U.PORTO

FEP FACULDADE DE ECONOMIA UNIVERSIDADE DO PORTO

MODELLING BANANA'S PERISHABILITY CYCLE WITH COMPUTER VISION

By

José Manuel Marques Vieira Rodrigues

Master Thesis in Modeling, Data Analysis
and Decision Support Systems

Supervised by:

João Manuel Portela da Gama

Nuno Vasco Lopes

Faculdade de Economia

Universidade do Porto

2020

Bibliographical note

José Rodrigues is a finalist of the Master in Modeling, Data Analysis and Decision Support Systems at the University of Porto Faculty of Economics. He received a bachelor's degree in economics from the Instituto Superior de Economia e Gestão at the University of Lisbon in 2010 and a postgraduate degree in Quantitative Methods applied to Economics and Management from the University of Porto Faculty of Economics in 2012. He lives in Madrid since 2016 where he has been working in various projects related to data science. Besides from the field of work and study, he has an interest in World and European history, politics and sports particularly football and chess.

Acknowledgements

In these acknowledgement lines, one is allowed to briefly abandon the scientific spirit of the thesis and roam freely on which forces drove the academic effort, be them subjective or more pragmatic.

I could not start without thanking the Faculdade de Economia do Porto for accepting me back after all these years. Especially the Academic Services and Professor Maria Eduarda da Rocha for the ever kind guidance throughout the process of finding whether I would be eligible to reenter the faculty in a different Master's study. Also, deep gratitude to all the Professors I had in the last twelve months for their comprehension of my "expat" status, especially Professors João Gama, Bruno Veloso and, again, Professor Maria Eduarda.

Also thank you to my employer Accenture, for making it so easy obtaining the thesis theme. It is wonderful to work on something we feel can have immediate applications in the daily work, at the same time attempting to change business paradigms. Special thanks to Jesus Figures and Carolina Redondo for their extreme empathy and availability, Manuela Vaz, Ricardo Pinto and Alexandre Pinto from Accenture Portugal for their kindness at all times and, of course, Nuno Vasco Lopes and Adriano Azevedo from the Digital Transformation CoLab for sharing their wonderful data and asking my annoying questions.

I could not finish without thanking my dearest parents, sisters and Giovanni. Although we are far from each other most of the time, I feel we could not be closer because, as the saying goes, we are where our heart is. Last but not least, my precious Raquel, thank you for your constant encouragement and for putting up with the inevitable time sacrifices. Everything turns to easy mode when you are around.

Abstract

Food retailers continuously struggle with the challenge of how to efficiently manage the stock and rotation of large quantities of perishable food items. This issue takes particular importance when considering the factors leading to the waste of fruits before the act of purchase by the consumer in stores.

The advent of newer and increasingly more sophisticated image recognition deep learning models based on Convolutional Neural Networks(CNN) and its potential applications in real use cases has motivated this study, which uses a dataset of hyperspectral images taken during the eight days of a set of banana´s complete perishability cycle to feed different CNN models. The aim is to attempt to predict the exact moment of degradation of a given banana and this way potentially providing the retailers with an automated and intelligent tool to efficiently monitor the perishables in the store.

Bananas were chosen for this experiment for their high demand in Portugal and Europe and due to the fact that they are not native to the continent requiring import from areas such as South America or Macaronesia with the associated costs that implies for the retailer. The images were gathered by a team from the Digital Transformation CoLab(dTX) using an hyperspectral camera on 204 wave lengths of the magnetic spectre.

This study employs three different supervised CNN architectures on the banana's perishability cycle. These are trained, validated and tested to classify between smaller and bigger moment windows of the cycle. In the first case, the objective is to predict the day of the cycle, in the the latter it is to predict intervals of days. The results are rather satisfactory, with global accuracy results higher than 0.82 and individual class f1 scores ranging between 0.71 and 0.97.

Keywords: Computer Vision, Industry 4.0, Convolutional Neural Networks, Transfer Learning, Image classification, Image processing, Pattern recognition, Spectral Imaging, Spoilage detection

Resumo

Os distribuidores alimentares deparam-se continuamente com o desafio de descobrir maneiras de gerir eficientemente o *stock* e rotação de grandes quantidades de produtos perecíveis. É especialmente relevante o significativo desperdício alimentar que ocorre antes do acto de compra pelo consumidor nas lojas.

O advento de novos e cada vez mais sofisticados modelos de aprendizagem profundo baseados em redes neuronais convolucionais (CNN) e a sua aplicação em contextos reais é o motivante para este estudo, que faz uso de um conjunto de imagens hiperespectrais captadas em laboratório durante um ciclo completo de perecibilidade de bananas para que sirvam de entrada a distintos modelos de CNN. O objectivo é tentar prever o exato momento de degradação de uma dada banana com o intuito de desenvolver um modelo industrializável e usável pelos distribuidores alimentares.

Foram escolhidas bananas para desenvolver esta experiência pela sua alta procura em Portugal e na Europa e pelo facto de não serem espécies de frutas nativas ao continente Europeu, o que requer a sua importação desde áreas como o continente Sul Americano ou ilhas da Macaronésia com os custos adicionais que isso implica para os distribuidores. As imagens foram recolhidas por uma equipa do Digital Transformation CoLab(dTX) usando uma câmara hiperespectral em 204 comprimentos de onda do espectro magnético.

Este estudo emprega três arquiteturas diferentes de modelos supervisados de CNNs para modelar o ciclo de perecibilidade das bananas. Estes modelos foram treinados, validados e testados para classificar entre janelas maiores e mais pequenas do ciclo. No primeiro caso, o objetivo é de prever o dia exato do ciclo de perecibilidade, enquanto no segundo caso o objectivo é pr-

ever intervalos mais alargados de dias. Os resultados são relativamente satisfatórios, tendo-se registado níveis de precisão globais de 82% e medidas f1 variando entre 71% e 97%.

Palavras-chave: Computer Vision, Industry 4.0, Convolutional Neural Networks, Transfer Learning, Image classification, Image processing, Pattern recognition, Spectral Imaging, Spoilage detection

"Faith is to believe what you do not see; the reward of this faith is to see what you believe."

Augustine of Hippo

Contents

Biographical note ii
Acknowledgements iii
Abstract v
Resumo v

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**

1.1 The problem of food waste in the retail sector 1
1.2 The Digital Transformation CoLab 3
1.3 The research problem and objective: modeling banana's perishability cycle . . . 5
1.4 Document outline 6

2 Literature Review **8**

2.1 Computer vision in agriculture and food industry and retail 8
2.2 Performance of Convolutional Neural Networks in image recognition tasks . . . 10
2.3 Usage of computer vision to evaluate food quality 17
2.4 Summary of the literature review: limitations and gaps 19

3 Modelling methodology **20**

3.1 The image data from the dTX experiment 20
3.2 Research hypothesis 21

3.3 Transfer learning with CNNs . . . 26
3.4 CNNs to employ . . . 27
3.5 Summary . . . 28

4 Implementation framework **29**
4.1 Implementation on Google Cloud Platform with Tensorflow . . . 29
4.2 Train, validation and test data . . . 31
4.3 Pre-training activities . . . 32
4.4 Model training and validation . . . 33
4.5 Evaluating methodologies . . . 38
4.6 Summary . . . 40

5 Evaluation and results **41**
5.1 Model specifications . . . 41
5.2 Results of modelling Group 1 with 6 outcomes . . . 42
5.3 Results of modelling Group 2 with 3 outcomes . . . 48
5.4 Statistical comparison of the T1 results . . . 52
5.5 Summary . . . 53

6 Conclusions **54**
6.1 Overview of the efforts . . . 54
6.2 Future work and recommendations . . . 55

Bibliography **56**

A Python source code **i**
A.1 Preprocessing code . . . ii
A.2 CNN modelling code . . . ix

List of Figures

1.1 *Per capita* food losses and waste, at the production/retail and consumption phases [1] 2

1.2 *"dTX works on the intersection of physical, digital and cybernetic domains, with the purpose of creating the next generation of advanced cyber-physical systems capable of blur the boundaries between the real and the virtual world."* [2] 4

1.3 *Per capita* Food losses in fruits and vegetables [1] 5

1.4 *hypercube* obtention with HSI techniques - an example 6

2.1 Illustration of a simplified ANN: The leftmost layer of the network is called the input layer, and the rightmost layer the output layer (which, in this example, has only one node). The middle layer of nodes is called the hidden layer because its values are not observed in the training set. 11

2.2 A CNN: Convolutional layers are inter-weaved with pooling layers. Lastly, standard fully connected layers are introduced for the final classification. 13

2.3 Convolutional layer kernel sliding. 14

2.4 Maximum Pooling layer operating on region of the image. 15

3.1 The objective is to have a model which can accurately tell how many days of freshness are left, given an image. 22

3.2 Division of an image into 4 subsections to obtain isolated images of banana bunches, with the help of python's *PIL* library. 23

3.3 Random selection of images from the 6 Classes from Group 1, described above. The images are directly read as they arrive. Sizes obtained are (436, 320, 3). . . . 24
3.4 Illustration of the TL framework [3]. 27

4.1 A high level research flow of the thesis effort. 31
4.2 Transformations from *ImageDataGenerator* in action. 32
4.3 Target and prediction vectors in ANN, for a 3 class classification problem 34
4.4 The range of cross entropy loss, given a true value in one the classes, for all predicted values 35
4.5 Recommended statistical test for different scenarios, from [4] 39

5.1 ResNet-50 training performance 42
5.2 Xception training performance 43
5.3 Xception+Nasnet training performance 43
5.4 Count of error by wavelength across the 29 badly classified images with Xception+Nasnet 47
5.5 ResNet-50 training performance on Group 2 48
5.6 Xception training performance on Group 2 - here only using 15 epochs instead of the 30 from Group 1 49
5.7 Xception+Nasnet training performance on Group 2 49
5.8 Count of error by wavelength across the 12 badly classified images with Xception 52

List of Tables

2.1 Some of the most famous CNNs's results against the ImageNet database 17

3.1 The image data from the dTX, by moment of capture. 21

3.2 A guide to figure 3.3 25

4.1 Confusion matrix example 38

5.1 Confusion matrix for ResNet-50 44

5.2 ResNet-50's Precision, Recall and F1-Scores per class 44

5.3 Confusion matrix for Xception 45

5.4 Xception's Precision, Recall and F1-Scores per class 45

5.5 Confusion matrix for Xception+Nasnet 46

5.6 Xception+Nasnet's Precision, Recall and F1-Scores per class 46

5.7 Accuracy and deviation of the predicitons for Group 1 across the three models . 47

5.8 Confusion matrix for ResNet-50 in Group 2 50

5.9 ResNet-5's Precision, Recall and F1-Scores per class in Group 2 50

5.10 Confusion matrix for Xception in Group 2 50

5.11 Xception's Precision, Recall and F1-Scores per class in Group 2 51

5.12 Confusion matrix for Xception+Nasnet in Group 2 51

5.13 Xception+Nasnet's Precision, Recall and F1-Scores per class in Group 2 51

5.14 Accuracy and deviation of the predictions for Group 2 across the three models . 52

5.15 Global T1 Accuracy results for all three models across the two problems. Rankings for Friedman test in parenthesis 53

Abbreviations

dTX	Digital Transformation CoLab
CNN	Convolutional Neural Networks
HSI	Hyperspectral Imaging
AI	Artificial Intelligence
CV	Computer vision
ML	Machine Learning
DL	Deep Learning
FAO	Food and Agriculture Organization
FSC	food supply chain
SVM	Support Vector Machines
ANN	Artifical Neural Networks
RELU	Rectified Linear Activation Function
BPP	Back propagation
GCP	Google Cloud Platform
TL	Transfer learning

Chapter 1

Introduction

This chapter introduces the general spirit of the thesis, starting by providing an overview on the problem of food waste in retail and its extent, motives and ramifications. Then a presentation of the research problem and objectives will be provided, with a description of the data that will be worked on, the techniques to employ and the potential limitations of the study. Finally, a complete document outline will be provided to serve as guide for the reader.

1.1 The problem of food waste in the retail sector

According to a 2011 report from the Food and Agriculture Organization (FAO) [1], roughly one third of the of all yearly production of food for human consumption is lost or wasted, amounting to 1.3 billion tons per year.

In Europe, the same study concludes that the per capita food loss is estimated to be 280 Kg per year and that 68% of it occurs before reaching the final consumer. In a world witnessing an accelerated global population growth and an ever presence consciousness in the societies of the limits of the available natural resources, managing the food waste is an increasingly clear priority for most actors since there are obvious and significant economic, health and environmental gains to be obtained for every reduction of food loss.

It is possible to generally isolate food loss in the food supply chain in five distinct moments:

1. **Losses in Production**: losses due to mechanical damage and/or spillage of products

during harvest;

2. **Losses in Post-harvest**: losses due to degradation and/or spillage during handling, storage and transportation;

3. **Losses in Processing**: losses due to degradation and/or spillage during industrial or domestic processing;

4. **Losses in Distribution**: losses occurring in the market system, e.g. wholesale markets, supermarkets, retail and wet markets;

5. **Losses in Consumption**: losses and waste during consumption at the household level.

Figure 1.1 illustrates the *per capita* food loss between Production to Retail (the three first phases mentioned above) and the Comsumption phase mentioned above.

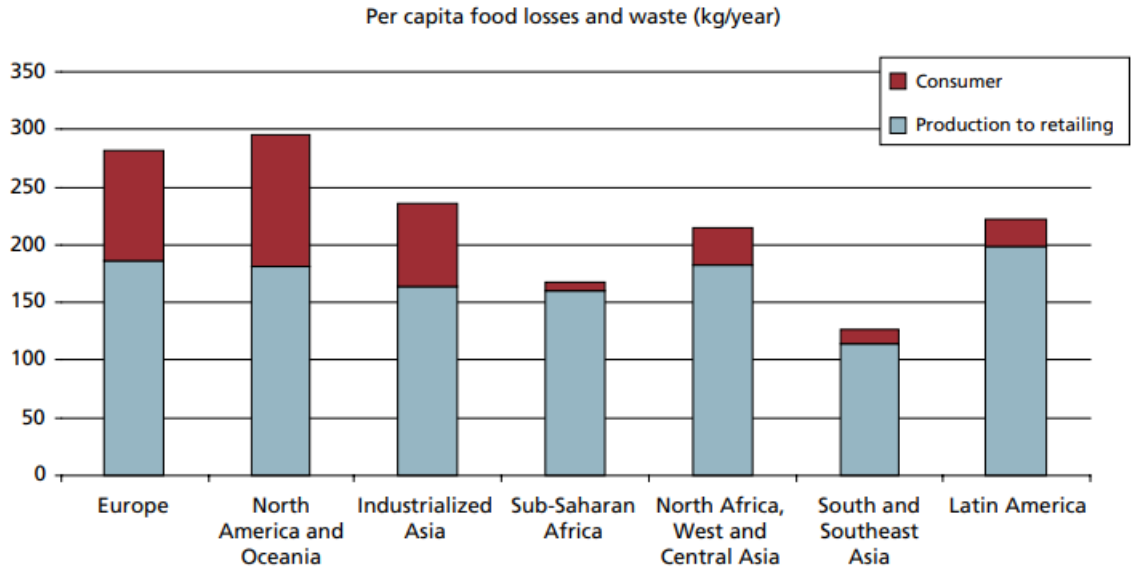


Figure 1.1: *Per capita* food losses and waste, at the production/retail and consumption phases [1]

Food losses in developing countries are as high as in industrialized countries. However, the data shows that the proportion of waste occurring in retail and consumption phases of the food supply chain is much higher in industrialized richer countries. In the developing areas of the world the waste mostly occurs in the pre-retail phases.

Still according to the report from FAO [1], the causes to this high waste in retail and consumption phases in industrialized countries derive from the fact that this market is characterized by a chronic case of the supply exceeding the demand which many times leads to a relaxation in the handling of this excess abundant food. Another cause of food waste in the retail phase is occurs from malpractices in display, especially in the case of fresh food like fruits and vegetables. Often excessively large quantities of these products are put on display to meet the consumer expectation of a *full shelf*. When piled all together in excessive numbers the life cycle of these types of products is accelerated leading to increased food waste.

1.2 The Digital Transformation CoLab

The Digital Transformation CoLab (dTX) [2] is a non-profit private association between three Portuguese universities (University of Minho, University of Évora and Universidade Católica Portuguesa), innovation hubs and thirteen companies that carries out its activity doing applied research in different areas linked to digital transformation in industry and society.

dTX's aim is to deploy projects in manufacturing, retail, banking and other sectors with a goal to achieve a complete integration between the physical, digital and cybernetic domains. By utilizing tools such as advanced analytics, internet of things or big data, the objective of a typical dTX project is to deploy networked systems which are a copy and work with the physical world in order to increase productivity in industry and society through gains in efficiency, resiliency and cooperation.

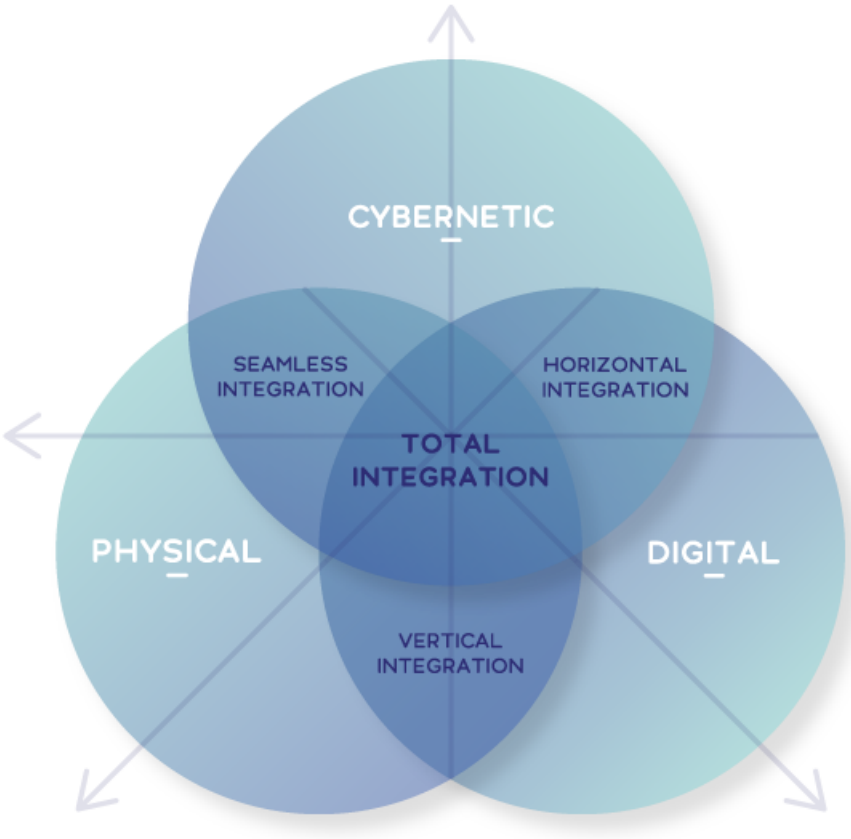


Figure 1.2: *"dTX works on the intersection of physical, digital and cybernetic domains, with the purpose of creating the next generation of advanced cyber-physical systems capable of blur the boundaries between the real and the virtual world."* [2]

The seamless transmission of data between physical and the cyber/digital worlds gives origin to what is often called the Digital Twin [5]; it can be seen as a digital replica of the physical world which emulates its behaviours, learns from its errors and rewards efficiency in order to optimize productivity in the production processes. Physical objects and twin models interact in a mutually beneficial manner.

dTX's projects are therefore driven with the objective of bringing innovative solutions to its partners, helping them to partake in the new widespread industry transformation many times called Industry 4.0 [6].

1.3 The research problem and objective: modeling banana's perishability cycle

Returning to [1], it is estimated that the food waste in the distribution phase consumer in Europe is as high as 10% in the cases of fruits and vegetables.

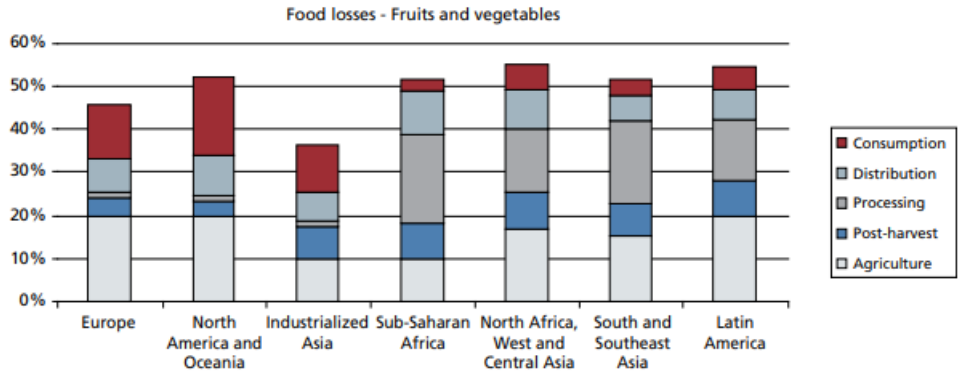


Figure 1.3: *Per capita* Food losses in fruits and vegetables [1]

According to [7], 55% of the spoilage of fruits and vegetables in developed countries are due to poor environmental (air and temperature) conditions during display which favour the progress of biochemical microbiological processes which, in turn, act quickly on rotting the produce [8].

In a cooperation with Accenture Portugal, dTX has used cameras and sensors to gather data in its laboratories on the complete perishability cycles of bananas. Bananas were chosen due to its high economic and environmental cost of transportation to the country as a result of not being a natively grown fruit in continental Europe.

The collected data has been of two types:

1. **Quantitative information**: on environmental indicators such as temperature, humidity, O2/CO2 concentration, ethylene among others;

2. **Image data**: on the perishability cycles collected by hyperspectral imaging (HSI). This is a rapid, non-destructive and chemical free method that has been emerging as a powerful analytical tool for product inspecting because it simultaneously offers spatial information and spectral signs from the object [9].

Optical measurement is contactless and requires no human intervention hence allowing for objects to be inspected without danger of being damaged or contaminated. Image data is processed in real-time and gives immediate results to systems or operators to act upon.

Figure 1.4 illustrates how HSI works: the data structure of hyperspectral image is commonly called hypercube or data cube and can be viewed as a set of spectrums of each pixel in one two-dimensional image clustered together [9].



Figure 1.4: *hypercube* obtention with HSI techniques - an example

HSI images will be the source of data used in the CNN models developed for this document efforts which will be described in chapter 3. In February 2020, image data regarding a complete perishability cycle on 204 wave lengths of the magnetic spectre has been gathered amounting to 140 Mb of information. Also in chapter 3, a complete assessment of the data will be provided as well as its the potential and possible limitations.

The objective is, therefore, to build models which are capable of, given any unlabeled image of bananas, accurately predict the moment of the perishability cycle.

1.4 Document outline

The remaining chapters of this thesis will be laid out as follows:

1. **Chapter 2 (Literature Review)**: review of the main literature in the problematic of employing image processing techniques to model the perishability cycle of fresh food, especially fruits. The review will also focus on listing the main areas of advancement of the

CNN models and identifying the past existing work related to the usage of hyperspectral images together with CNNs.

2. **Chapter 3 (Modelling methodology)**: deep dive into the research practical objective, considering the available data and models to implement.

3. **Chapter 4 (Implementation framework)**: Cloud technology to employ, deep learning framework and tactics for assessment the robustness of the model's predictive power;

4. **Chapter 5 (Evaluation and results)**: Practical implementation of the models and evaluation of the results according to various statistical measures like accuracy, the classes precision and recall and F1 scores;

5. **Chapter 6 (Final conclusions)**: discussion of the results obtained and how they could contribute to help in real world situations. Also considerations about possible improvements to the developed models will be presented.

Chapter 2

Literature Review

This chapter will review the literature work regarding the methodologies employed in computer vision in agriculture and distribution of fresh food. A review of the later advances in the field of CNN will be laid out and, finally, with a particular emphasis on studies which make use of HSI imaging. Also a review of the later advances in the field of CNN will be laid out. In order to facilitate the read, the review will be subdivided in three sections:

1. **Computer vision in agriculture and food industry and retail**;
2. **Performance of Convolutional Neural Networks in image recognition tasks**.
3. **Usage of computer vision to evaluate food quality**;

Finally, in section 2.4, a brief summary of the contents of this chapter will be presented.

2.1 Computer vision in agriculture and food industry and retail

The task of detecting spoilage in food is fundamental for producers, retailers and consumers. The process of manually inspecting the quality of each aliment can be a time consuming task, given the quantity of units to check, the high error rate and the risk of damaging the produce which is inherent to it.

The field of computer vision emerged in the late 1960s and has since been abundantly applied in the aerospace industry, industrial automation, security inspections, intelligent transportation systems, medical imaging, military, robot guidance, autonomous vehicles, safety inspection. Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do. This is achieved by working a labeled image database, converting all the images to arrays of numbers representing each of the pixels in it and then subject them to various software techniques or algorithms which allow the computer to hunt down patterns in all the elements that relate to those labels.

These algorithms can be founded in Machine Learning principles where a set of "features" or smaller applications that can detect specific patterns in images are programmed and then a statistical learning algorithm such as linear regression, logistic regression, clustering algorithms, decision trees or support vector machines (SVM) is deployed to detect patterns, classify images and detect objects in them. An example of computer vision with a machine learning approach is the scale-invariant feature transform (SIFT) [10]. SIFT uses key points of objects and stores them in a database. When categorizing an image, SIFT checks the key points of the image, which matches those found in the database. Deep learning (DL) provided a different approach to doing computer vision with machine learning. In rigour, Deep Learning is an extension of Machine Learning whereas it seeks to add depth or increased complexity into a model. For the computer vision tasks, DL algorithms don´t specifically need defined image features, they are capable of learning by themselves with the help of neural networks during training without the need for any manual intervention besides labeling. DL techniques applied to Computer vision have been gaining much attention from the business and academic world since Krizhevsky et al [11] developed AlexNet in 2012 which obtained groundbreaking results in classifying images from the huge ImageNet database [12]. The training of these models, despite not requiring much human intervention, still requires quite a substantial amount of computer power; the latest advances in cloud technology and computer hardware capabilities together the increasing availability of image data have provided fertile ground to more and more advances in the field. Section 2.2 will provide a review of the advances of Convolutional Neural Networks which is the main DL architecture in

Computer Vision.

In the field of agriculture, computer vision is often used to tackle problems such as land identification [13], recognition of disease infected plantations based on shape, texture and color [14] [14], detection and classification of fruits and vegetables [15] and expected crop yields [16] and, more to the interest of this study, the grading and evaluation of food quality. Section 2.3 will contain a review of the main lines of work on the usage of computer vision with ML or DL techniques to assess the quality of fresh fruit or vegetables.

2.2 Performance of Convolutional Neural Networks in image recognition tasks

As mentioned in the previous section, Deep learning (DL) can be seen as an extension of Machine learning (ML). DL relies heavily on the principles of Artificial Neural Networks. A frequent analogy of these ANN based DL models is made with the functioning of the human brain. A standard neural network consists of many simple connected processors called neurons, each producing a sequence of real-valued non-linear transformation, i.e., activations. There is a wide variety of activation functions like sigmoid , tanh, Softmax or ReLU [17] each being more efficient than the others according to the problem in question or the type of data beeing used. Input neurons get activated through sensors perceiving the environment, other neurons get activated through weighted connections from previously active neurons. Some neurons may influence the environment by triggering actions [18].

An ANN is always formed by layers, these are nothing but collections of neurons which take in an input and provide an output. Inputs to each of these neurons are processed through the activation functions assigned to the neurons. Any ANN has at least one input and an output layer. The number of hidden layers varies between different networks depending upon the complexity of the problem to be solved. Each of the hidden layers can have a different activation function, again depending on the type of problem and/or data in question.

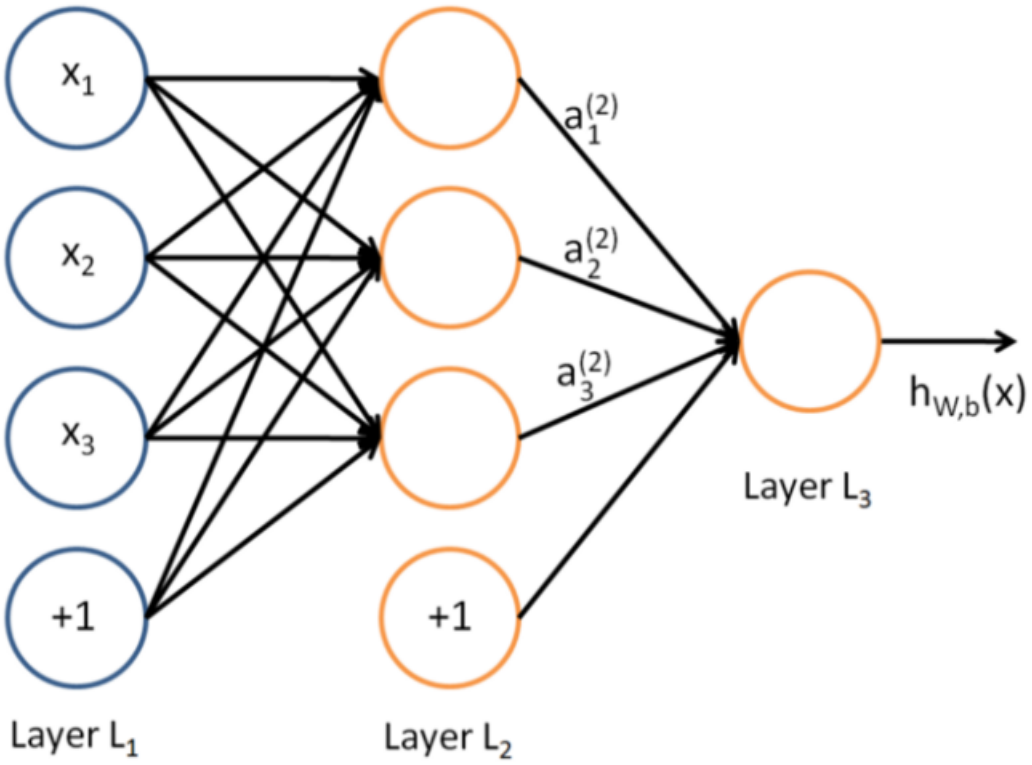


Figure 2.1: Illustration of a simplified ANN: The leftmost layer of the network is called the input layer, and the rightmost layer the output layer (which, in this example, has only one node). The middle layer of nodes is called the hidden layer because its values are not observed in the training set.

The training of these ANN models follows the principles of forward and back propagation, they are the essence of neural network training. The network weights between neutrons and layers is randomly initialized and the neurons get to work on their activation functions and produce the final output *h(x)*, this is called forward propagation within an epoch, i.e., a full passage of the training data within the ANN. Since we are in the training phase, we know the extent of the error rate between *h(x)* and the actual value (i.e. loss). This loss is then back propagated across the ANN using optimization algorithms such as Adam [19], RMSprop [20] or SGD [21]. The back propagation is an optimization problem because the ANN is aiming to minimize the training loss obtained in previous iterations by recalculating of the weights of each neuron. The ”velocity” of the optimization process can, in the case of most available optimisers, be regulated by the

learning rate. It defines how quickly a network updates its parameters. Low learning rate slows down the learning process but converges smoothly to the minimum loss. Larger learning rate speeds up the learning but may not converge to the minimum loss. The velocity of learning (i.e. learning rate), the number of iterations or the number of layers of the ANN are all considered to be hyperparameters. These are variables which determine the network structure and how the network is trained and are set before training. Section 3.4 will describe the CNN models employed in this study and provide details on some of these hyperparameters.

Convolutional Neural Networks (CNN) is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions referenced above, convolution and pooling functions are used as activation functions. CNNs are different from ordinary ANN in the sense that neurons in the convolutional layers are only scarcely connected to the neurons in the next layer – each hidden activation is computed by multiplying a small local input against the weights. The weights are then shared across the entire input space. Neurons that belong to the same layer share the same weights. Weight sharing is a critical principle in CNNs since it helps in reducing the total number of trainable parameters and leads to more efficient training and more effective image modelling.

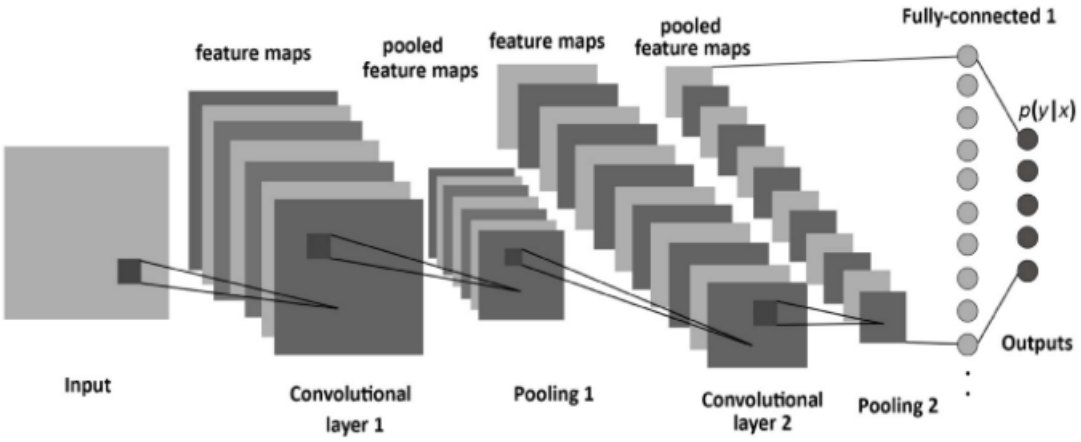


Figure 2.2: A CNN: Convolutional layers are inter-weaved with pooling layers. Lastly, standard fully connected layers are introduced for the final classification.

Convolution operates on two signals (in 1D) or two images (in 2D): one can think of 1D as the "input" signal (or image), and 2D (called the kernel) as a "filter" on the input image, producing an output image (so convolution takes two images as input and produces a third as output). In the case of computer vision, it is easier to visualize a kernel as sliding over an entire image and changing the value of each pixel in the process [22].

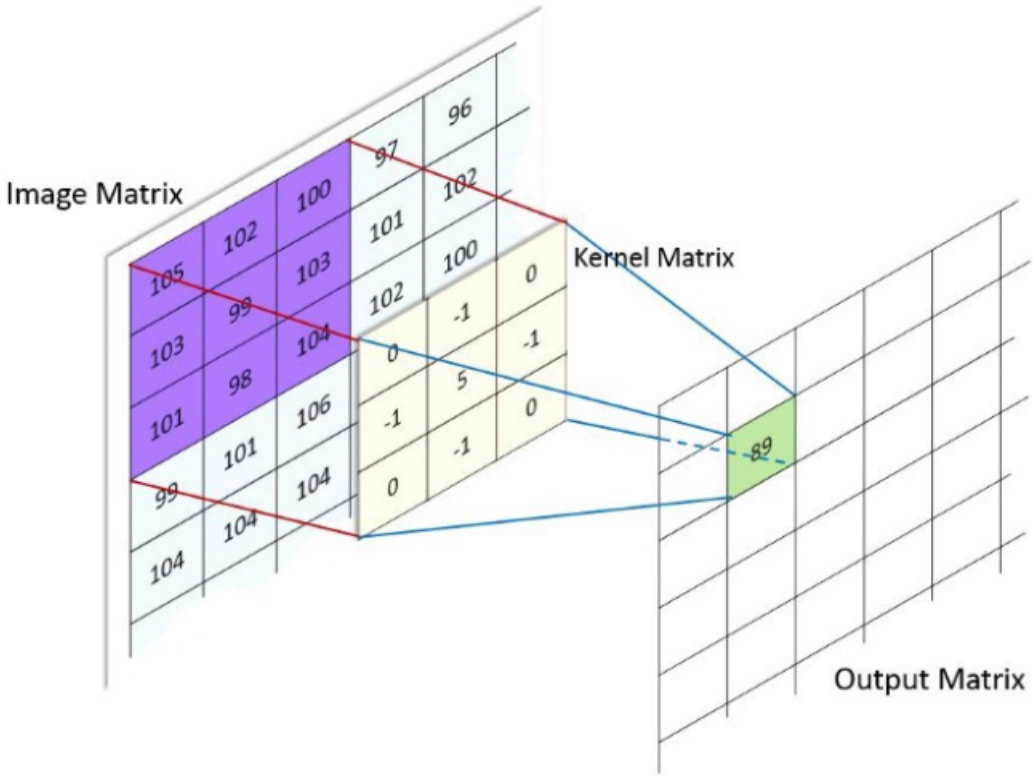


Figure 2.3: Convolutional layer kernel sliding.

The feature maps produced in the convolutional layers are fed as inputs to the pooling layers. Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions. There are 2 main types of pooling commonly known as max and min pooling. As the name suggests max pooling is based on picking up the maximum value from the selected region and min pooling is based on picking up the minimum value from the selected region.

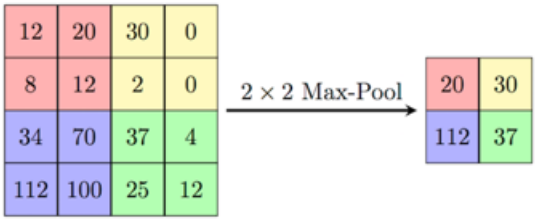


Figure 2.4: Maximum Pooling layer operating on region of the image.

The final task of the CNN is the classification of the images. Here, a traditional final fully connected multi-layer perceptron neural network takes a flattened vectorized form of the features identified with the convolutional and pooling layers and uses it to make a prediction about the input image's class.

Important to mention that these type of models with large numbers of interconnected parameters often tend to incur in the problem of overfitting, which happens when the training and pattern recognition process excessively fixates in specific characteristics of the input data used for training which may not be generalizable to the testing data. In order to deal with situations of overfitting, in 2014 Hinton et al [23] came up with a simple idea: randomly drop units (along with their connections) from the neural network during the iterations of training. This way, if there is an over-adaptness of the model to a specific feature of the data, this feature will randomly be "shut down" along the iterations.

The recent years have witnessed the birth of numerous CNN architectures. Despite this, the first to stack convolutions and average pooling layers and ending the network with one or more fully-connected layers was LesNet-5 in 1998 [24]. It had 60 thousand parameters.

In 2012, AlexNet [11] achieved groundbreaking results against the ImageNet database. This model can be said to be LesNet-5 stacked with a few more layers but was the first to implement ReLu as activation functions. It had 60 million parameters.

Due to the continuous improvements in hardware and computation evolution, CNNs were free to keep growing in layers and getting deeper. In 2014, the Visual Geometry Group from the University of Oxford created VGG-16 [25] which took the ReLu activation functions from

AlexNet with more layers but smaller filters on the convolutional layers for better classification performances. VGG-16 grew to 138 million parameters which was twice as deep as AlexNet.

Google's Inception-v1 [26] introduced a novelty in the CNN architectures: instead of stacking convolutional and pooling layers in a "vertical" manner, it has parallel towers of convolutions with different filters which are later concatenated, this way capturing different features at 1×1, 3×3 and 5×5, thereby "clustering" them. This parallel layer approach resulted in a significantly lighter network of around 5 million parameters. A later version of Inception-v1, Inception-v3 was developed in 2015 [27] with 24 million parameters which incorporates some changes to the original optimiser and loss function and also introduces batch normalization for the first time [28]. Batch normalization tackles the problem of having distinct input sizes (in height, width or depth) in the batches of data that go into the hidden layers of the CNN by constantly scaling and reducing the amount of variability of the weights of the network of during training due to this eventual heterogeneity in some of the training data. In 2016, Google's team published yet another version of Inception, Inception-v4 [29]. In this version the original architecture is maintained with increased depth in layers (56 million parameters) and minor improvements in relation to Inception-v3.

In 2015, Microsoft Research published their own CNN architecture ResNet-50 [30]. This was a very deep network (152 layers) but similar in size to Google´s Inception-v3 (26 million parameters). It was also one the first to use batch normalization.

Google's AI lead AI researcher François Chollet introduced Xception architecture in 2016 [31]. The original assumptions from the Inception family of models is taken to an extreme: having multiple cross channel feature maps beeing captured in parallel and then finding correlations between them via the normal 3x3 and 5x5 convolutions. In Xception, this tactic is applied to every possible channel.

In 2017, a team from Google Brain developed the NasNet architecture [32] applying AutoML principles, where thousands of different architectures are trained and evaluated for a particular problem. In this approach, the final architecture was automatically and iteratively searched based on the accuracy in training (Neural Architecture Search (NAS)). This approach is different from others where there is a general human pre-thought architecture. NasNet was developed on a small

academic image dataset (CIFAR-10) and subsequently transferred with minor changes to the very large ImageNet, registering excellent results. A mobile version of NasNet was also developed, designed to operate with limited computational resources, also with good results.

These family of models are being continuously improved with newer versions. Below a summary of some of the results obtained by these models against the ImageNet database:

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 Mb	0.713	0.901	138 Million	23 layers
Inception-v3	92 Mb	0.779	0.937	23 Million	159 layers
Resnet-50	98 Mb	0.749	0.921	25 Million	152 layers
Xception	88 Mb	0.790	0.945	22 Million	15 layers
Nasnet	88 Mb	0.827	0.962	89 Million	1042 layers

Table 2.1: Some of the most famous CNNs's results against the ImageNet database

In Figure 2.1, some of the most famous CNN architecture's performance against ImageNet is displayed. "Top-1Accuracy" and "Top-5Accuracy" refer to correctly classifying the output with the highest probability and getting the correct actual value in one of the first five predictions.

2.3 Usage of computer vision to evaluate food quality

Although it is hard to find related work on the modelling of bananas's perishability cycles, there is abundant work in the field of the development of image recognition and classification of fruits and vegetables with computer vision, mainly in initiatives to improve efficiency in the agricultural sector. In [33], a very extensive review of the usage of ML and DL methods for preprocessing, segmentation, feature extraction and classification to address fruits and vegetables quality based on color, texture, size, shape and defects.

In [34], a method for recognizing and counting green and red peppers in a greenhouse is presented. In a first step, the fruits were located in a single image and in a second step multiple views were combined to increase the detection rate of the fruits. The approach to find the pepper fruits in a single image is based on a combination of finding points of interest, applying a complex high-dimensional feature descriptor of a patch around the point of interest and finally using a

so-called bag-of-words model for classifying the patch. The results were correlation of 74.2% between automatic and manual counts of fruit numbers and 94.6% in identifying the fruits.

In [15], an approach to train a faster region-based CNN to be used by a robot harvesting sweet pepper by automatically detecting them in a plantation is presented. The model is trained using RGB (Red, Green and Blue) and NIR (near infra-red) images. The accuracy results were 83,8%.

In [35], the same faster region-based CNN technique used in [15] is employed but this time to detect apples and mangoes in images collected outside where a lot of variance in luminosity, fruit size, clustering and view-points greatly difficult the task. The accuracy results were higher that 90%.

A study from 2017 [36] for detecting the diseases that affects a rice plant was developed. Images from healthy and diseased rice leaves and stem were captured. And fed into a CNN to train a model. Different hyperparameters were tried and tested in the CNN to obtain optimal prediction accuracy. The model was able to successfully classify ten rice diseases associated with the rice plants. CNNs were concluded to offer better training performances and convergence rates when compared to other models.

In [37], a computer vision analysis on packaged fresh-cut lettuce is proposed. A deep learning architecture based on convolutional neural network (CNN) was used to identify vegetable with minimum colour distortions. The results showed that the performance loss due to the presence of packaging is negligible (83% instead of 86% with packaging) and that the proposed system can be used to monitor the quality level of fresh-cut lettuce regardless of packaging at all the critical check points along the supply chain.

In a Master dissertation with an effort very similar to the one presented in this document, CNNs were employed to detect spoilage in raspberry fruit using spectral imaging [38] and obtained accuracy of around 86% on good vs bad results although they yielded not so good results in predicting the number of days before rotting.

The works cited above all include usage of CNNs to grade the quality or recognize fruits and vegetables. There is no literature available on the usage of CNNs in tandem with HSI imaging. Still, some related works of applying other ML techniques to analyse the quality of fruits will be

provided below.

In [39], principal component analysis was applied to select the four best wavebands for a multi-spectral imaging test. The objective was to identify the best waveband images of bruised and non bruised golden apples in order to submit them to a classification algorithm based on moments thresholding. 93% of the non bruised were identified while 86% of the bruised were correctly classified.

In [40], spatial feature extraction from hyperspectral images is developed to distinguish defective from normal oranges with 98.9% accuracy.

In [41], the authors apply multiplicative scatter correction to preprocess the hyperspectral images of apples followed by partial least squares to extract firmness and sugar content feature wavelengths of the apples. The perform the predictions, ANN, partial least square regression and principal components regression were applied, yielding satisfactory results.

2.4 Summary of the literature review: limitations and gaps

This chapter presented an extensive review on the applications of computer vision in agriculture and the food supply chain industry. Special emphasis was then put on the origins, architecture and recent advancements of CNNs, as these will be the models applied in the practical efforts of this dissertation. Chapter 3 will contain a description of the types of CNNs that will be applied to analyse the data at hand.

The sub-field of computer vision to analyse food quality has seen quite a lot of work. There is an abundance of available literature describing the usage of various ML methods such as clustering, regression or deep learning to identify, count or distinguish fruits, vegetables or diseases in aliments. However, what does lack in the literature are works containing the application of CNNs in tandem with HSI imaging of any type of fruits.

In the next chapter, the data used in the dissertation and the CNN models that will be tried will be presented.

Chapter 3

Modelling methodology

This chapter will provide details about the data and models to implement in order to model the banana´s perishability cycle.

3.1 The image data from the dTX experiment

In February 2020, a team from the dTX [2] has developed an experience in laboratory. Five pairs of bananas were photographed two times a day starting the 4th of February and ending the 12th. During this 8 day period, the image acquisitions were done across 204 magnetic spectral channels, or wavelengths. So, in each of the acquisitions, 204 different photos were obtained from the same objects but with different appearances, amounting for a total of 2448 images. The 204 channels vary from a wavelength of 397.32 all the way to 1003.58 incrementing by a constant of 3. The environment where the bananas stood during the experiment attempted to emulate the conditions one would find of bananas in a supermarket, i.e., with no refrigerating.

Day of capture	Moment	Number	Day of the cycle	Hours since previous
2020-02-04	11:02:26	204	1	
2020-02-04	18:38:49	204	1	7.6
2020-02-05	11:03:58	204	2	16.4
2020-02-05	18:53:55	204	2	7.8
2020-02-06	11:15:52	204	3	16.4
2020-02-06	17:47:29	204	3	6.5
2020-02-07	11:01:40	204	4	17.2
2020-02-07	17:59:01	204	4	7
2020-02-10	11:38:26	204	7	65.7
2020-02-10	16:56:04	204	7	5.3
2020-02-11	11:01:54	204	8	18.1
2020-02-12	10:26:50	204	9	23.4

Table 3.1: The image data from the dTX, by moment of capture.

In the first four days of the perishability process, two daily photos were taken. There are almost three days of interval between the fourth and seventh day. In the eight and ninth day of the process only a single photo was taken in the morning. Each of the photos were individually tagged with the moment of capture and the corresponding wave length.

Although the images differ due to the distinct wavelengths, it is true that the position of the bananas within the photos is always the same, which can very well lead to problems in the training of the models. To account for this, sampling techniques like image rotating, flipping, or mirroring operations will be applied in modelling of the CNNs. These will be further discussed in section 4.3 from chapter 4.

3.2 Research hypothesis

The thesis objective will therefore be to, by making use of the image data mentioned in the previous section, develop a predictive model which would be able to, given a certain image of bananas captured in real time by the same sensor camera, identify the moments of perishability and thus predicting the remaining freshness time.

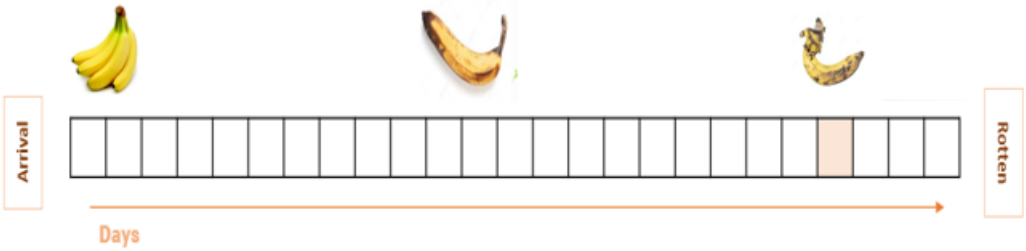


Figure 3.1: The objective is to have a model which can accurately tell how many days of freshness are left, given an image.

Given the characteristics of the experiment by the team from the dTX and the intervals of hours between the captures shown in Table 3.1, the CNN models will be programmed to predict two groups of outcomes:

1. **Group 1 with 6 outcomes**: Predict between:
    1.1. Day 1;
    1.2. Day 2;
    1.3. Day 3;
    1.4. Day 4;
    1.5. Day 7;
    1.6. Day 8 to day 9.

2. **Group 2 with 3 outcomes**. Predict between:
    2.1. Day 1 to day 2;
    2.2. Day 3 to day 4;
    2.3. Day 7 to day 9.

It was put into consideration, prior to the model development phase, a split of each image data into four smaller pieces during the data preprocessing phase, taking advantage of the fact that the experiment yielded records of banana bunches in the same positions. These smaller images could be treated as individual input to the CNN models:

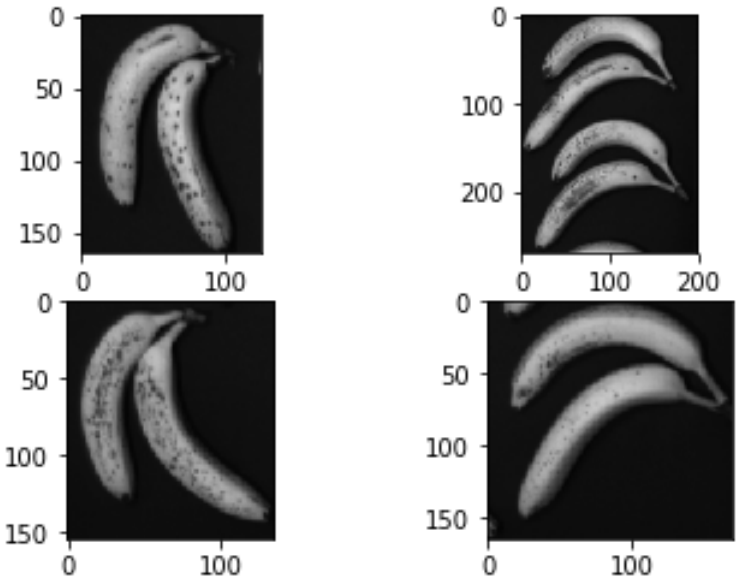


Figure 3.2: Division of an image into 4 subsections to obtain isolated images of banana bunches, with the help of python's *PIL* library.

This possibility was finally not brought into the development logic of the models exposed in this document because it would contradict the logic of the data gathering by the dTX's experiment which is assuming that the sensors would be collecting images in a warehouse or supermarket display of groups of bananas which are in a similar moment of the perishability process. Nonetheless, given the practical spirit of the research and the possibility of implementation in the market, the code was developed with the mindset of, everywhere possible, having the possibility of easily manipulating the research objective.

When an image file is processed by a computer program, it is treated as a three dimensional object of Height, Width and Number of Channels. Height represents the number of pixel rows in the image or the number of pixels in each column of the image array. Width is the number of pixel columns in the image or the number of pixels in each row of the image array. Number of channels are the number of components used to represent each pixel, a number of channels 3 would mean that the picture is represented by red, green and blue channels.

Below, a sample of the image data to be used in the modelling.

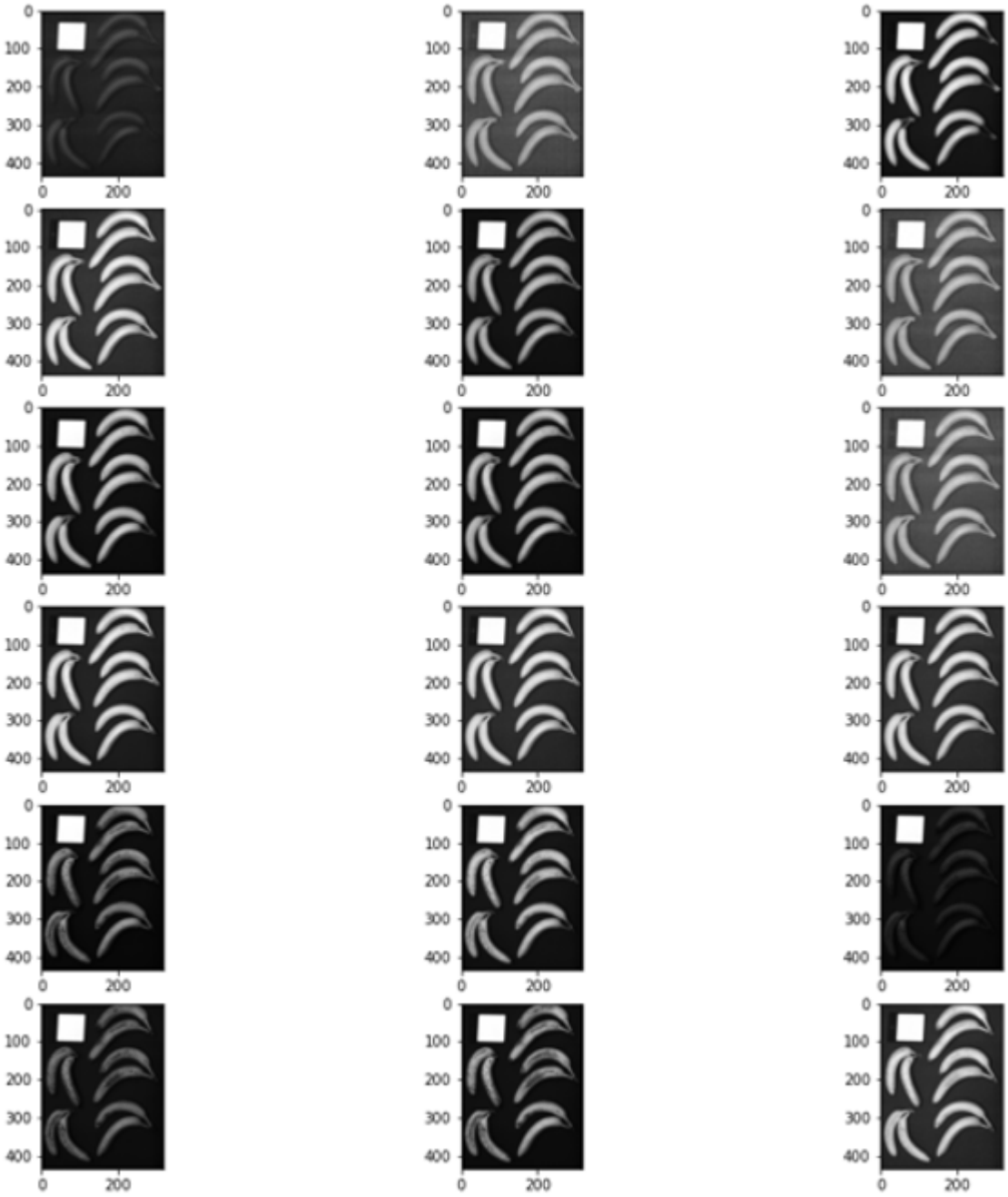


Figure 3.3: Random selection of images from the 6 Classes from Group 1, described above. The images are directly read as they arrive. Sizes obtained are (436, 320, 3).

Day of cycle	Wavelength sample 1	Wavelength sample 2	Wavelength sample 3
Day 1	423.29	997.4	690.64
Day 2	911.3	563.24	997.4
Day 3	631.15	601.55	1000.49
Day 4	768.6	877.69	905.18
Day 7	548.55	660.84	492.97
Day 8 to 9	528.03	622.26	886.84

Table 3.2: A guide to figure 3.3

In all pictures, there is a light coloured rectangle in the top left corner. This is the calibration block used to help adjust the wavelength during the experiment. There was also the option of segmenting the images in order to remove it from all of them, but it was decided to maintain it with the objective of using it as a proxy to an external object not related to bananas, which can likely appear in a real life scenario where the images are beeing gathered in a supermarket store or in warehouse.

The success of applying this image recognition supervised model would, if set up facing a banana's display present in a supermarket would enable quicker and more automated decision making. Given a certain input from the predictive model, measures could be taken like:

1. Take preservation measures in order to prolong the freshness of the product in display, i. e., readjusting environment conditions like the temperature, spraying the products with water or repackaging the products;

2. Decide on removing the products from display and redirecting them to other purposes like becoming ingredients for other food processing (juices, soups, etc.) or charity donations; while at the same time liberating space for fresher products stored in warehouse to come to display;

3. Adjust the price at which the product is being offered to the customer thus allowing less fresh products to find its demand at alternative prices.

In the limit, this monitoring system and the constant feedback it would give to the store staff could be integrated in a wider smart surveillance system; the data coming from cameras pointing

to the products in display would constantly be used by the trained predictive model to forecast the remaining days of freshness. These forecasts would in turn be used by other eventually automatized and programmed systems to decide on actions to take like the ones listed above (e.g., automatically activate water sprays over the produce for some seconds, adjust the price per kilo of the items in an electronic display). Revisiting the topics discussed in chapter 1, these in-store smart monitoring systems would be examples of Digital Twin systems maximizing efficiency and productivity gains from the integration of the physical, digital and cybernetic domains.

3.3 Transfer learning with CNNs

Training a CNN requires a large number of sample to learn the huge number of parameters. However, as shown in section 3.1, the amount and variety of the available data is not abundant. Researchers trying to develop similar architectures against HSI data often face the same challenge since the collection of these type of data is often financially expensive and/or time demanding. This issues may very well result in poor classification performance [3].

Transfer learning [42] is the technique of using some knowledge and parameters of previously defined models and apply them to a new problem. TL comes in handy especially when the training available data is scarce and difficult to obtain.

In CNNs, TL is a process in which low and middle layers from the networks usually capture generic features of input images which have high generalization for other images making it is feasible to directly transfer the network parameters of low and middle layers to a new network which has the same architecture as the previous network. The idea behind transfer learning is to reuse the convolutional layers from well known architectures trained with millions of images like the ones mentioned in chapter 2 which can extract general features like edges or shapes to apply it to the data at hand.

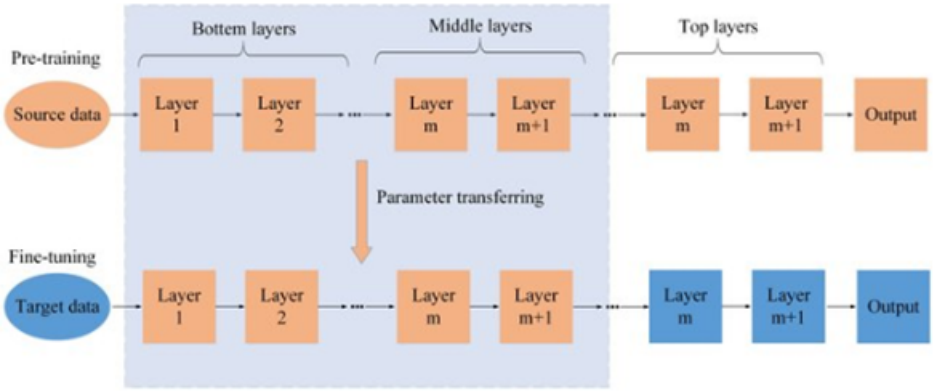


Figure 3.4: Illustration of the TL framework [3].

Figure 3.4 from [3] illustrates the general TL framework. The low and middle layers usually capture generic features of input images which has high generalization for other images, it is feasible to directly transfer the network parameters of low and middle layers to a new network which has the same architecture as the previous network. The classification parameters of top layers are still initialized in a random way in order to deal with the specific task.

3.4 CNNs to employ

Based on the low availability of sample, the HSI type of data and the good results of TL when applied to CNNs, three well established and transferable were chosen to attempt to classify the moment of perishability:

1. **Model 1: ResNet-50**: famous Microsoft architecture review cited in [30] stacked with fully connected layers for classification;

2. **Model 2: Xception**: Monsieur Chollet's architecture mentioned in [31] stacked with fully connected layers for classification;

3. **Model 3: Parallel architecture between Xception and Nasnet Mobile**: there are several papers and applications defending the merits of parallel CNN architecture, specially when attempting to detecting tumours or small organ lesions of varying sizes and positions

in medicine data [43], which is a similar task to detecting spoilage in the bananas. Also, instead of stacking fully connected layers for classification on top of the convolutional layers, a different approach will be tested which is passing the vectorized form of the features identified with the convolutional and pooling layers to another pooling layer (the global one) which will subsequently feed its results to the traditional normalizing activation function which finally outputs the probability outputs (softmax, sigmoid, tanh, etc) [44]. The argument for this "Network in Network" strattegy is to provide remedy for over adjustment of the model to the training data, i.e., overfitting, by eliminating the large number of parameters in the traditional fully conected layers.

3.5 Summary

This chapter provided with a description and sampling of the data to use, the research objectives and the methodology to use in order to build image classifier models.

In the next chapter of implementation and results, a full description of the experiment will be laid out, touching the cloud technology employed, the DL application chosen for the efforts, the implementation methodology and, finally, the results obtained.

Chapter 4

Implementation framework

In this chapter, the deep learning framework for implementing the models listed in chapter 3 will be given along with the technological tools employed for obtaining them.

4.1 Implementation on Google Cloud Platform with Tensorflow

While training, a CNN takes on inputs of millions of pixels which are then processed as arrays of vectors and in turn subjected to more and more abstract algebraic representations of the features to identify. To do so, every neuron's set of weights in the network are iteratively being updated as the data is propagated forward and backward along the entire net through a very large number of matrix multiplications.

CNN models are a form of neural networks whose architecture, due to its design and image recognition objectives, usually have millions of parameters and hundreds of layers, all of them performing simultaneous calculations.

The typical computer task requires executing a sequential set of processes, Central Processing Units (CPU) are great for these serial processing tasks because they are designed to execute complex computational tasks one at a time. Architecturally, the CPU is composed of just a few cores with lots of cache memory that can handle a few software threads at a time.

Moore's law posits that the number of transistors that can be crammed into an integrated circuit will double about every two years. For decades, that's driven a rapid increase of computing power. That law, however, has run up against hard physical limits towards the end of the XX century [45]. The surge in the development of Graphical Processing Units (GPU)s offers a way to continue accelerating applications — such as graphics, supercomputing or DL through its ability to divide tasks among many processors and perform thousands of operations at once.

All modern computers come equipped with GPU, because tasks like watching a video online or playing a light video game require mathematical tasks which, although complex and simultaneous, are easily met by the domestic graphic cards sold with a normal computer. Activities which require high usage of GPU's power like high level computer gaming, mining crypt currency or, in the case of this thesis, training very deep neural networks are not sufficiently main stream for the mass production of GPU cards by the sole GPU provider in the market, NVIDIA. This causes an imbalance in the market between supply and demand of GPUs causing the prices of computers with good enough GPU power prohibitive for the average purse.

Fortunately, there are several DL cloud providers which offer the possibility of "renting" the time of a remote machine with good GPU power. One of such providers is Google Cloud Platform [46], which allows for the configuration of remote virtual machines with access to strong hardware at very competitive prices.

In 2015, the Google brain team developed the framework for DL Tensorflow [47]. It is written in C++, but can be interacted with using other languages like Python, Javascript, Go and R. With Tensorflow, models can be trained on both GPU's and CPU's and deployed through a web service, desktop application or on a mobile device making it a primary candidate for trainign DL models. A Tensorflow instance with two CPU cores and 13GB of RAM memory plus 1 GPU with 2560 cores was hired in GCP for training and running the models.

Keras [48] is a high-level neural networks API, capable of running on top of Tensorflow, Theano and CNTK. It enables fast experimentation through a high level, user-friendly, modular and extensible API and, being part of the Tensorflow core, is capable as well of running on the CPU and GPU. It was first developed and is maintained by the creator of the CNN Xception[31] architecture François Chollet.

4.2 Train, validation and test data

As mentioned in the chapter 3, section 3.1, the available data consists of 2448 images across 204 wavelength channels with size a dimension of 436 * 320 pixels across red, green and blue channels.

From this data, a random sample of 10% was extracted and from the remaining 2203 were destined to train and validate the models (90% for training and 10% for validation of training).

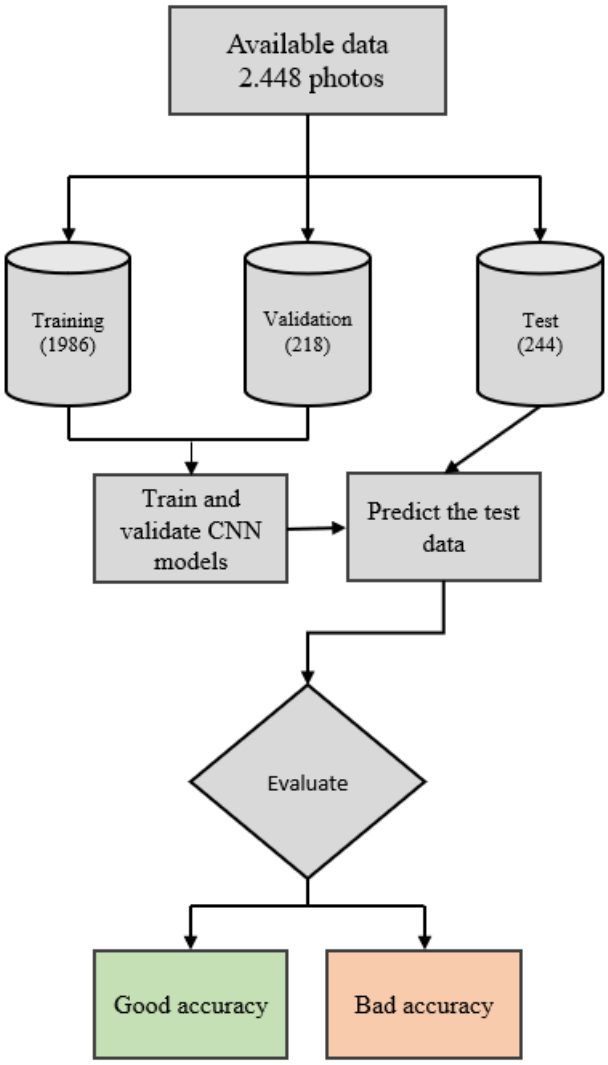


Figure 4.1: A high level research flow of the thesis effort.

After the training phase, the predictions will be made using the photos reserved for the test

set and precision measures will be made of them, calculating Top-1 Accuracy, Precision, Recall and F1 Scores for each possible class on Groups 1 and 2, mentioned in section 3.2.

4.3 Pre-training activities

After setting up the virtual machine in GCP and moving the files to it and, with the help of the Keras API, one can very quickly start the preparations for the modelling phase. The first task is to prepare the folder structure: in one folder, the test data for the predictions and, in another, the labeled training and validation data. In this second folder, it is necessary to have a sub-folder for each of prediction classes.

The *ImageDataGenerator* function from Keras starts defining some important specifications from the pre-processing phase. These are all related to data augmentation tecnhiques. Some of the most relevant:

1. **validation split**: decides on the percentage of validation data to be used, 10% in this case;
2. **horizontal flip**: a Boolean to decide if an horizontal flip of the image is desired;
3. **vertical flip**: a Boolean to decide whether a vertical flip of the image is desired;
4. **rotation range**: the range of rotation of the image, in degrees;

Below an example of these transformations:

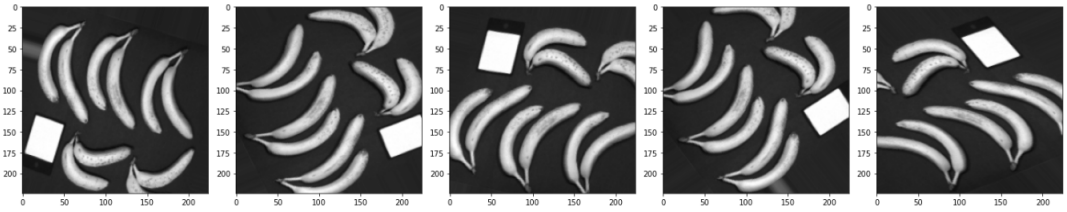


Figure 4.2: Transformations from *ImageDataGenerator* in action.

By enabling these data transformations, Keras will take a batch of images used for training, a batch being a number of training samples that will be propagated through the network. A series

of random transformations is then applied to each image in the batch (random rotation, resizing, shearing, zooming, etc.). The original batch with untouched data is then replaced by this new batch with random transformations and then is fed into the training and validation of the CNN. Using batches of data is a great tool to optimize the running time of the model, especially in the case of having large amounts of data. The batch size to be used in the models is 32, which produce good results in comparison with 16 and 64.

4.4 Model training and validation

The training is the crucial part of any ANN methodology. As mentioned in chapter 3, the modelling approach will be transfer learning of the convolutional and feature extraction layers of three of the more famous and most accurate CNN architectures and stack them with the upper layers for classification. This is where the training will occur, all previous convolutional and pooling layers will be frozen. As part of the supervised class of models, CNNs are subject to a series of hyperparameters which can be thought as the driving wheels behind the training process.

A deep learning neural network training consists in learning to map a set of inputs to a set of outputs from training data. The problem of learning is cast as a search or optimization problem and an algorithm is used to navigate the space of possible sets of weights the model may use in order to make good or good enough predictions. In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective or loss function. As the problem at hands is multi class classification, categorical cross entropy loss [49] will be the loss function used across the models. In ANNs, the final prediction is a probability vector with as many elements as the number of classes to predict and with all probabilities adding to 1. This predictions are then evaluated against the target of known outcome where only one of the elements is 1:

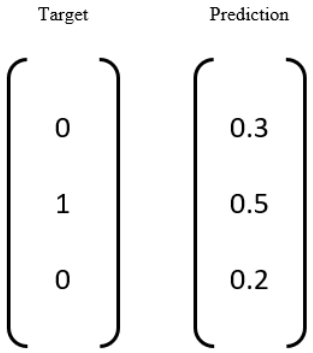


Figure 4.3: Target and prediction vectors in ANN, for a 3 class classification problem

The categorical cross entropy loss in a multi class problem is given by:

$$-\sum_{m=1}^{M} y_{o,m} log(p_{o,m}) \tag{4.1}$$

Where **M** is the number of classes, **log** is the natural log, **y** is a binary indicator (0 or 1) if class label **m** is the correct classification for observation and **p** is the predicted probability observation **o** is of class **m**. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0:

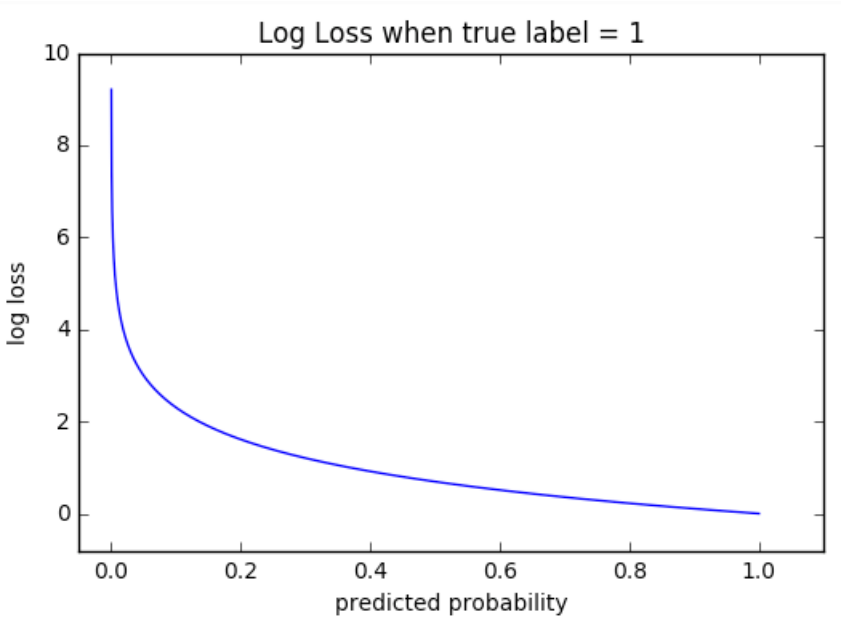


Figure 4.4: The range of cross entropy loss, given a true value in one the classes, for all predicted values

Figure 4.4 shows the range of possible loss values given a true observation (prediction = 1). As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong.

ANNs's optimization problem consists in finding the best networks weights which minimizes the cost or loss function. The process of iteratively finding the best combination of weights that enable the optimal minimum is called gradient descent [50] and happens during the back propagation phase. It can be described by the following equation:

$$x_{t+1} = x_t - \alpha \nabla Loss(x_t) \tag{4.2}$$

At each moment *t*, the gradient ∇ of the Loss function is multiplied by the learning rate α and then subtracted from the current loss to obtain the next loss. The gradient corresponds to the partial derivatives at moment *t* which indicates the slope of the loss function at this moment. If the slope is sufficiently high, it means that the process is rapidly converging into a minimum, so x_{t+1} will be greatly different from x_t, otherwise it means that the process has reached a minimum.

The parameter α represents a stabilizer of this learning process and serves to avoid aggressive oscillations and smooth the descent. Normally the learning rate is set to a decaying factor, meaning that it will slowly get smaller as training progresses, this way further smoothing the process.

There are many optimization algorithms working with gradient descent. In all three models from the dissertation, the same optimizer, Adam [19], will be used. Adam is short for "Adaptive Moment Estimation" and gained wide popularity in latest years because it tackles the danger of converging to local minimums of the loss function arriving from performing gradient descent on, not the entire training dataset, but on random subsets of the data called batches (mentioned in section 4.3) by adding to 4.2 information regarding past gradients:

$$x_{t+1} = x_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.3}$$

where:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.4}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.5}$$

$$m_t = (1 - \beta_1) G_t + \beta_1 m_{t-1} \tag{4.6}$$

$$v_t = (1 - \beta_2) G_t^2 + \beta_2 v_{t-1} \tag{4.7}$$

The parameters m_t and v_t are exponentially-weighted moving averages from past gradients G. This way, the optimization process is maintaining a recent memory of previous gradients so that it does not fall into local minimums. The β parameters are learning rate decay hyperparameters defined *a priori* and serve the purpose of getting the learning rate to be adaptive and decaying as the training progresses. the ϵ is just a random very small number to avoid division by zero.

A critical building block of CNN models is the usage of RELUs. These activation functions are used alongside the Convolutional layers because of their capacity to grant higher levels of abstraction from the input data through the introduction of non linearity. Behind the logic of RELU is a very simple computational principle:

$$f(x) = \begin{cases} 0, & \text{for x<0} \\ x, & \text{for x} \geq 0 \end{cases} \tag{4.8}$$

The function is linear for values greater than zero, meaning it has a lot of the desirable properties of a linear activation function useful when training a neural network using BPP. Yet, it is also a nonlinear function as negative values are always outputted as zero.

Within the image tasks of CNNs, RELUs work very well in the convolutional phase as what it does is to dispose of certain segments of the feature maps extracted by the sliding kernels as illustrated in figure 2.3 (for example, black colours) while leaving others (grey and white colors for example) and passing them on to the subsequent layers. As shown in 2, RELUs constitued a big jump in the accuracy and performance of CNN's recent history and are considered to be fundamental in any CNN architecture.

A final important aspect of the training parameters is the choice of the final activation function which produces the class predictions, exemplified in Figure 4.3. Again, since the problem is one of multi classification, Softmax will be used, following the example of the grand majority of estabilished architectures:

Softmax turn logits (numeric output of the last linear layer of a multi-class classification neural network) into probabilities by taking the exponents of each output and then normalizing each number by the sum of those exponents so that the entire output vector adds up to one. So, Softmax produces a vector of probabilities with as many elements as the number of classes. Each of these is calculated by:

$$S(y_i) = \frac{e^{y_i}}{\sum_{m=1}^{M} e^{y_m}} \tag{4.9}$$

Where *M* is the number of outcome classes. The *y* vector contains any real number negative or positive. Each of the numbers in *y* is subjected to the exponential function and then divided by the sum of all the exponentials present in *y*, this way guaranteeing a probabilistic interpretation of the output layer (all output values are between 0 and 1) and reducing the influence of extreme values or outliers in the data without removing data points from the set.

4.5 Evaluating methodologies

After the training phase, the model will predict the 244 unlabeled photos. In order to assess the performance of the results, some statistical measures will be used.

A confusion matrix is an useful tool for classification methods in the sense that it allows rapid visualization of the prediction performances:

		Actual					
		Day 1	Day 2	Day 3	Day 4	Day 7	Day 8+
Predicted	Day 1	16	0	0	0	0	0
	Day 2	13	36	0	0	0	0
	Day 3	0	4	40	0	0	0
	Day 4	1	1	7	50	0	0
	Day 7	0	0	0	0	27	4
	Day 8+	0	0	0	0	0	45

Table 4.1: Confusion matrix example

In table 4.1, the six classes from Group 1, mentioned in chapter 3 section 3.2 are illustrated. In this case, the T1 Accuracy woud be:

$$\textbf{T1 Accuracy} = \frac{16 + 36 + 40 + 50 + 27 + 45}{244} = \frac{214}{244} = 0.87 \tag{4.10}$$

Another measure which will be used is Precision for each class - Precision tells what proportion of predicted positives is truly positive, for each class.. Taking Day 1 as example:

$$\textbf{Precision Day 1} = \frac{16}{16 + 0 + 0 + 0 + 0 + 0} = \frac{16}{16} = 1 \tag{4.11}$$

Recall will also be calculated, it represents the proportion of actual positives which are cor-

rectly classified, for each class:

$$\textbf{Recall Day 1} = \frac{16}{16+13+0+1+0+0} = \frac{16}{31} = 0.52 \tag{4.12}$$

Lastly, the F1-score offers a way to combine precision and recall into a single number. F1-score is computed using the harmonic mean, which is given by:

$$\textbf{F1-score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{4.13}$$

In the case of class Day 1:

$$\textbf{F1-score Day 1} = 2 * \frac{1 * 0.52}{1 + 1.52} = 0.65 \tag{4.14}$$

Finally, after obtaining all the accuracy measures, statistical based on hypothesis testing methods for comparing the global performance of the three models in the two problems will be employed.

Figure 4.5 provides a recommendation for comparing multiple algorithms in more than one dataset:

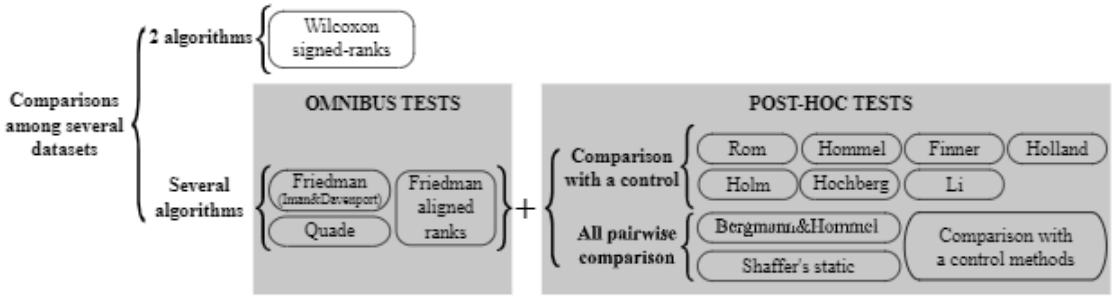


Figure 4.5: Recommended statistical test for different scenarios, from [4]

Based on the number of models and the datasets where they will fall upon and, following the recommendations in [4], an Omnibus Friedman test [51] will be used to determine if at least one of the algorithms performs differently than the others. This non parametric test considers the ranking of the accuracy results of the models in each of the data groups to obtain a statistic which is distributed like the chi-square:

$$\chi^2 = \frac{12}{Nk(k+1)} \sum_{k=1}^{K} R_k^2 - 3N(k+1) \tag{4.15}$$

Where K is the number of algorithms and N is the number of datasets. Under the null-hypothesis, all the algorithms are equivalent and so their ranks R_k should be equal.

If Friedman's null hypothesis of no difference between the models across data groups is rejected , a Finner post-hoc test [52] will be used to compare the three models across the two data groups. The logic behind it is to used the pivot quantities obtained by the Friedman ranking test to test the hypothesis that the ranking of a control method (usually the lowest ranked in Friedman's test) is different to each of the other methods.

4.6 Summary

In this chapter, the framework for the discussion of the results was laid. In the next chapter, all the results from the implementation of the three CNN models on the 2 research groups will be provided and discussed.

Chapter 5

Evaluation and results

In this last chapter before the final conclusions, the training and prediction results obtained from the CNN models on the image data will be presented and discussed. The results discussion will be divided in two sections, one for each of the research groups tested. These will be preceded by a section briefly summarizing the three model specifications. Finally, the global results T1 Accuracy results for each of the three models in both problems will be compared using statistical approaches in order to conclude if there are significant differences between them.

5.1 Model specifications

Before finally introducing the results, a brief summary of the architecture of each of the models enumerated in chapter 3, section 3.4.

Specifications	ResNet-50	Xception	Xception+Nasnet
Architecture	Fully connected layer stacked on top of the convolutional + 1 Dropout Layer + Softmax Activation Layer	Fully connected layer stacked on top of the convolutional + 1 Dropout Layer + Softmax Activation Layer	Two Global Average Pooling Layers stacked on top of NasNet and Xception architectures + 1 Dropout Layer + Softmax activation Layer
Number of layers			
Number of parameters	23.9 Million	21.2 Million	25.2 Million
Number of epochs	30	30 (15 in Group 2)	15

This final number of epochs was the result of several proofs with the modelling, and was the one which reached a stabler result in terms of training and validation performance and prediction power.

5.2 Results of modelling Group 1 with 6 outcomes

With the help of Keras, it is fairly easy to program visualization of the the training performance of the models.

For ResNet-50:

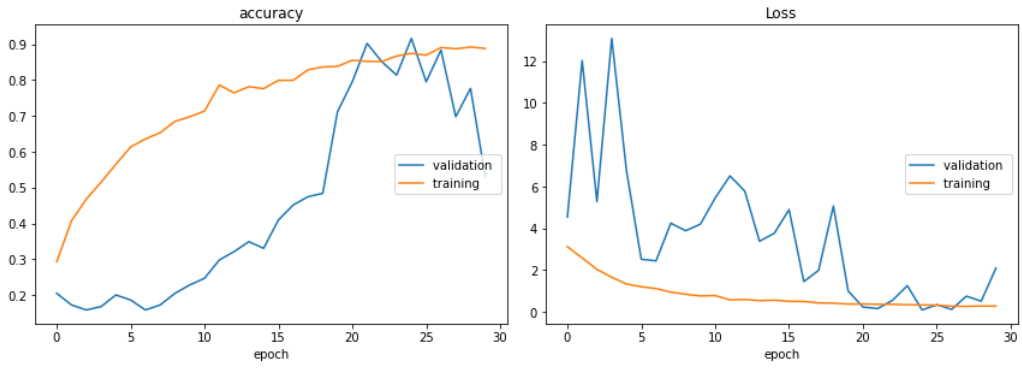
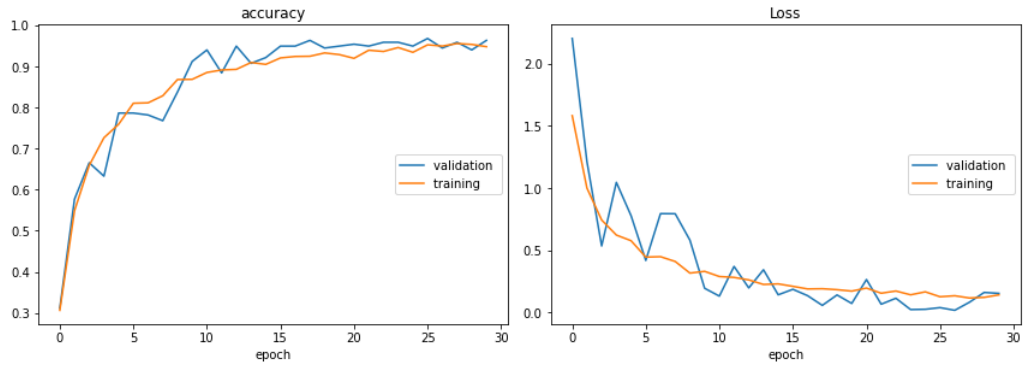


Figure 5.1: ResNet-50 training performance

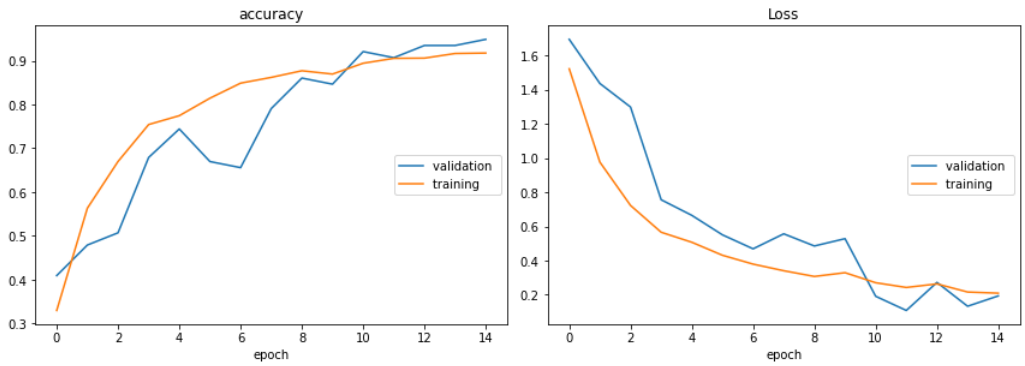
For Xception:




```
accuracy
        validation              (min:    0.312, max:    0.967, cur:    0.963)
        training                (min:    0.306, max:    0.955, cur:    0.947)
Loss
        validation              (min:    0.017, max:    2.200, cur:    0.153)
        training                (min:    0.116, max:    1.580, cur:    0.141)
```

Figure 5.2: Xception training performance

For Xception+Nasnet:




```
accuracy
        validation              (min:    0.409, max:    0.949, cur:    0.949)
        training                (min:    0.330, max:    0.918, cur:    0.918)
Loss
        validation              (min:    0.108, max:    1.695, cur:    0.194)
        training                (min:    0.210, max:    1.522, cur:    0.210)
```

Figure 5.3: Xception+Nasnet training performance

Observing the evolution of training and validation along the epochs for the three models, all of them reached reasonably high or excellent accuracy against the training data but the ResNet-50 model was not able to register the same accuracy in the validation data, which seems to indicate a problem of overfitting to the training data in this model. The spiky progression of the categorical entropy loss minimization also suggests some difficulty to converge. The Xception and

Xception+Nasnet models had a much smoother training process, having similar results in the training and validation datasets. The Xception+Nasnet needed half the epochs to reach a close to zero loss.

The raw scores for Resnet-50 model:

		Actual					
		Day 1	Day 2	Day 3	Day 4	Day 7	Day 8+
Predicted	Day 1	22	3	1	0	0	0
	Day 2	7	38	6	1	0	0
	Day 3	0	0	29	2	2	2
	Day 4	0	0	11	47	1	1
	Day 7	0	0	1	0	24	1
	Day 8+	0	0	0	0	0	45

Table 5.1: Confusion matrix for ResNet-50

Class	**Precision**	**Recall**	**F1-Score**
Day 1	0.846	0.759	0.8
Day 2	0.731	0.927	0.817
Day 3	0.829	0.604	0.699
Day 4	0.783	0.94	0.855
Day 7	0.923	0.889	0.906
Day 8+	1	0.918	0.957

Table 5.2: ResNet-50's Precision, Recall and F1-Scores per class

The model is better at identifying the bananas which are more advanced in the perishability cycle. It performs worse when identifying the Day 3 bananas.

The raw scores for Xception model:

		Actual					
		Day 1	Day 2	Day 3	Day 4	Day 7	Day 8+
Predicted	Day 1	22	2	0	0	0	0
	Day 2	7	31	1	0	0	0
	Day 3	0	6	28	0	0	0
	Day 4	0	2	19	50	0	0
	Day 7	0	0	0	0	27	4
	Day 8+	0	0	0	0	0	45

Table 5.3: Confusion matrix for Xception

Class	Precision	Recall	F1-Score
Day 1	0.917	0.759	0.83
Day 2	0.795	0.756	0.775
Day 3	0.824	0.583	0.683
Day 4	0.704	1	0.826
Day 7	0.871	1	0.931
Day 8+	1	0.918	0.957

Table 5.4: Xception's Precision, Recall and F1-Scores per class

A similar behaviour to ResNet-50 from the Xception model, although it registers perfect score in precision or recall in more classes and measures.

The raw scores for Xception+Nasnet model:

		Actual					
		Day 1	Day 2	Day 3	Day 4	Day 7	Day 8+
Predicted	Day 1	16	0	0	0	0	0
	Day 2	12	36	0	0	0	0
	Day 3	0	4	41	0	0	0
	Day 4	1	1	7	50	0	0
	Day 7	0	0	0	0	27	4
	Day 8+	0	0	0	0	0	45

Table 5.5: Confusion matrix for Xception+Nasnet

Class	Precision	Recall	F1-Score
Day 1	1	0.552	0.711
Day 2	0.75	0.878	0.809
Day 3	0.911	0.854	0.882
Day 4	0.847	1	0.917
Day 7	0.871	1	0.931
Day 8+	1	0.918	0.957

Table 5.6: Xception+Nasnet's Precision, Recall and F1-Scores per class

This model differs from the others for its relatively better results in the Day 3 class. Furthermore, it reaches four perfect Precision or Recall scores which is better than the three from Xception and the one from Resnet-50.

The research objective is to build a model which is capable of detecting where exactly a given banana is in the perishability process. Another useful metric to acess it, considering this would be one which tells how many days of distance there are between the predictions and the reality. The following table will illustrate the global T1 Accuracy for each model plus the average of the day difference between wrong predictions and its standard deviation (SD):

Model	T1 Accuracy	Average Day error	SD Day error
Resnet-50	0.840	1.615	1.248
Xception	0.832	1.049	0.218
Xception + Nasnet	0.881	1.103	0.409

Table 5.7: Accuracy and deviation of the predicitons for Group 1 across the three models

The paralleled architecture Xception+Nasnet model is the clear winner in T1 Accuracy and does fairly well in the day error difference, although not as well as the Xception model which is clearly the winner in this criteria. ResNet-50 is not at all a bad model, but does not excel in any measure of comparison against the other two models.

One other aspect which rose some curiosity was to know if, from the test data, there were certain wave length images which yielded more errors in classification. This suspicion was not founded, below data from the 29 badly classified images with the best performing Xception+Nasnet model:

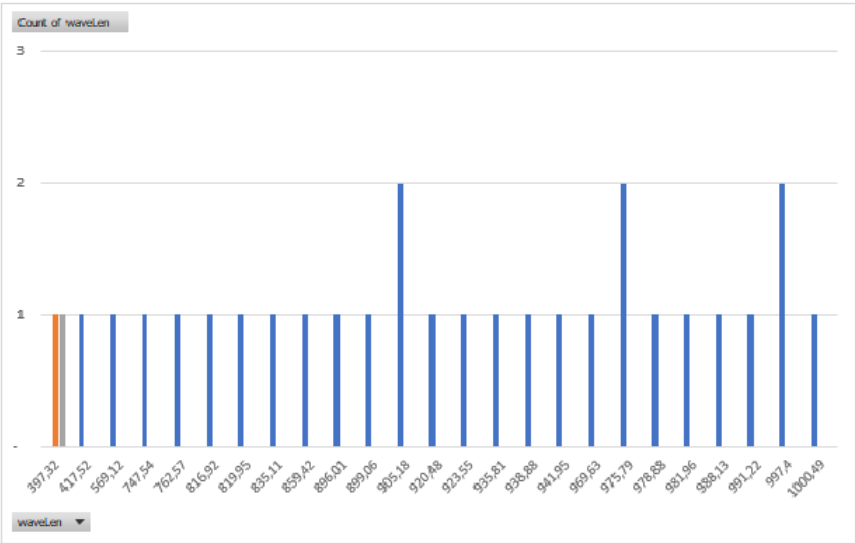


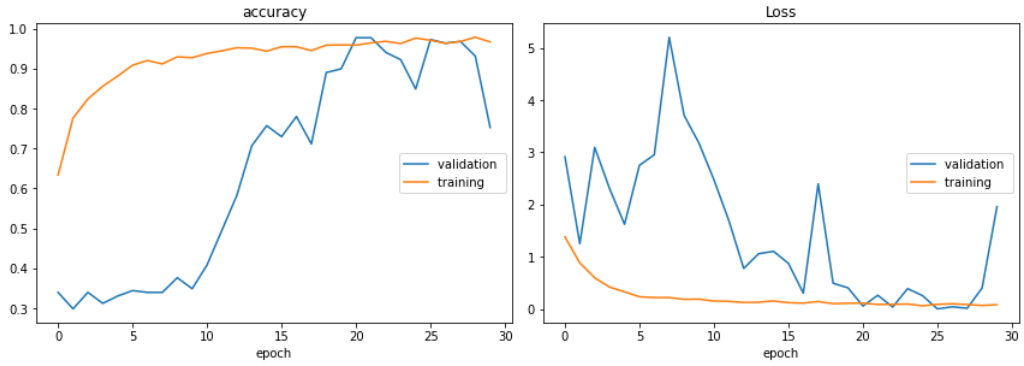
Figure 5.4: Count of error by wavelength across the 29 badly classified images with Xception+Nasnet

The distribution of the error *per* wavelength is quite flat although it is true that the instances with more day error came from images with the lowest wavelength of 397.32.

5.3 Results of modelling Group 2 with 3 outcomes

Another purpose of the research was to train and evaluate the same exact models but with less classes in order to distinguish the perishability moment between day 1 or day 2, day 3 or day 4 and day 7 or day 8. Like in the previous chapter, the confusion matrix plus the precision, recall and F1-scores will be presented individually before presenting global accuracy measures at the end.

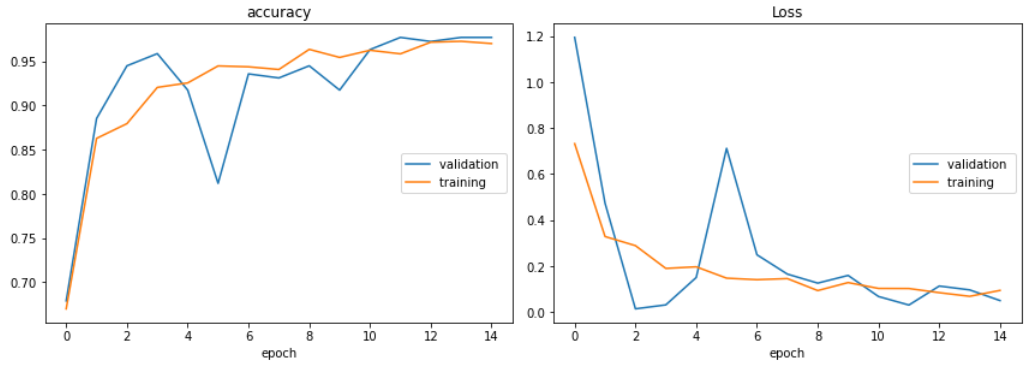
The training graphical evolution for ResNet-50:




```
accuracy
        validation              (min:    0.298, max:    0.977, cur:    0.752)
        training                (min:    0.633, max:    0.978, cur:    0.967)
Loss
        validation              (min:    0.000, max:    5.205, cur:    1.959)
        training                (min:    0.059, max:    1.380, cur:    0.080)
```

Figure 5.5: ResNet-50 training performance on Group 2

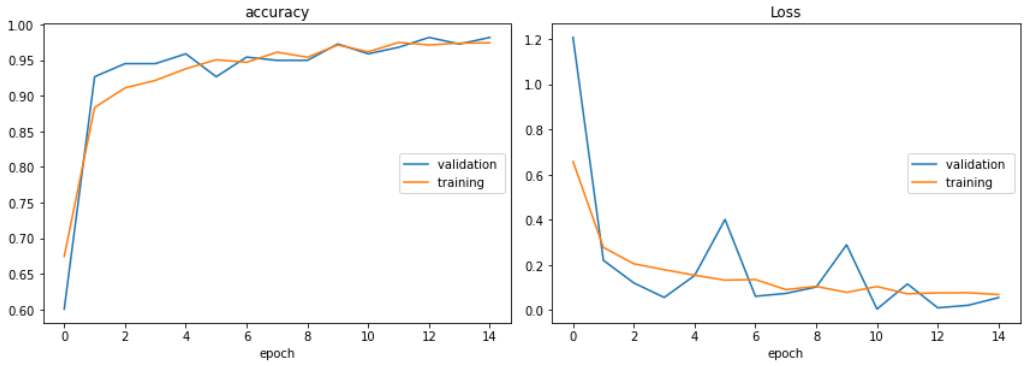
For Xception:




```
accuracy
        validation              (min:    0.679, max:    0.977, cur:    0.977)
        training                (min:    0.670, max:    0.973, cur:    0.970)
Loss
        validation              (min:    0.014, max:    1.193, cur:    0.050)
        training                (min:    0.069, max:    0.732, cur:    0.094)
```

Figure 5.6: Xception training performance on Group 2 - here only using 15 epochs instead of the 30 from Group 1

For Xception+Nasnet:




```
accuracy
        validation              (min:    0.601, max:    0.982, cur:    0.982)
        training                (min:    0.675, max:    0.975, cur:    0.974)
Loss
        validation              (min:    0.005, max:    1.208, cur:    0.056)
        training                (min:    0.069, max:    0.658, cur:    0.069)
```

Figure 5.7: Xception+Nasnet training performance on Group 2

Again, like in Group 1, Xception and Xception+Nasnet seem to fare better than ResNet-50 both in training and validation accuracy as in minimizing the categorical cross entropy loss.

The raw results from each model follow:

The raw scores for ResNet-50 model:

		Actual		
		1 or 2	3 or 4	7 or 8+
Predicted	1 or 2	70	15	0
	3 or 4	0	79	8
	7 or 8+	0	4	68

Table 5.8: Confusion matrix for ResNet-50 in Group 2

Class	Precision	Recall	F1-Score
Day 1 or 2	0.824	1	0.903
Day 3 or 4	0.908	0.806	0.854
Day 7+	0.944	0.895	0.919

Table 5.9: ResNet-5's Precision, Recall and F1-Scores per class in Group 2

With fewer classes, ResNet-50 performs considerably better hitting all Day 1 or 2 actual samples and predicting day 7 or plus quite well also.

The raw scores for Xception model:

		Actual		
		1 or 2	3 or 4	7 or 8+
Predicted	1 or 2	70	8	0
	3 or 4	0	85	0
	7 or 8+	0	4	76

Table 5.10: Confusion matrix for Xception in Group 2

Class	Precision	Recall	F1-Score
Day 1 or 2	0.897	1	0.946
Day 3 or 4	1	0.876	0.934
Day 7+	0.95	1	0.974

Table 5.11: Xception's Precision, Recall and F1-Scores per class in Group 2

Xception with 15 epochs has even better results than ResNet-50, gaining three perfect scores and F1-scores higher than 0.93 across the three classes.

The raw scores for Xception+Nasnet model:

		Actual		
		1 or 2	3 or 4	7 or 8+
Predicted	1 or 2	66	3	0
	3 or 4	3	81	0
	7 or 8+	1	14	76

Table 5.12: Confusion matrix for Xception+Nasnet in Group 2

Class	Precision	Recall	F1-Score
Day 1 or 2	0.957	0.943	0.95
Day 3 or 4	0.964	0.827	0.89
Day 7+	0.835	1	0.91

Table 5.13: Xception+Nasnet's Precision, Recall and F1-Scores per class in Group 2

This time the Xception+Nasnet architecture was not able to surpass Xception although it renders good overall results also with F1-Scores higher than 0.93. It did predict some samples very far away from its reality (1 from day 7 or 8+ which is in reality from day 1 or 2).

Like in the previous section, general T1 Accuracy results follow, as well measures of the day error. It is important to notice that this day error difference will be the minimum because, although the real moment of capture is known, since we are dealing with classes describing more than one days the exact day difference is not calculable:

Model	T1 Accuracy	Average Day error	SD Day error
ResNet-50	0.889	2.889	1.013
Xception	0.951	2.667	0.985
Xception + Nasnet	0.914	3.524	1.078

Table 5.14: Accuracy and deviation of the predictions for Group 2 across the three models

The hierarchical Xception model is clearly the best overall model for the rearranged classes in all measures and again presents good results in the "SD Day error" criteria. Xception also is the lightest and fastest to train of all the models with 21.2 million parameters which seems to demonstrate its efficiency in the tasks at hand. Below the same profiling of its errors across their wavelength:

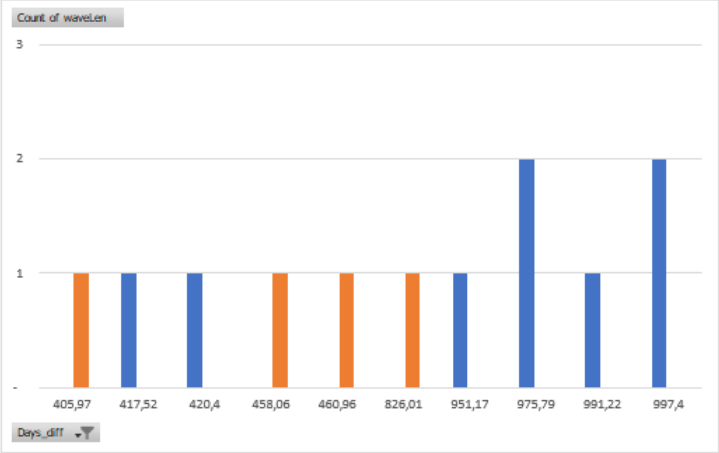


Figure 5.8: Count of error by wavelength across the 12 badly classified images with Xception

The same conclusion is taken: the distribution is relatively flat across all wavelengths but the errors corresponding to a bigger difference in days are coming from images with lower wavelength.

5.4 Statistical comparison of the T1 results

Applying the Friedman test on the global T1 Accuracy results:

T1 Accuracy	ResNet-50	Xception	Xception + Nasnet
Group 1	0.840 (2)	0.832 (3)	0.881 (1)
Group 2	0.889 (3)	0.951 (1)	0.914 (2)
Average Ranks	1.5	2	1.5

Table 5.15: Global T1 Accuracy results for all three models across the two problems. Rankings for Friedman test in parenthesis

Gives the following chi-square statistic for test:

$$\chi^2 = \frac{12}{2 * 3 * 4}(5^2 + 4^2 + 3^2) - 3 * 2 * 4 = 0.5 * 50 - 24 = 1 \tag{5.1}$$

The degrees of freedom are given by $(k-1) = 2$ which corresponds to a 5.991 critical value in the chi-square distribution at 5% significance level. This means that, for a 95% confidence level, the null hypothesis of all models performing similarly in the two data groups is not rejected.

This results should be interpreted with caution as the test recommends at least a sample of more than 15 data groups to be securely interpreted and it does not take into account the differences in scale between the accuracy results.

5.5 Summary

In this chapter all the results obtained in the deep learning online cluster platform with the data from dTX were extensively presented and discussed. Conclusions were made about which of the three CNN architectures was the most suited for the two research objectives. In general, the models yield satisfactory results with this type of data which constitutes an incentive to further research in the field.

Chapter 6

Conclusions

In this last chapter, a brief global overview of the developed work is delivered, followed by the author's thoughts on the impact of study and speculation about future work in this field and what should be the path to follow. All the while, conclusions extracted will be offered in the text.

6.1 Overview of the efforts

The research objective was to utilize HSI data gathered in laboratory regarding a complete perishability cycle of a set of bananas and build CNN architectures which would attempt to model this rotting process and be able to generalize with sufficient accuracy the predictions to other banana produces. Bananas were chosen for its high demand in the Portuguese and European markets and because of its high transportation costs due to not being a native fruit.

The literature review efforts concentrated on identifying related work in the field of computer vision applied to agriculture and food retail with a particular emphasis on the usage of CNN methodologies due to its rapid growth in sophistication and accessibility to the public in later years. Also special attention was given to previous studies utilizing computer vision in tandem with HSI input data. It was found that this type of projects are relatively abundant but, surprisingly, there are not any relevant studies applying CNN to detect spoilage in fruits with the help of HSI data.

Transfer learning was selected as the tactic for building the CNN models as that is the rec-

ommendation of most authors when the situation exhibits low amount of input data, as was the case. Besides, CNN is a field where TL is particularly adequate as the most renowned CNN architectures are all trained with image data which is very generalizable to other problems so they are adequately prepared for feature extraction tasks, which are previous to classification.

This expectation was verified when applying the transferred models to the data from this study. After performing some data preparations and fine tuning on the upper layers of the CNNs, predictions were made on a previously untouched test dataset which yielded satisfactory results with T1 Accuracy not inferior to 0.832 across all models and outcome classes.

6.2 Future work and recommendations

The significant limitation from this research was the relatively scarce amount of data available and its low variability. Although various data augmentation techniques were applied prior to modelling, there lingers the curiosity of building the same or similar CNN models with HSI images from other perishability processes and/or different experiment scenarios (for example with the presence other fruits than bananas in the input data to try and confuse the models).

HSI technology seems to work very favourably with CNNs so there is clearly a vast field of opportunities in this type of projects involving the monitoring of food quality. It would definitely be very interesting to perform a demo in a supermarket with the help of sensor apparatus which would constantly feed an AI model with input data to help in its continuous improvement. As mentioned in chapter 1, the eventual success of an initiative like this one would potentially be a game changer in the retail management and food waste reduction.

Bibliography

[1] J. Gustavsson, C. Christel, and S. Ulf. Global food losses and food waste: extent, causes and prevention. Available at http://www.fao.org/3/a-i2697e.pdf, Accessed last time in Aug 2020, 2011.

[2] "dtx digital transformation colab". [online]. https://www.dtx-colab.pt/en/homepage-en/. Accessed: 2020-08-06.

[3] Shutao Li, Weiwei Song, Leyuan Fang, Yushi Chen, Pedram Ghamisi, and Jon Benediktsson. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, PP:1–20, 04 2019.

[4] Borja Calvo and Guzmán Santafé. Scmamp: statistical comparison of multiple algorithms in multiple problems. *R J.*, 8:1–8, 01 2016.

[5] B. Marr. "what is digital twin technology - and why is it so important?" [online]. https://www.forbes.com/sites/bernardmarr/2017/03/06/what-is-digital-twin-technology-and-why-is-it-so-important/#9d19eec2e2a7. Accessed: 2020-08-06.

[6] M. Moore. "what is industry 4.0? everything you need to know?" [online]. https://www.techradar.com/news/what-is-industry-40-everything-you-need-to-know. Accessed: 2020-08-06.

[7] Erin G. Killeen Davis. *Food Waste at Retail.* PhD thesis, University of Arkansas, Fayetteville, USA, 2015.

[8] Margaret Barth, Thomas R. Hankinson, Hong Zhuang, and Frederick Breidt. *Microbiological Spoilage of Fruits and Vegetables*, pages 135–183. Springer New York, New York, NY, 2009.

[9] Yuan-Yuan Pu, Yao-Ze Feng, and Da-Wen Sun. Recent progress of hyperspectral imaging on quality and safety inspection of fruits and vegetables: A review. *Comprehensive Reviews in Food Science and Food Safety*, 14(2):176–188, 2015.

[10] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[12] ”imagenet database”. [online]. http://image-net.org/index. Accessed: 2020-08-11.

[13] B. Erdenee, T. Ryutaro, and G. Tana. Particular agricultural land cover classification case study of tsagaannuur, mongolia. In *2010 IEEE International Geoscience and Remote Sensing Symposium*, pages 3194–3197, 2010.

[14] Jayamala K. Patil and Raj Kumar. Advances in image processing for detection of plant diseases. 2011.

[15] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. 2016.

[16] Suchet Bargoti and James P. Underwood. Image segmentation for fruit detection and yield estimation in apple orchards. *Australian Centre for Field Robotics*, 206.

[17] ”keras: Layer activation functions”. [online]. https://keras.io/api/layers/activations/. Accessed: 2020-08-09.

[18] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan 2015.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[20] "dive into deep learning: Rmsprop". [online]. https://d2l.ai/chapter_optimization/rmsprop.html/. Accessed: 2020-08-09.

[21] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *30th International Conference on Machine Learning, ICML 2013*, pages 1139–1147, 01 2013.

[22] "convolution layers in cnn". [online]. https://www.cs.cornell.edu/courses/cs1114/2013sp/sections/S06_convolution.pdf/. Accessed: 2020-08-09.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[24] Yann LeCun, Léon Bottou, Yoshua Bengios, and Patrick Haffner. "gradient-based learning applied to document recognition". In *Proceedings of the IEEE*, 1998.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[29] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[31] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2018.

[32] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017.

[33] Atul Bansal Anuja Bhargava. Fruits and vegetables quality evaluation using computer vision: A review. *Journal of King Saud University - Computer and Information Sciences*, 2018.

[34] Y. Song, C. A. Glasbey, G. W. Horgan, G. Polder, G. W.A.M. van der Heijden, and J. A. Dieleman. "automatic fruit recognition and counting from multiple images". *Elsevier BV*, 2014.

[35] Suchet Bargoti and James Underwood. Deep fruit detection in orchards, 2016.

[36] X. Yang, C. Liu, Z. Wang, J. Yang, H. Min, L. Wang, and K Cheng. Identification of rice diseases using deep convolutional neural networks, 2017.

[37] Dario Pietro Cavallo, Maria Cefola, Bernardo Pace, Antonio Francesco Logrieco, and Giovanni Attolico. Identification of rice diseases using deep convolutional neural networks, 2017.

[38] K.K.J. Prakash. *Spoilage Detection in Raspberry Fruit Based on Spectral Imaging Using Convolutional Neural Networks*. PhD thesis, Technological University Dublin, Dublin, Ireland, 2018.

[39] Juan Xing, Cédric Bravo, Pál T. Jancsók, Herman Ramon, and Josse De Baerdemaeker. Detecting bruises on 'golden delicious' apples using hyperspectral imaging with multiple wavebands, 2005.

[40] Jiangbo Li, Xiuqin Rao, Fujie Wang, Wei Wu, and Yibin Ying. Automatic detection of common surface defects on oranges using combined lighting transform and image ratio methods, 2013.

[41] X. Chen, T. Pang, H. Tao, M. Lin, and H. Yang. Prediction of apple internal qualities using hyperspectral imaging techniques. In *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 450–455, 2017.

[42] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[43] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Modeling the intra-class variability for liver lesion detection using a multi-class patch-based cnn, 2017.

[44] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2013.

[45] "no more transistors: The end of moore's law". [online]. https://interestingengineering.com/no-more-transistors-the-end-of-moores-law. Accessed: 2020-08-12.

[46] "google cloud". [online]. https://cloud.google.com/. Accessed: 2020-08-12.

[47] "tensorflow". [online]. https://www.tensorflow.org/. Accessed: 2020-08-12.

[48] "keras". [online]. https://keras.io/. Accessed: 2020-08-12.

[49] "categorical cross entropy". [online]. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. Accessed: 2020-08-12.

[50] "gradient descent and adam". [online]. https://towardsdatascience.com/understanding-gradient-descent-and-adam-optimization-472ae8a78c10l. Accessed: 2020-08-12.

[51] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[52] H. Finner. On a monotonicity problem in step-down multiple test procedures. *Journal of the American Statistical Association*, 88(423):920–923, 1993.

Appendix A

Python source code

In this Appendix section, the source code employed in GCP to read, preprocess and prepare the images for modelling will be presented. All the codes and details about the experiment are also available for private consultation in https://github.com/jmmvr/Madsad_thesis_2020. For any reader interested in having access to the repository, all needed is a contact to 100485023@fep.up.pt.

A.1 Preprocessing code

```
import os
from matplotlib import pyplot
from matplotlib.image import imread
import pandas as pd
import shutil
from sklearn.metrics import precision_recall_fscore_support
    as score

# Create master label with filename and day 1 to 6
df_master_label = pd.DataFrame()

for i in range(204):

    onlyfiles = [f for f in os.listdir('Specim-
        pilot01_processed/' + str(i)) if os.path.isfile(os.
        path.join('Specim-pilot01_processed/' + str(i), f))]
    day_cycle = ['day1', 'day1', 'day2', 'day2', 'day3', '
        day3', 'day4', 'day4', 'day7', 'day7', 'day8', 'day8']
    # type(onlyfiles)
    directory_iter = 'Specim-pilot01_processed/' + str(i)
    print('directory is:', directory_iter, sep = '\n')
    onlyfiles.sort()
    print('i is:', i)
    print('files in dir are:', onlyfiles, sep = '\n')
    print(day_cycle)
    print(len(onlyfiles), ' - ', len(day_cycle))
```

```
        df_master_label_iter = pd.DataFrame(
            data = list(zip(onlyfiles, day_cycle)),
            columns = ['filename', 'cycle_day'])

        # Append to df_master_label
        df_master_label = pd.concat([df_master_label,
           df_master_label_iter])

        # Move all photos to same dTX_bananas folder
        for file in os.listdir(directory_iter):
            if os.path.exists('dTX_bananas/'):

                if len(os.listdir('dTX_bananas/')) == 0:

                    file_path = directory_iter + '/' + file
                    print('file_path is:', file_path)

                    shutil.copy(file_path, 'dTX_bananas/' + str(i
                       ) + '___' + file)

df_master_label.to_csv('01_master_label.csv', index = False,
   sep = ';')


df_check = df_master_label[df_master_label['cycle_day'] !=
   df_master_label['cycle_day_reality']]
print("Number of Rows:", len(df_check.axes[0]))
print("Number of Columns:", len(df_check.axes[1]))
```

```
# Randomly pick 10% of the photos to later make predictions
    and access goodness of accuracy

import random
import shutil
output_folder = 'dTX_bananas_test/'
for x in range(round(int(len(os.listdir('dTX_bananas/')) *.1)
    , 0)):
      to_copy = random.choice(os.listdir('dTX_bananas/'))
      shutil.move(os.path.join('dTX_bananas/', to_copy),
          output_folder)

print(len(os.listdir('dTX_bananas_test/')))
print(len(os.listdir('dTX_bananas/')))

# Create classes for prediction and fill them with the files

nodups = df_master_label.drop_duplicates()
print(nodups)

# output_folder = 'dTX_bananas_train_valid/'
print(len(os.listdir(os.path.join(output_folder, 'day1'))))
print(len(os.listdir(os.path.join(output_folder, 'day2'))))
print(len(os.listdir(os.path.join(output_folder, 'day3'))))
print(len(os.listdir(os.path.join(output_folder, 'day4'))))
print(len(os.listdir(os.path.join(output_folder, 'day7'))))
print(len(os.listdir(os.path.join(output_folder, 'day8'))))
```

```
for file in os.listdir('dTX_bananas/'):

    if ('2020-02-04_11_02_26.png' in file or '2020-02-04
       _18_38_49.png' in file):
        if os.path.exists(os.path.join(output_folder, 'day1')
           ):
            shutil.move(os.path.join('dTX_bananas/', file),
               os.path.join(output_folder, 'day1'))

    if ('2020-02-05_11_03_58.png' in file or '2020-02-05
       _18_53_55.png' in file):
        if os.path.exists(os.path.join(output_folder, 'day2')
           ):
            shutil.move(os.path.join('dTX_bananas/', file),
               os.path.join(output_folder, 'day2'))

    if ('2020-02-06_11_15_52.png' in file or '2020-02-06
       _17_47_29.png' in file):
        if os.path.exists(os.path.join(output_folder, 'day3')
           ):
            shutil.move(os.path.join('dTX_bananas/', file),
               os.path.join(output_folder, 'day3'))

    if ('2020-02-07_11_01_40.png' in file or '2020-02-07
       _17_59_01.png' in file):
        if os.path.exists(os.path.join(output_folder, 'day4')
           ):
```

```
            shutil.move(os.path.join('dTX_bananas/', file),
                os.path.join(output_folder, 'day4'))

    if ('2020-02-10_11_38_26.png' in file or '2020-02-10
        _16_56_04.png' in file):
        if os.path.exists(os.path.join(output_folder, 'day7')
            ):
            shutil.move(os.path.join('dTX_bananas/', file),
                os.path.join(output_folder, 'day7'))

    if ('2020-02-11_11_01_54.png' in file or '2020-02-12
        _10_26_50.png' in file):
        if os.path.exists(os.path.join(output_folder, 'day8')
            ):
            shutil.move(os.path.join('dTX_bananas/', file),
                os.path.join(output_folder, 'day8'))

print('------------------------')
print(len(os.listdir('dTX_bananas/')))
print(len(os.listdir('dTX_bananas_test/')))
print(len(os.listdir(os.path.join(output_folder, 'day1'))))
print(len(os.listdir(os.path.join(output_folder, 'day2'))))
print(len(os.listdir(os.path.join(output_folder, 'day3'))))
print(len(os.listdir(os.path.join(output_folder, 'day4'))))
print(len(os.listdir(os.path.join(output_folder, 'day7'))))
print(len(os.listdir(os.path.join(output_folder, 'day8'))))

# Crop images
```

```
from PIL import Image
import matplotlib.pyplot as plt

img_big = Image.open("Specim-pilot01_processed/79/2020-02-10
    _16_56_04.png")
print(img_big.size)

# the first two numbers define the top-left coordinates of
    the outtake (x,y)
# the last two define the right-bottom coordinates of the
    outtake.

img1 = img_big.crop((0, 110, 0 + 125, 110 + 165))
img2 = img_big.crop((0, 275, 0 + 135, 110 + 320))
img3 = img_big.crop((125, 255, 125 + 170, 110 + 310))
img4 = img_big.crop((120, 0, 120 + 200, 0 + 270))

f, axarr = plt.subplots(2,2)
axarr[0,0].imshow(img1)
axarr[0,1].imshow(img4)
axarr[1,0].imshow(img2)
axarr[1,1].imshow(img3)

# Prepare precision and recall score code

predicted = df_master_label['cycle_day'].tolist()
y_test = df_master_label['cycle_day_reality'].tolist()
```

```
print(predicted[:5])
print(y_test[:5])

precision, recall, fscore, support = score(y_test, predicted)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

A.2 CNN modelling code

```
from collections import defaultdict
import os
import glob
import random

import keras
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2



from keras.layers import Input, Average
from keras.layers.core import Dense, Flatten, Dropout
from keras.layers.merge import Concatenate
from keras.layers.normalization import BatchNormalization
from keras.layers.pooling import GlobalAveragePooling2D,
    GlobalMaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from keras.utils.vis_utils import plot_model
from keras.callbacks import CSVLogger, ReduceLROnPlateau,
    ModelCheckpoint
# !pip install livelossplot
from livelossplot import PlotLossesKeras
```

```
# Hyperparams

IMAGE_SIZE = 224
EPOCHS = 15
BATCH_SIZE = 32
# BATCH_SIZE = 16
VERBOSITY = 1
# TESTING_BATCH_SIZE = 50

# Output files
TRAINING_LOGS_FILE = "model_plots_csvs/training_logs.csv"
MODEL_SUMMARY_FILE = "model_plots_csvs/model_summary.txt"
MODEL_FILE = os.path.join('model_plots_csvs/', '
    model_banana_dTX.h5')
TRAINING_PLOT_FILE = "model_plots_csvs/training.png"
VALIDATION_PLOT_FILE = "model_plots_csvs/validation.png"
RES_FILE = 'model_plots_csvs/
    nasnet_xception_test_res_3_classes.csv'
TEST_DIR = '../input/'

# 5 first images from each class

classes = ['day1_2', 'day3_4', 'day7_8']
img_names = []
for i in classes:

    currentFolder = 'dTX_bananas_train_valid/' + i + '/'
    first_n = os.listdir(currentFolder)[0:5]
```

```
    full_path = [currentFolder + s for s in first_n]
    img_names = img_names + full_path

plt.figure(figsize=[15,15])
i = 1
for img_name in img_names:
    img = cv2.imread(img_name)[...,[2, 1, 0]]
    plt.subplot(6, 5, i)
    plt.imshow(img)
    i += 1
plt.show()

# Data augmentation
training_data_generator = ImageDataGenerator(rescale=1./255,
                                             validation_split
                                                 = 0.1,
                                             horizontal_flip=
                                                True,
                                             vertical_flip=
                                                True,
                                             rotation_range
                                                =90,
                                             zoom_range=0.2,
                                             width_shift_range
                                                =0.1,
                                             height_shift_range
                                                =0.1,
                                             shear_range
```

```
                                                    =0.05,
                                                channel_shift_range
                                                    =0.1)

# Train generator
train_generator = training_data_generator.flow_from_directory
    (
      directory = "dTX_bananas_train_valid/",
      target_size = (IMAGE_SIZE, IMAGE_SIZE),
#       color_mode = 'grayscale',
      classes = classes,
      class_mode = 'categorical',
      batch_size = BATCH_SIZE,
      shuffle = True ,
      seed = 42,
      save_to_dir = "dTX_bananas_train_valid/01
         _ImageDataGenerator/",
      save_prefix = "train_",
      subset = 'training'
)

n_plot = 5

def plotImages(images_arr, n_plot):
      fig, axes = plt.subplots(1, n_plot, figsize=(20,20))
      axes = axes.flatten()
      for img, ax in zip(images_arr, axes):
          ax.imshow(img)
```

```
    plt.tight_layout()
    plt.show()


augmented_images = [train_generator[0][0][0] for i in range(
    n_plot)]
plotImages(augmented_images, n_plot)


# Valid generator
val_generator = training_data_generator.flow_from_directory(
    directory = "dTX_bananas_train_valid/",
    target_size = (IMAGE_SIZE, IMAGE_SIZE),
#     color_mode = 'grayscale',
    classes = classes,
    class_mode = 'categorical',
    batch_size = BATCH_SIZE,
    shuffle = True,
    seed = 42,
    save_to_dir = "dTX_bananas_train_valid/01
        _ImageDataGenerator/",
    save_prefix = "val_",
    subset = 'validation'
)

augmented_images = [val_generator[0][0][0] for i in range(
    n_plot)]
plotImages(augmented_images, n_plot)
```

```
from keras.applications.nasnet import NASNetMobile
from keras.applications.xception import Xception
from keras.optimizers import Adam

input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3)
inputs = Input(input_shape)

#### Xception architecture

xception = Xception(include_top=False, input_shape=
    input_shape)

print(xception.output.shape.ndims)

if xception.output.shape.ndims > 2:
    output = Flatten()(xception.output)
else:
    output = xception.output

#### ResNet50 architecture

input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3)
inputs = Input(input_shape)

xception = ResNet50(include_top=False, input_shape=
    input_shape)
```

```
print(xception.output.shape.ndims)

if xception.output.shape.ndims > 2:
    output = Flatten()(xception.output)
else:
    output = xception.output

#### Xception + Nasnet paralell architecture

xception = Xception(include_top = False, input_shape =
    input_shape)
nas_net = NASNetMobile(include_top = False, input_shape =
    input_shape)

outputs = Concatenate(axis = -1)([GlobalAveragePooling2D()(
    xception(inputs)),
                                  GlobalAveragePooling2D()(
                                      nas_net(inputs))])
outputs = Dropout(0.5)(outputs)
outputs = Dense(len(classes), activation='softmax')(outputs)

# plot_model(model,
#            to_file='model_plots_csvs/model_plot.png',
#            show_shapes=True,
#            show_layer_names=True)

# from IPython.display import SVG
# from keras.utils.vis_utils import model_to_dot
```

```
# SVG(model_to_dot(model).create(prog='dot', format='svg'))

# from keras.utils.vis_utils import plot_model
# plot_model(model,
#                to_file='model_plots_csvs/
    model_plot_expand_nested.png',
#                show_shapes=True,
#                show_layer_names=True,
#                expand_nested=True)

#  Training
model_checkpoint_callback = ModelCheckpoint(
     filepath = MODEL_FILE,
     save_weights_only = True,
     monitor = 'val_accuracy',
     mode = 'max',
     save_best_only = True)

history = model.fit_generator(train_generator,
                              steps_per_epoch = len(
                                 train_generator),
                              validation_data = val_generator
                                 ,
                              validation_steps = len(
                                 val_generator),
                              epochs = EPOCHS,
                              verbose = VERBOSITY,
                              callbacks = [PlotLossesKeras(),
```

```
                                        model_checkpoint_callback
                                            ,
                                        CSVLogger(
                                            TRAINING_LOGS_FILE
                                            ,
                                                    append =

                                                        False
                                                        ,
                                                    separator
                                                        = ';
                                                        ')],
                                pickle_safe = True)

model.load_weights(MODEL_FILE)

# Training and validation plots
import matplotlib.pyplot as plt
epochs = [i for i in range(1, len(history.history['loss']) +
    1)]

plt.plot(epochs, history.history['loss'], color='blue', label
    ="training_loss")
plt.plot(epochs, history.history['val_loss'], color='red',
    label="validation_loss")
plt.legend(loc = 'best')
plt.title('training')
plt.xlabel('epoch')
```

```
plt.savefig(TRAINING_PLOT_FILE, bbox_inches='tight')
plt.show()

plt.plot(epochs, history.history['accuracy'], color = 'blue',
    label = "training_accuracy")
plt.plot(epochs, history.history['val_accuracy'], color = '
   red',label = "validation_accuracy")
plt.legend(loc = 'best')
plt.title('validation')
plt.xlabel('epoch')
plt.savefig(VALIDATION_PLOT_FILE, bbox_inches='tight')
plt.show()

predictions = model.predict_generator(
   roc_validation_generator,
                                        steps = len(
                                           roc_validation_generator
                                           ),
                                        verbose = VERBOSITY)



test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    directory = "dTX_bananas_test/",
    target_size = (IMAGE_SIZE, IMAGE_SIZE),
    color_mode = "rgb",
    batch_size = BATCH_SIZE,
    class_mode = None,
```

```
    shuffle = False
)
test_generator.reset()
print(len(test_generator))

#Predict from generator (returns probabilities)
pred = model.predict_generator(test_generator, steps=len(
   test_generator), verbose = 1)

print(pred)

# Get classes by np.round
predicted_class_indices = np.argmax(pred, axis=1)
print(predicted_class_indices[:5])

labels = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels.items())
predictions = [labels[k] for k in predicted_class_indices]
print(predictions[:5])

filenames = test_generator.filenames
results = pd.DataFrame({"Filename": filenames,
                        "Predictions": predictions})
# export to csv
results.to_csv(RES_FILE, index=False)
results
```