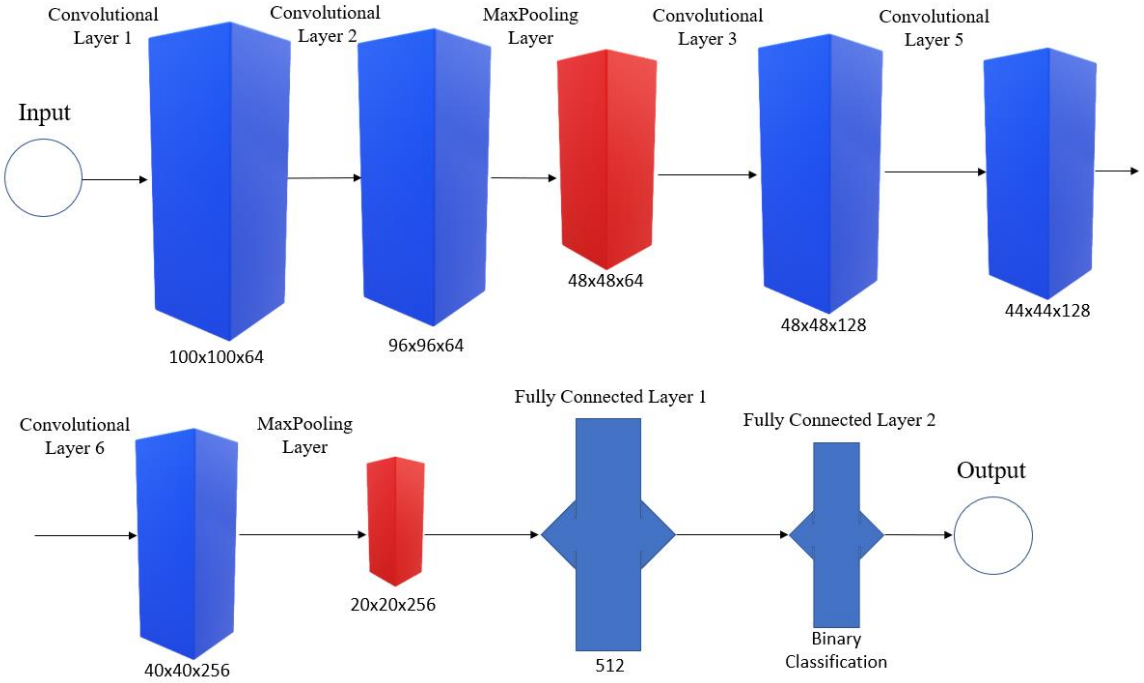

Figure 4.6 – Second architecture tested.

These architecture modifications also affected some specifications of the layers. The layers of the CNN that appear after the removed layer changed the size of the input and output matrix. Kernel size and Stride are not affected since those values are manually defined.

The main difference in this architecture is the fact that a MaxPooling Layer is skipped, which hugely modifies the resulting matrix size in the following Convolutional Layers and substantially increases the *features* of the first Fully Connected Layer. Figure 4.7 shows the evolution of the accuracy and loss values per epoch.

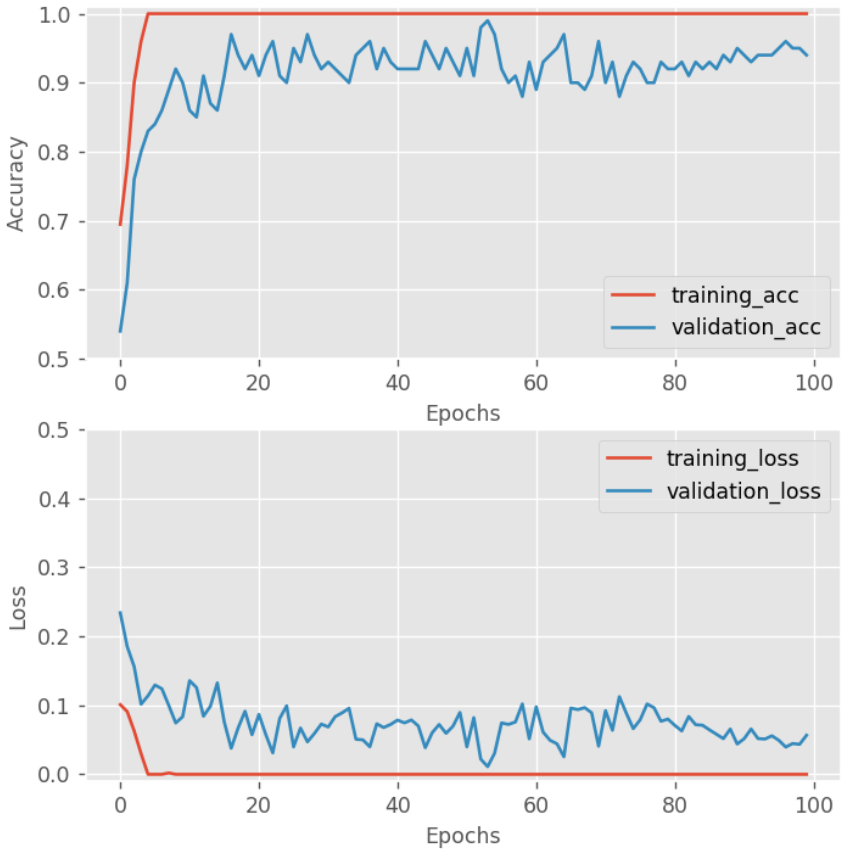

Figure 4.7 – Results for the second architecture modification per epoch.

Accuracy values seem to be slightly higher, and loss values seem slightly lower. However, since the *features* of the first Fully Connected Layer increased substantially (due to the fact that one less Max Pooling Layer did not halved the size of the matrix to be processed), the training time of the CNN has substantially increased from approximately 12 to 20 hours. This architecture took about 1200 minutes or 20 hours to train. The accuracy and loss values begin to be less unstable around epoch 80. The problems with overfitting mentioned in the previous architecture are equal since the dataset used was the same.

Another architectural modification was done by removing the last Convolutional Layer and the subsequent MaxPooling Layer, generating an architecture that is detailed in Figure 4.8.

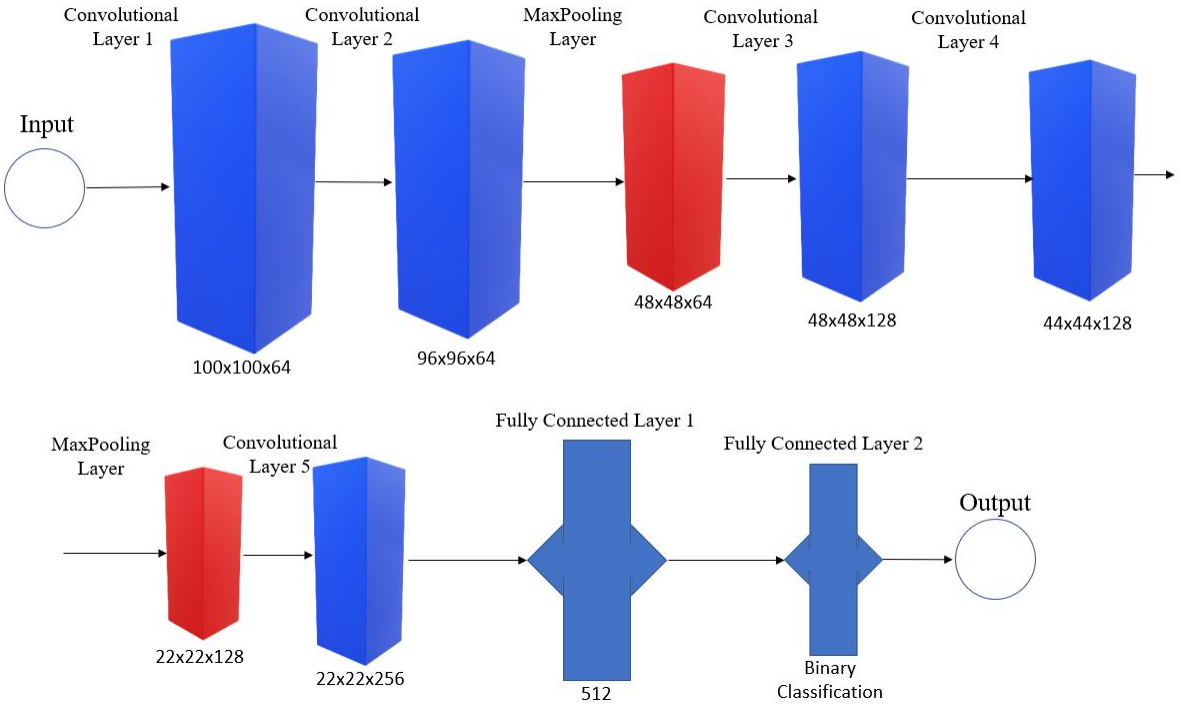

Figure 4.8 – Third architecture tested.

Similarly to the previous architecture, one MaxPooling Layer was skipped, which affected the number of *features* received in the first Fully Connected Layer. The other Convolutional Layers did not have any changes since the removal of the Convolutional Layer 6 did not influence the previous layers.

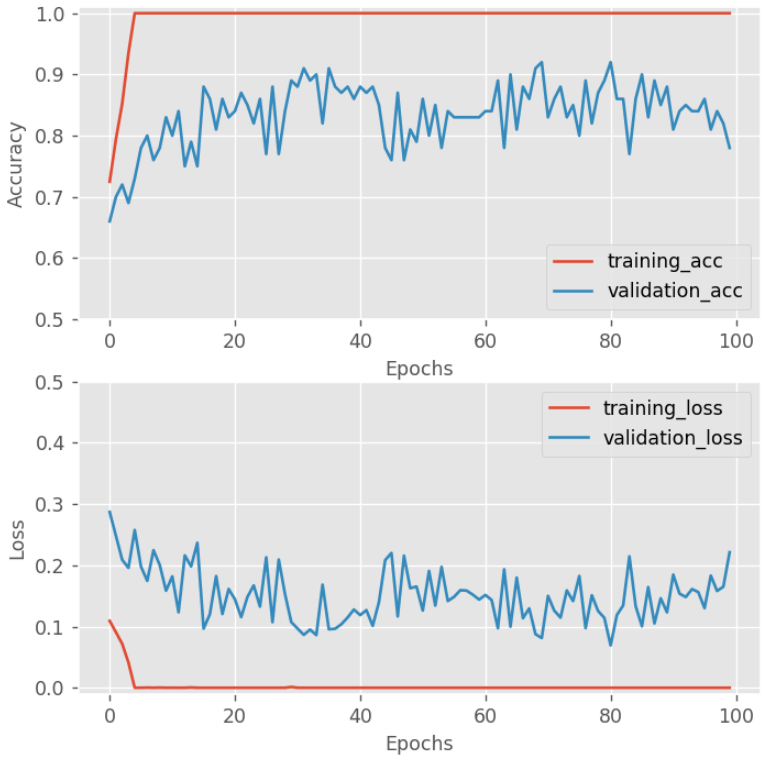

Figure 4.9 – Results for the third architecture modification, per epoch.

As shown in Figure 4.9, this architecture was also trained for 100 epochs. Since the scale of the graphics is the same, it should be noted that the accuracy values of this architecture rarely exceed 0.9 (or 90%) while the loss values are higher (except for rare exceptions) than 0.1 (or 10%). These facts show that there was a decrease in the overall performance of CNN compared to the other two experiences. Although the values do not stabilize, it is possible to verify that around the 90th epoch, the accuracy values start to decrease while the loss values start to increase. This architecture took about 915 minutes or 15 hours and 15 minutes to train.

The overfitting problem happens again as well as the representativeness problem since the dataset used was still the same. Although several data augmentations have been performed in this preliminary dataset, these are not enough to dispel the fact that there are too many images generated from so few original images.

The results of the accuracy values presented in Table 4.1 represent the accuracy obtained in the 100th epoch of each experiment.

Summary of the architectural experiences		
	Accuracy in %	**Time in Minutes**
Original Architecture	92.7	716
First Modification	94.5	1 200
Second Modification	78.9	925

Table 4.1 – Summary of architectural experiences.

To summarize, the first architectural modification was removing the Convolutional Layer 4 from the original architecture. The second architectural modification was removing the Convolutional Layer 6 from the original architecture.

The model obtained with the first modification in the CNN architecture was the one with the highest validation accuracy and the lowest validation loss, which makes it the one with the best accuracy results. However, this algorithm turned out to be heavier than the original architecture. The difference in training time between this architecture from which Convolutional Layer 4 was removed and the original architecture is approximately 500 minutes (8 hours and 20 minutes).

Bearing in mind that the most relevant factors for the adaptation of this algorithm were its performance and training time, it was considered that both architectures are good to progress in the work. It was decided to proceed with the original architecture since this architecture has a fast training time. Besides, this architecture has already shown great results in the binary classification application described in [33].

The second architectural modification was discarded since the validation accuracy, and validation loss do not reach such high values.

4.4 Transfer Learning Tests

The Pytorch platform includes pre-trained architectures that can easily be adapted to the purpose of classifying a specific dataset. To adapt these pre-trained architectures, it is necessary to fine-tune the last Fully Connected Layer without changing the previous weights of the CNN that were setup during the pre-train process. Fine-tuning consists of changing the number of

outputs of CNN to the number of classes to be classified. Experiments using the following architectures were performed:

- AlexNet;
- VGG16;
- ResNet18;
- InceptionV3.

These architectures were chosen because they obtained good image classification and detection results in the ILSVRC challenge mentioned in section 2.2.1.

The images were resized and normalized to the specific requirements of the architecture, and the batch size was also redefined due to memory constraints. The higher the batch size, the less time it takes to the CNN to process the dataset. In the other hand, more computational memory is needed. Having this in mind, these experiments were performed with the batch size equal to 32 in all architectures except in the inceptionV3, since this architecture is the heaviest in terms of computational memory. The batch size used in this last architecture was equal to 8. The results of these experiments are shown in Table 4.2.

Transfer Learning Results		
	Accuracy in %	**Time in minutes**
AlexNet	75.8	480
VGG16	95.2	2 150
ResNet18	90.2	990
InceptionV3	82.1	4 800

Table 4.2 – Transfer Learning results.

Table 4.2 shows that VGG16 model was the model that obtained better results in terms of validation accuracy.

The results of VGG16 exceeded the results of the previous, adapted CNN architecture. However, VGG16 took 2150 minutes or 35 hours and 50 minutes to train. The number is considerably higher than the number of hours that the adapted architecture took to process the dataset (8 hours and 53 minutes). In a simple classification task of only two classes (with or

without defect), despite the improvement in the accuracy values, it was considered that architecture so dense that requires a high computational capacity such as VGG16 is not necessary to the purpose of this work. In addition, it is practically impossible to adapt a learning transfer architecture to a specific problem since these are very dense architectures and not all their parameters are configurable.

It is also relevant to note that the InceptionV3 architecture was the one that took the longest to process the data. This was mainly because its computational weight is very high. Since the computational resources are limited, in order to obtain results with this architecture, it was necessary to decrease the batch size, which increased the training time.

Despite the accuracy of VGG16 being superior to the accuracy obtained by the adapted CNN architecture, it was considered that its computational weight was too high, and it was more favorable to progress in this project with the adapted architecture.

The following chapter contains experiences and results of using the chosen architecture with a more realistic dataset.

Chapter 5. Realistic Dataset Evaluation

This chapter describes the experiments performed using the dataset generated through the script developed at INOV INESC Inovação. This dataset was used since it allows to obtain more realistic dishes images, compared to the dataset mentioned in Section 3.4 of Chapter 3. In addition to experiments with images of the entire dishes presented in Section 5.1, experiments with segments of the original images were also carried out in Section 5.2. These last experiments had the goal to evaluating the advantages and disadvantages of classifying each segment individually.

5.1 Full Image Classification

The following subsections contain the results of the experiments with the images generated by the script described in Section 3.5 of Chapter 3. Although these are not images of real dishes, they are images that allow classifying industrial dishes closer to reality.

5.1.1 Adapted Architecture

The model generated with the original, adapted architecture in Chapter 4. was retrained and retested with images generated by this image generator.

Twenty-five thousand two hundred images were generated to perform these tests. Half of the images used (12600) represented defective dishes, while the other half represented non-defective dishes, as said in Chapter 3. 23200 images were used to train the CNN while 2000 were used to validate the results. These distributions are illustrated in Table 5.1. To perform these tests, the generated dishes of the class “with defects” contained between 1 to 4 defects, since in a production line it is unlikely that a dish will be produced with more than 3, a fact provided by the managers of the factory. The generated defects were alternated, with dishes with only cracks, dishes with only stitches and dishes with both. The intensity and length of the defects was also generated in a varied way to cover a wider range of possible real defects. The generated dishes did not present noise or shadows since they can disturb the accuracy results.

Dataset distribution			
	Total number of images	**Images with defect**	**Images without defect**
All Data	25 200	12 600	12 600
Training Data	23 200	11 600	11 600
Validation Data	2 000	1 000	1 000

Table 5.1 – Dataset distribution for the first experiment.

All the images generated by the script were resized using the *torchvision.transforms* python function, that uses bilinear interpolation, to match the input conditions of the CNN. The accuracy and loss evolution is shown in Figure 5.1.

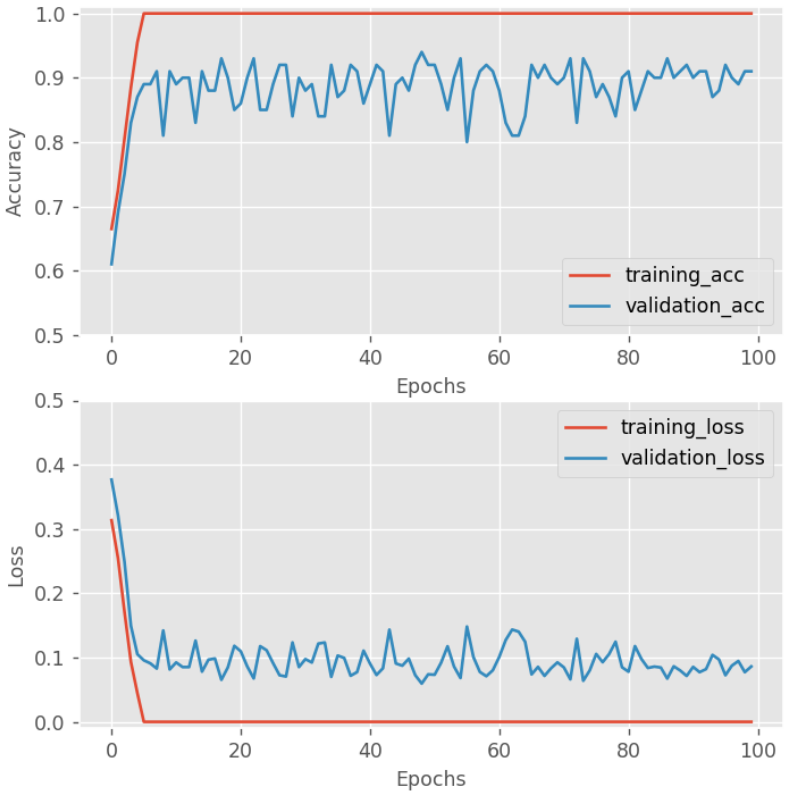

Figure 5.1 – Accuracy and loss evolution per epoch.

In order to ascertain which classifications were wrong, the confusion matrix for this model was generated and depicted in Figure 5.2. A confusion matrix evaluates the performance of a

classification model. The size of this confusion matrix is 2x2 since this CNN has 2 target classes. This confusion matrix was generated using the validation set results achieved at the last epoch of the CNN training.

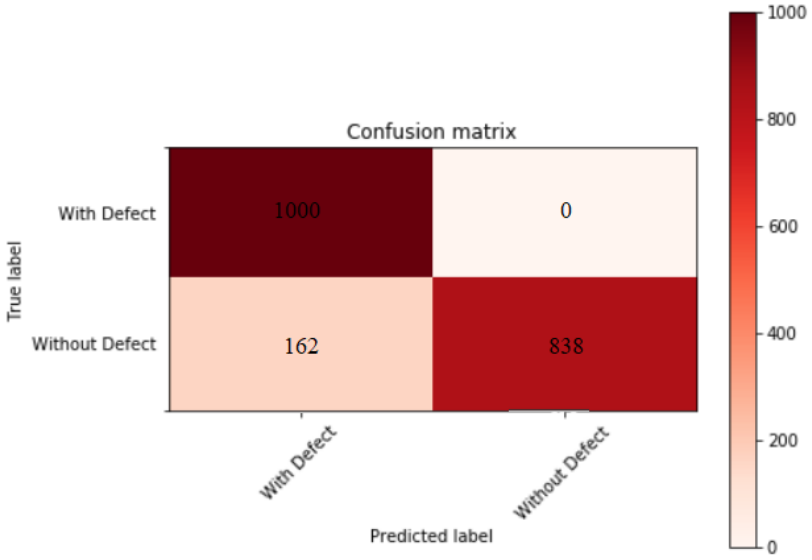

Figure 5.2 – Confusion matrix of the validation data.

By analyzing this confusion matrix, it can be noticed that this model correctly classifies all dishes that contain defects. 16.2% of these dishes are wrongly classified as "with defect". The accuracy of CNN is 91.9%.

It is now possible to compute the precision, recall and F1-score of the model. Precision evaluates how accurate are the predictions for a given class. The precision is computed as

$$\text{precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{5.1}$$

Recall can be interpreted as the measure that calculates the fraction of the actual positives that are correctly classified by the model. This measure is given by

$$\text{recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{5.2}$$

The F1-score combines the two previous measures in order to obtain a measure that covers the entire range of values of the confusion matrix. F1-score is calculated as

$$\text{F1} = \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{5.3}$$

Table 5.2 shows the results of these calculations. These values allow a better interpretation of the confusion matrix.

	Precision	**Recall**	**F1-Score**
Without Defect	1.00	0.84	0.91
With Defect	0.86	1.00	0.92

Table 5.2 – Precision, Recall and F1-score values of the previous test.

The precision value of the dishes “without defect” as well as the recall value of the dishes with defect reaches the maximum value of these parameters, 1.00 since all the defective dishes are correctly classified.

The precision value for dishes with the defect is an important parameter of this research work since the factory wants to minimize the number of non-defective dishes wrongly classified as "defective". 0.86 is a reasonable value, although a value closer to 1.00 would be better. It was preferable that some defective dishes were miss classified instead of non-defective dishes, as this represents a waste of properly produced dishes.

5.1.2 Transfer Learning

In addition to the adapted CNN architecture, the four previously architectures used for Transfer Learning used in section 4.4 were also reevaluated using the realistic dataset. These experiments were performed because the dataset used in this chapter is different, which modifies the results, compared to those obtained in the previous Chapter 4. Table 5.3 synthesizes the accuracy results.

Accuracy of the Transfer Learning architectures	
	Accuracy
AlexNet	73.1
VGG16	88.3
ResNet18	86.8
InceptionV3	87.4

Table 5.3 – Accuracy of the Transfer Learning architectures.

It should be noted that the VGG16 architecture was again the architecture that achieved the best results in identifying industrial defects in dishes, with an accuracy of 88.3%. The precision, recall and F1-score values of VGG16 are shown in Table 5.4.

	Precision	**Recall**	**F1-Score**
Without Defect	1.00	0.77	0.87
With Defect	0.81	1.00	0.90

Table 5.4 – Precision, Recall and F1-score values for VGG16 architecture.

The evaluated architecture in subsection 5.1.1 achieved a higher precision in the "with defect" class compared to the VGG16. Therefore, it is proven that for this dataset, it is also not worth using the transfer learning architectures suggested in this research work.

In order to try to improve the results obtained in this section, and based on the work [23], the following section demonstrates the experiences and results obtained when training and testing the CNN with small parts of the original image.

5.2 Classification Based on Image Parts

This section contains the results of experiments performed on smaller parts of the original images.

The size of CNN's inputs is usually small. In the previous section, to obtain defect detection results on dishes, the images have been down sampled to the pre-defined size of the CNN. This down sampling resulted in a loss of image quality, which probably led to a decrease in the effectiveness of the classification.

In [23], the authors propose to split the images of their dataset into smaller parts, in order to detect small scratches in cars. Bearing this in mind, an alternative to image down sampling is to split the original image a priori into parts that match CNN's input size. This way, loss of image quality when resizing is avoided. That said, each original image was divided into 100x100 parts.

After splitting, each image part was labelled into the classes “with defect” and “without defect”. Figure 5.3 illustrates the sub-images containing dish defects and images that simply contain non-defective parts of the dish.

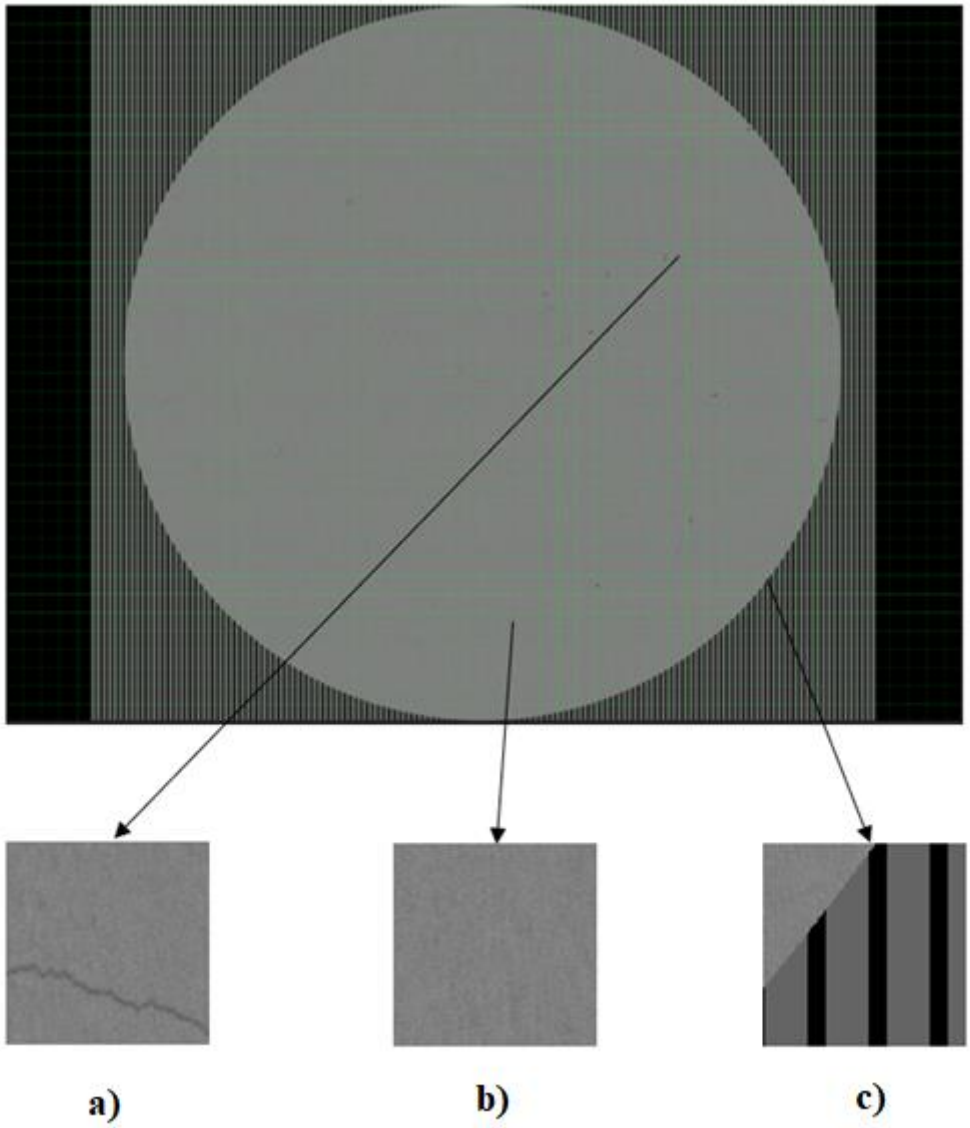

Figure 5.3 – Dish parts, a) with defect; b) without defect; c) with dish and background.

All images have the same dimensions. Each original image is divided into the same number of 100x100 parts, more specifically in 2028 100x100 sub-images. Of these 2028 sub-images, only 1223 are located on the surface of the dish. The images that only contained background were manually removed.

To develop the segmented image dataset, 47 full dish images were generated, all with 10 defects of each type (stitches and cracks) in order to obtain more sub-images belonging to the defective class using a smaller number of images. This allowed to significantly reduce the time needed to create the dataset. It is important to note that there are some defects that can be found in more than one sub-image.

After generating the 47 images, splitting them into 100x100 sub-images and removing the sub-images located on the image background (*i.e.*, sub-images located outside the dish boundaries), the dataset ended up with 58704 100x100 images. Of these 58704, 2580 contained defective dish parts while 56124 contains non-defective dish parts, as shown in Table 5.5.

Image distribution		
Total number of images	**Images with defect**	**Images without defect**
58 704	2 580	56 124

Table 5.5 – First 100x100 dataset distribution.

The CNN was trained with all the image parts available in the dataset. Since only about 5% of the dataset contains image parts with defect, it was likely that the CNN, in order to maximize the accuracy, would classify all images without defect, since this class represents 95% of all dataset. 500 images of each class were used to validate the results. Figure 5.4 shows the confusion matrix obtained.

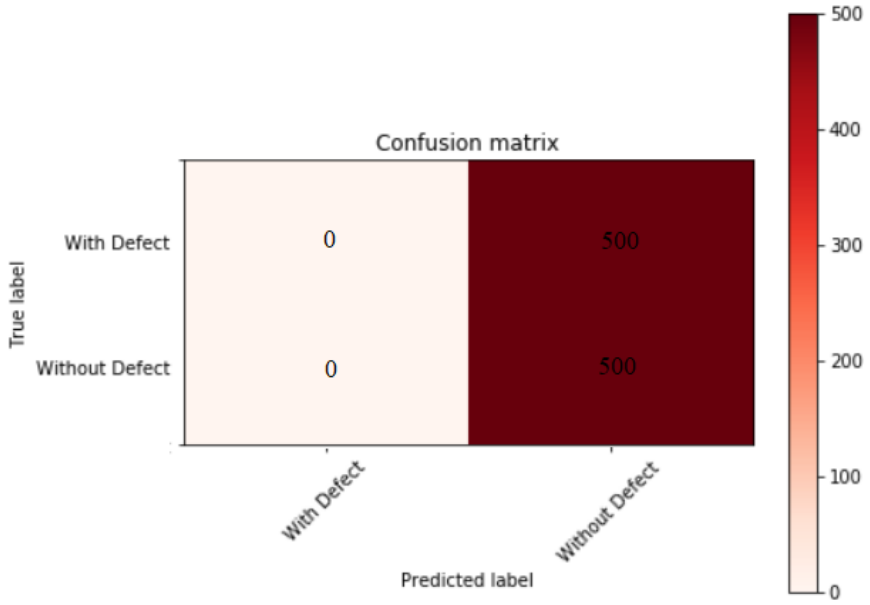

Figure 5.4 – Confusion matrix of the unbalanced dataset.

As expected, the algorithm classified all dish images belonging to the class “without defect”.

The next subsection focuses on solving problems due to imbalanced classes in the image parts dataset, since there are far more sub-images in the "without defect" class than sub-images in the "with defect" class.

5.2.1 Dataset Balance

This section shows the methods used to balance the dataset and the results obtained after training and validating the adapted CNN with the new generated image parts dataset.

To try to change the tendency shown in the previous subsection, data augmentation rotational was used with 90, 180 and 270 degrees. This allowed the number of defective images to be increased from 2580 to 10320. Thus, the distribution of the dataset has become approximately 20% images with defect and 80% images without defect.

The dataset was then divided into training data and validation data, as shown in Table 5.6.

Dataset distribution			
	Total number of images	**Images with defect**	**Images without defect**
All Data	66 444	10 320	56 124
Training Data	54 704	7 385	47 319
Validation Data	11 740	2 935	8 805

Table 5.6 – Second dataset distribution.

The CNN has trained once again, and the results were clarified through the confusion matrix regarding validation data illustrated in Figure 5.5.

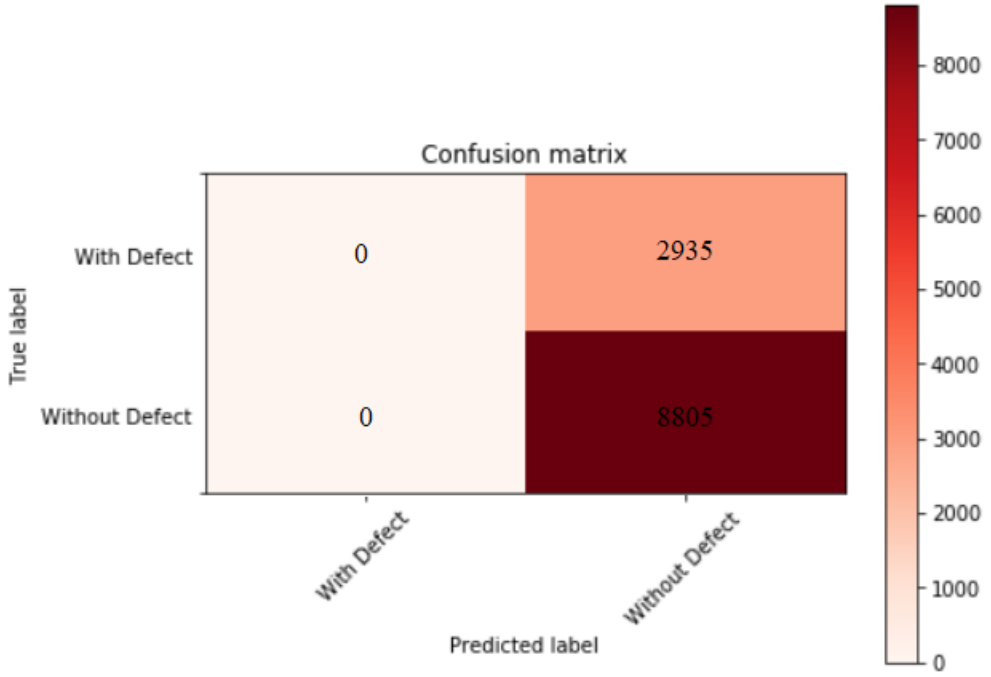

Figure 5.5 – Confusion matrix after rotational techniques

Based on the confusion matrix, the CNN continues to classify all results as "without defect", due to imbalanced training data. To dispel the disparity in the difference between the number of dish parts with and without defect in the training data, two methods were used.

The first method was to randomly remove images of segments of the dish without defect. Those images were saved in another local folder, and the same number of defective and non-defective images in the training and validation data were used. Table 5.7 shows the length of the datasets.

Dataset distribution			
	Total number of images	**Images with defect**	**Images without defect**
All Data	20 640	10 320	10 320
Training Data	14 770	7 385	7 385
Validation Data	5 870	2 935	2 935

Table 5.7 – Dataset distribution with the first method

Figure 5.6 depicts the evolution of the accuracy and loss values along 30 epochs, and Figure 5.7 shows the confusion matrix obtained at the last training epoch.

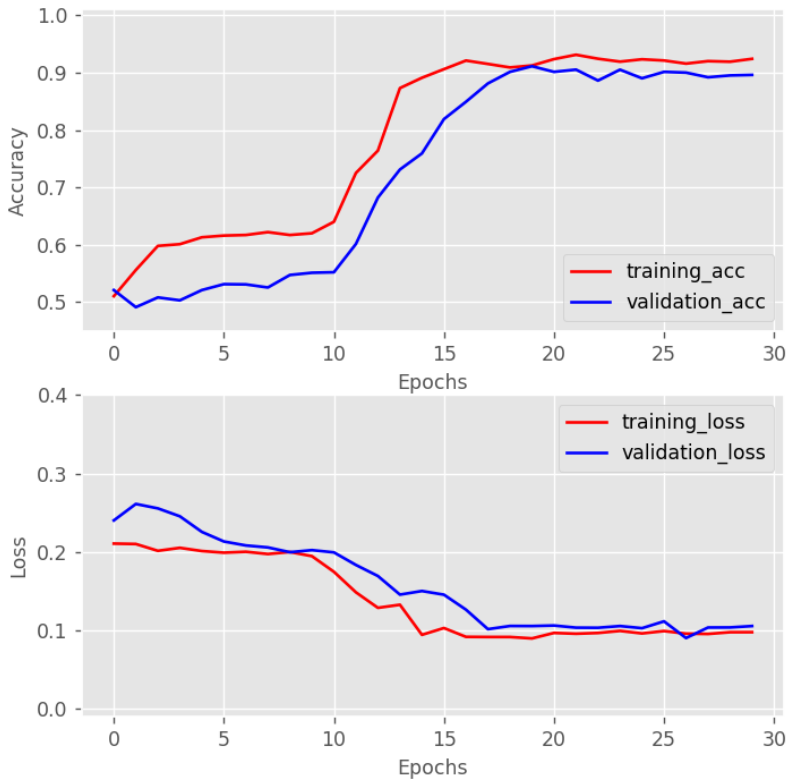

Figure 5.6 – Accuracy and Loss over the number of epochs.

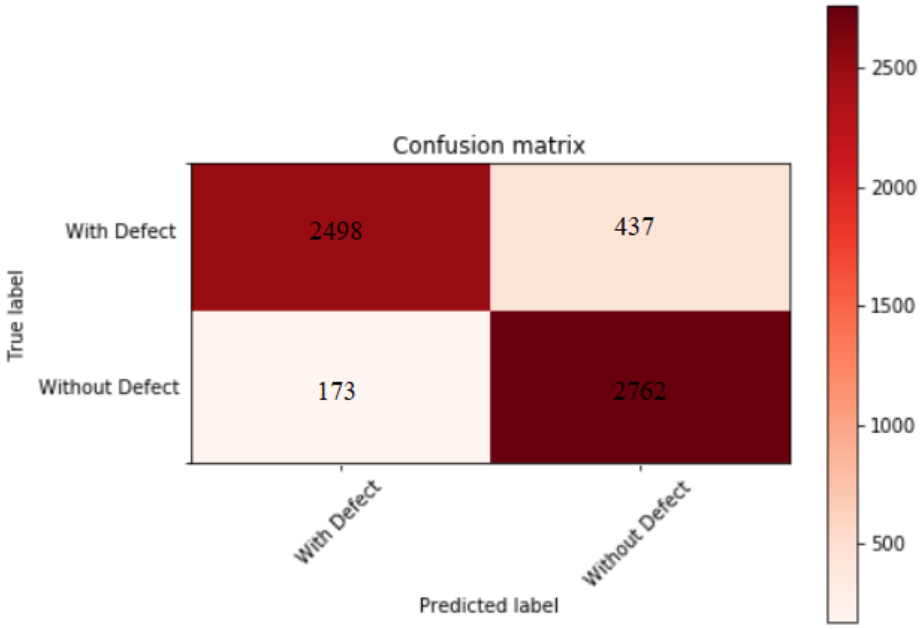

Figure 5.7 – Confusion matrix after manual dataset balance.

By analyzing the learning curves in Figure 5.5, as well as the confusion matrix, it can be concluded that the previous problem, due to an imbalanced dataset, has been overcome. It is also possible to notice that the training and validation curves have similar values, which indicates that overfit does not occur with this data. Since the images of this dataset are not subject to a resize of any kind, they contain more details which facilitates the image processing of the CNN. The confusion matrix is again related to the validation data. The CNN classified the 100x100 segments of the dishes with an accuracy of approximately 89.6%. It is possible to verify that 437 image parts of defective dishes were classified as non-defective, while 173 image parts of non-defective dishes were classified as defective. The parts of dishes classified as "without defect" but that were defective are dishes in which the defects are hardly visible because they are quite small in terms of size as well as the pixel intensity. An example of a hardly detected defect is present in Figure 5.8. The precision, recall and F1-score values corresponding to this confusion matrix are shown in Table 5.8.

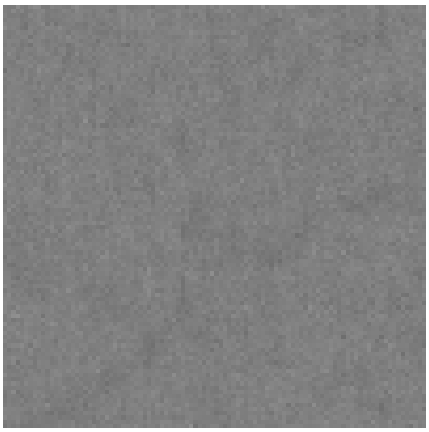

Figure 5.8 – Example of a hardly detected defect.

	Precision	**Recall**	**F1-Score**
Without Defect	0.86	0.94	0.90
With Defect	0.94	0.85	0.89

Table 5.8 – Precision, Recall and F1-score values of the manual balance.

.

The achieved results on the evaluation metrics show that this CNN performs well on classifying parts of the dish, with an accuracy of approximal 89.6%. It is important to mention that the value that has more advantages in being maximized (the precision for the dishes with defect), increased to 0.94. However, this value cannot be compared with the one previously calculated in Section 5.1.1. since it refers only to segments of the original image and not to the image of the entire dish. It only takes a wrong classification in one of these segments to misclassify the entire dish.

An alternative data balancing method was to use class weighting before training the CNN. This method balances the data by modifying the weight that each training sample contributes to the loss function. Usually, all training classes are equally weighted in the loss function. However, it is sometimes preferable to assign different class weights during the training process, especially if the number of examples for each class are noticeable imbalanced. Class weighting implies that the contribution of each example to the loss function is multiplied by the corresponding class weight. Therefore, during the training of CNN, this algorithm will obtain a greater loss if it classifies classes incorrectly with a greater factor. Since the CNN aims to

minimize the loss, the class with the higher weight will become the most relevant to the algorithm.

To use this approach, every available image in the dataset was used. The validation set for all these experiments was the same as the previous method, with 2935 100x100 images with the defect and 2935 100x100 images without defect.

Several weight variations in the classes and adjustments to the number of examples in each class were tested, in order to obtain results that match the initial expectations. Based on trial and error, the weight values of the classes were set to 0.9 for the “with defect” class and 0.1 for the “without defect” class.

With 35000 images of "without defect" class with a weight of 0.1 and 7385 images of "with defect" class with a weight of 0.9, the CNN obtained an accuracy of 92.4% in the binary classification of 100x100 segments of the original image. The confusion matrix was once again generated from the results of the last training epoch (30th epoch). Figure 5.9 shows the evolution of the accuracy and loss values, and Figure 5.10 the confusion matrix.

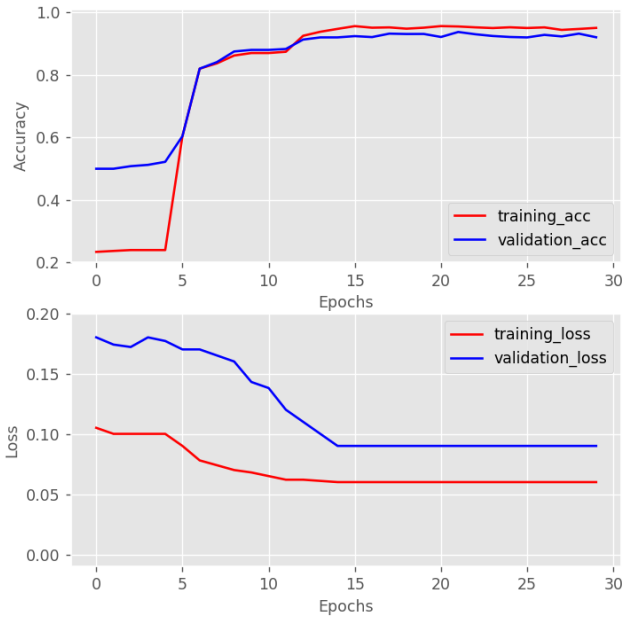

Figure 5.9 – Accuracy and Loss values over epochs.

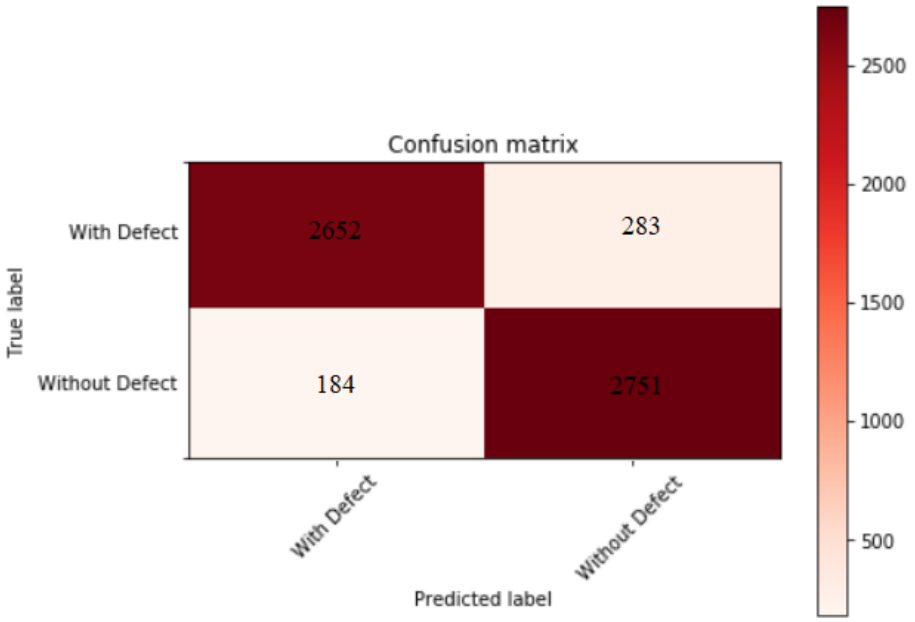

Figure 5.10 – Confusion matrix of the weight classes experiment.

The CNN training lasted 30 epochs since both validation accuracy, and loss values were stable at this point. In the first 5 epochs, CNN only achieved a validation accuracy of approximately 20%. This happened because the weight of the "with defect" class was much higher than the weight of the "without defect" class, which led the CNN to try to minimize losses and classify the vast majority of validation data as belonging to the "with defect" class. From epoch 5 to epoch 12, the CNN acquires greater knowledge through training data, which results in an increase in accuracy values, both for training and for validation. Until epoch 30, these values, as well as the loss values, remained stable.

From the 2935 images with defect, 2652 were correctly classified while 283 were classified as “without defect” by the CNN. From the 2935 images without defect, 2751 were correctly classified while 184 were classified as “with defect”. The precision, recall and F1-score values are calculated in Table 5.9.

	Precision	Recall	F1-Score
Without Defect	0.91	0.94	0.92
With Defect	0.94	0.90	0.92

Table 5.9 – Precision, Recall and F1-score values with class weights.

The method used turned out to be "hybrid", since, in addition to having adjusted the class weights, the number of images in the "without defect" class was also modified.

The overall accuracy obtained with this method is higher than the overall accuracy obtained by manually balance the dataset. This method allowed to reduce the number of false negatives (segments of the dish with defect that were classified as “without defect”) from 437 to 283. The number of false positives (segments of the dish without defect that were classified as “with defect”) increased from 173 to 184, but this change is not significant.

Based on the obtained results in the accuracy, as well as in the confusion matrix, it was concluded that with this architecture, it is preferable to balance the dataset with “hybrid” method instead of trying to balance it manually.

The following subsection compares the results the classification of dish images based on image parts classification with those obtained by directly classifying a subsampled representation of the full dish image, described in section 5.1.

5.2.2 Dish Classification Results

By dividing the dish images into 100x100 sub-images and identifying these segments in order to know which dish they refer to, it is possible to try to use this strategy to improve the quality control system of the factory. However, the accuracy percentage obtained earlier is not comparable with the remaining percentages obtained through the classification of an entire image of a dish, since only one misclassified segment is enough to misclassify the entire dish.

In order to test the best results obtained in the previous section, the image generator was used once again to generate 100 new dishes. Fifty dishes were generated with defect/s while

the other 50 were generated without defects. These 100 dishes were again split into 100x100 sub-images and those containing background only were removed. Each generated sub-image was identified in order to allow an association with the corresponding dish image. By providing this data to the CNN, it was possible to verify that 29 dishes (58%) of the dishes without defects were correctly classified, while 23 (46%) of the dishes with defects were correctly identified. The overall accuracy of the model was 52%.

Figure 5.11 shows the confusion matrix obtained and the precision, recall and F1-score values are calculated and described in Table 5.10.

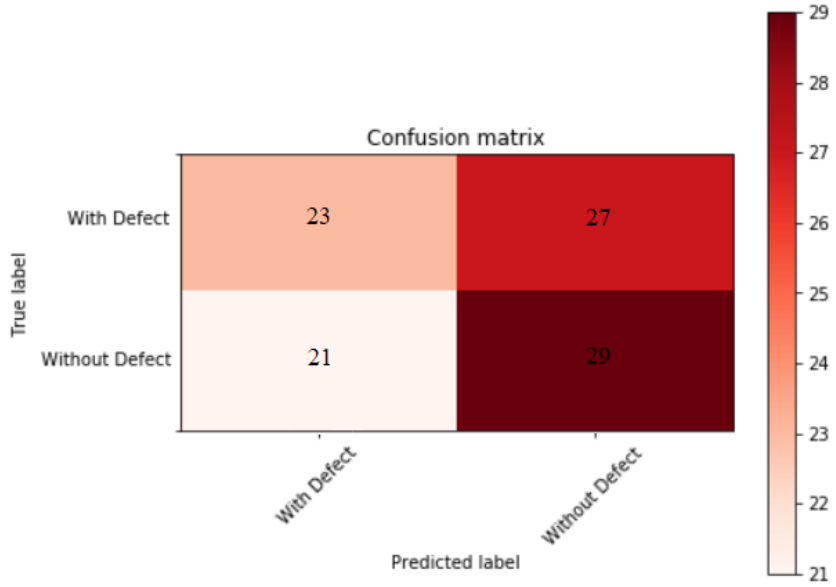

Figure 5.11 – Confusion matrix applied to the entire dish.

	Precision	**Recall**	**F1-Score**
Without Defect	0.52	0.58	0.55
With Defect	0.52	0.46	0.49

Table 5.10 – Precision, Recall and F1-score values applied to the entire dish.

The overall values of precision, recall and F1-score decreased compared to the results obtained when directly classifying a down sampled version of the full dish image. It is thus possible to conclude that for this dataset, dividing the image into smaller segments, classifying

these segments and then classify the entire dish based on the segment classification does not decrease the value of false positives, and even decreases the overall accuracy of the CNN.

It is important to emphasize that the images and results present in this thesis represent a simulated environment. Although real textures of dishes supplied by the factory were used in this realistic dataset, the images are generated in a local context. By the results obtained through these data, it is difficult to say if they will be similar to the results obtained with images of dishes from the production line, since there are not considered factors such as other types of defects, other size of these defects, lighting, ability to capture photographs from cameras, etc. However, it is possible to conclude that this adapted binary classification algorithm acquired knowledge through the generated datasets, and it was optimized considering the problem that was intended to solve, using deep learning.

Chapter 6. Conclusions and Future Work

6.1 Conclusion

The objective of this work was to adapt a deep learning-based algorithm capable of performing a binary classification on dish images, in order to assist the quality control procedures at a porcelain dish factory. This classification consisted of the detection of defects in images provided to the algorithm. When using this algorithm together with high resolution cameras placed directly on the factory's production line, it would be possible to perform quality control in real time, quickly, efficiently, and less susceptible to errors.

The main topics covered in this work were image classification and deep learning, more specifically, Convolutional Neural Networks. The literature review expressed in Chapter 2 explores these two topics, in particular, the way in which deep learning can be applied to image classification, and the excellent results it obtains by doing it.

The COVID-19 pandemic changed the path of this work, since it caused unexpected problems, as explained in Chapter 3. A deep learning based-algorithm needs data to be trained and to be validated, therefore, a dataset is required. In the course of this research work, two datasets were created. The first dataset was preliminary, used during confinement time, which allowed the realization of experiments involving the CNN architecture. The second dataset was later created, also artificial but more realistic. This second dataset was generated based on samples of dishes produced directly at the factory.

Several experiments in Chapter 4. were performed during confinement time. In these experiments performed with the preliminary dataset, different CNN architectures were used. Transfer Learning was also explored in this chapter. The time required for the training process and the accuracy of the different architectures were compared in order to understand if the adapted architecture would pay off in solving this problem, compared to more dense architectures.

After the confinement time and several months in, it was finally possible to receive samples of dishes coming directly from the factory, which allowed the to create the realistic dataset using the textures of the received dishes. This dataset creation used a python script developed in INOV, whose function is to generate artificial dish images, with or without defects, from texture samples. This second created dataset allowed to obtain more realistic data. In addition

to evaluating the classification model previously obtained in Chapter 4. with these new images, it was proposed to divide them in 100x100 parts, so as not to lose much image quality during resize and to try to improve the results. After performing this division of the original images, it was tested whether this method would generate better results compared to the results obtained when classifying the full dish image. The results of using the realistic dataset are shown in Chapter 5. of this research work.

Regarding the first and main objective of this work, although the first dataset created compromised some results due to its lack of variety of defects and lack of initial images and the second had also limitations in terms of image quality loss after resizing the data, the results obtained show that it was in fact possible to adapt a binary classification algorithm based in deep learning to this specific problem. An accuracy of 92.7% with the preliminary dataset and an accuracy of 91.9% with the realistic dataset shows that this adapted algorithm has the capacity to classifying dishes with and without defect.

This dissertation also allows to conclude that, for this binary classification case, despite some Transfer Learning architectures results are slightly higher – in terms of accuracy – than the results of the adapted architecture, their computational cost as well as their training time is also much higher. Therefore, the use of those algorithms is not worth when classifying these industrial defects in dishes.

With the realization of this dissertation, it was also possible to conclude that the proposed method of dividing the original image in smaller parts in order to increase the defect detection as well as reduce the number of false positives was not effective. This division did not improve the recall value of the class "without defect" and even decreased the overall accuracy of the CNN.

6.2 Future Work

As previously said, despite the good results obtained in the experiments performed, it is difficult to know whether this CNN architecture will obtain similar results in a real context. Therefore, the next step will be to install the cameras to obtain images in real context. Once these images are obtained, it will be possible to train and validate the CNN. Once these two steps are done, the last procedure will be to establish whether this CNN is more effective than the factory's current quality control system or not. It will be necessary to retrain the whole CNN with these

new data. Then, accuracy, loss, precision, recall and F1-score values will also be obtain. Finally, it will be possible to modify the CNN parameters in order to improve the previous results. This process can take time as it is not clear how the pandemic will evolve.

While it is not possible to use real images of manufactured dishes to create a dataset, it is possible to try to improve this binary classification model. One possible way to do this is generating more types of defects, which can reach a wider range of real defects in dishes produced in the porcelain dish factory. A solution that could also improve the quality of the dataset would be to resize the images with different dimension reduction algorithms, instead of only using bilinear interpolation.

It is also possible to perform further changes in terms of the CNN architecture, more specifically in the number and size of the kernels used in the convolutional layers, in order to ascertain whether there are significant differences in the obtained results.

Another possible implementation of this classification model would be to use it for quality control in other products or other factories. In this case, it would be necessary to training and validate the model with new data since incremental learning algorithms were not used.

Regarding the false positive results, the author of this research work suggests a measure that might improve this accuracy. The measure is to change the last activation function of the CNN to the sigmoid function. This function is widely used in cases of binary classification since it only exists between 0 and 1. Values closer to 0 are classified in a certain way while values closer to 1 are classified in another. Therefore, by changing the decision threshold, it is possible to modify the probability of the CNN to classify false positives, *a priori*. However, it is necessary to ensure that changing this decision threshold does not negatively influence the rest of the CNN classification.

References

[1] McKinsey & Company, Inc., "Smartening up with Artificial Intelligence (AI) - What's in it for Germany and its Industrial Sector?", April 2017. [Online]. Available: https://www.mckinsey.com/~/media/McKinsey/Industries/Semiconductors/Our%20Insights/Smartening%20up%20with%20artificial%20intelligence/Smartening-up-with-artificial-intelligence.ashx. [Accessed September 2, 2020].

[2] M. Nielsen, *"Neural Networks and Deep Learning"*, 2015, [E-book] Available: neuralnetworksanddeeplearning.com [Accessed December 14, 2019].

[3] I. Goodfellow, Y. Bengio, A. Courville, *"Deep Learning"*, MIT Press, 2016.

[4] A. Zhang, Z. C. Lipton, Mu Li, A. J. Smola, *"Dive into Deep Learning"*, 2019. [E-book] Available: d2l.ai [Accessed December 14, 2019].

[5] W. Rawat, Z. Wang, *"Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review"*, Neural Computation, Sept. 2017.

[6] A. Ng, *"Machine Learning"*, [Online course] Available: coursera, http://coursera.com. [Accessed December 9, 2019].

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*", Journal of Machine Learning Research, vol. 15, June 2014.

[8] Z. Wang, P. Yi, K. Jiang, J. Jiang, Z. Han, T. Lu,, J. Ma, *"Multi-Memory Convolutional Neural Network for Video Super-Resolution"*, IEEE Transactions on Image Processing, vol. 28, No. 5, May. 2019.

[9] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *"Gradient-Based Learning Applied to Document Recognition"*, IEEE Transactions, Nov. 2018.

[10] A. Krizhevsky, I. Sutskever, G. E. Hinton, *"ImageNet Classification with Deep Convolutional Neural Networks"*, in Advances in neural information processing systems 25, Jan. 2012.

[11] K. Simonyan, A. Zisserman, *"Very Deep Convolutional Networks for Large-Scale Image Recognition"*, in ICLR 2015, San Diego, USA, May 7-9, 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *"Going deeper with convolutions"*, in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, Boston, MA, USA, 7-12 June 2015.

[13] P. Langley, *"Elements of Machine Learning"*, Journal of Logic, Language and Information, vol. 7 Issue 1, Jan. 1998.

[14] F. Marini, R. Bucci, A.L. Magri, A.D. Magri, *"Artificial neural networks in chemometrics: History, examples and perspectives",* in Microchemical Journal 88(2), April 2008.

[15] Edwin R. Griff, *"How Neurons Work: An Analogy & Demonstration Using a Sparkler & a Frying Pan"*, in The American Biology Teacher, vol. 68, No. 7, Sep. 2006.

[16] J. Han, C. Moraga, *"The influence of the sigmoid function parameters on the speed of backpropagation learning"*, in Mira J., Sandoval F. (eds) From Natural to Artificial Neural Computation. IWANN 1995. Lecture Notes in Computer Science, vol. 930, Springer, Berlin, Heidelberg, 1995.

[17] S. Marsland, *"Machine Learning An Algorithmic Perspective"*, 2nd edition, R. Herbrich, T. Graepel, 2014, [E-book] Available: https://taylorandfrancis.com [Accessed January 3, 2020].

[18] S. T. Linnainmaa, *"Taylor Expansion of the accumulated rounding error"*. BIT 16, June 1976.

[19] M. Hussain, J.J. Bird, D.R. Faria, *"A Study on CNN Transfer Learning for Image Classification"*, in 18th Annual UK Workshop on Computational Intelligence, UKCI 2018, Nottingham, UK, June, 2018, A. Lofti, H. Bouchachia, A. Gegov, C. Langensiepen, M. McGinnity, Eds. vol. 840, Springer, 2018.

[20] L. Torrey and J. Shavlik, *"Transfer Learning"*, in Handbook of Research on Machine Learning Applications, E. Soria, J. Martin, R. Magdalena, M. Martinez and A. Serrano, IGI Global, Jan. 2009.

[21] X. Li, G. Zhang, H.H. Huang, Z. Wang, W. Zheng, *"Performance Analysis of GPU-Based Convolutional Neural Networks"*, in 45th International Conference on Parallel Processing, ICPP 2016, Philadelphia, PA, USA, IEEE, August 16-19, 2016.

[22] K. He, X. Zhang, S. Ren, J. Sun, *"Deep Residual Learning for Image Recognition"*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, IEEE, June 27-30, 2016.

[23] C.G.P. Suescún, J.O.P. Arenas, R.J. Moreno, *"Detection of Scratches on Cars by Means of CNN and R-CNN"*, International Journal on Advance Science Engineering Information Technology, vol. 9, No. 3, May 2019.

[24] C. Pinto, J. Furukawa, H. Fukai, S. Tamura, *"Classification of Green coffee bean images basec on defect types using convolutional neural network (CNN)"*, in International Conference on Advanced Informatics, Concepts, Theory, and Applications, ICAICTA 2017, Denpasar, Indonesia, IEEE, August 16-18, 2017.

[25] H. Lin, B. Li, X. Wang, Y. Shu, S. Niu, *"Automated defect inspection of LED chip using deep convolutional neural network"*, in Journal of Intelligent Manufacturing, vol. 30, 2019, Oct. 2017.

[26] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei, *"ImageNet: A large-scale hierarchical image database"*, in IEEE Conference on Computer Vision and Pattern Recognition, 2009, Miami, FL, USA, IEEE, June 20-25, 2009.

[27] X. Tao, D. Zhang, W. Ma, X. Liu, D. Xu, *"Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks"*, Aug. 2018, [Online]. Available: https://www.mdpi.com/journal/applsci. [Accessed Dec. 9, 2019].

[28] Z. Fan, Y. Yu, J. Lu, W. Li, *"Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network"*, Feb. 2018, [Online]. Available: https://arxiv.org/abs/1802.02208. [Accessed Dec. 5, 2019].

[29] X. Wang, Z. Hu, *"Grid-based Pavement Crack Analysis Using Deep Learning"*, in 4th International Conference on Transportation Information and Safety, ICTIS 2017, Banff, AB, Canada, IEEE, Aug. 8-10, 2017.

[30] X. Xu, Y. Lei, F. Yang, *"Railway Subgrade Defect Automatic Recognition Method Based on Improved Faster R-CNN"*, in Scientific Programming, vol. 2018, Article ID 4832972, Feb. 2018.

[31] J. Chen, Z. Liu, H. Wang, A. Núñez, Z. Han, *"Automatic Defect Detection of Fasteners on the Catenary Support Device Using Deep Convolutional Neural Network"*, in IEEE Transactions on Instrumentation and Measurement, vol. 67, Issue: 2 , IEEE, Feb. 2018.

[32] A. Birlutiu, A. Burlacu, M. Kadar, D. Onita, *"Defect Detection in Porcelain Industry Based on Deep Learning Techniques"*, in 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Department of Computer Science, West University of Timisoara, Romania, Sept. 21-24, 2017.

[33] C. Gonçalves, "Identificação Automática de Plantas Invasoras em Imagens Aéreas" Master thesis, Telecommunications and Computer Eng., ISCTE-IUL, Lisbon, 2019.

[34] D. P. Kingma, J. Ba, *"Adam: A Method for Stochastic Optimization"*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980. [Accessed May 5, 2020]