INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de monitorização de colmeia de abelhas

DIOGO MANUEL DE MATOS BRAGA
julho de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Bee Hive Monitoring System

Solutions for the automated health monitoring

Diogo Manuel de Matos Braga

Dissertation for the fulfilment of Masters' Degree in Informatics Engineering, Specialization in Information and Knowledge Systems

Advisor: Ana Maria Dias Madureira Pereira (AMD)

Porto, July 2020

Resumo

Cerca de um terço da produção global de alimentos depende da polinização das abelhas, tornando-as vitais para a economia mundial. No entanto, existem diversas ameaças à sobrevivência das espécies de abelhas, tais como incêndios florestais, *stress* humano induzido, subnutrição, poluição, perda de biodiversidade, agricultura intensiva e predadores como as vespas asiáticas. Destes problemas, pode-se observar um aumento da necessidade de soluções automatizadas que possam auxiliar na monitorização remota de colmeias de abelhas. O objetivo desta tese é desenvolver soluções baseadas em Aprendizagem Automática para problemas que podem ser identificados na apicultura, usando técnicas e conceitos de *Deep Learning*, Visão Computacional e Processamento de Sinal. Este documento descreve o trabalho da tese de mestrado, motivado pelo problema acima exposto, incluindo a revisão de literatura, análise de valor, design, planeamento de testes e validação e o desenvolvimento e estudo computacional das soluções. Concretamente, o trabalho desta tese de mestrado consistiu no desenvolvimento de soluções para três problemas – classificação da saúde de abelhas a partir de imagens e a partir de áudio, e deteção de abelhas e vespas asiáticas. Os resultados obtidos para a classificação da saúde das abelhas a partir de imagens foram significativamente satisfatórios, excedendo os que foram obtidos pela metodologia definida no trabalho base utilizado para a tarefa, que foi encontrado durante a revisão da literatura. No caso da classificação da saúde das abelhas a partir de áudio e da deteção de abelhas e vespas asiáticas, os resultados obtidos foram modestos e demonstram potencial aplicabilidade das respetivas metodologias desenvolvidas nos problemas-alvo. Pretende-se que as partes interessadas desta tese consigam obter informações, metodologias e perceções adequadas sobre o desenvolvimento de soluções de IA que possam ser integradas num sistema de monitorização da saúde de abelhas, incluindo custos e desafios inerentes à implementação das soluções. O trabalho futuro desta dissertação de mestrado consiste em melhorar os resultados dos modelos de classificação da saúde das abelhas a partir de áudio e de deteção de objetos, incluindo a publicação de artigos para obter validação pela comunidade científica.

Palavras-chave: Visão Computacional, *Deep Learning*, Monitorização da Saúde de Abelhas, Processamento de Sinal

Abstract

Up to one third of the global food production depends on the pollination of honey bees, making them vital for the world economy. However, between forest fires, human-induced stress, poor nutrition, pollution, biodiversity loss, intensive agriculture, and predators such as Asian Hornets, there are plenty of threats to the honey bee species' survival. From these problems, a rise of the need for automated solutions that can aid with remote monitoring of bee hives can be observed. The goal of this thesis is to develop Machine Learning based solutions to problems that can be identified in beekeeping and apiculture, using Deep Learning, Computer Vision and Signal Processing techniques and concepts. The current document describes master thesis' work, motivated from the above problem statement, including the literature review, value analysis, design, testing and validation planning and the development and computational study of the solutions. Specifically, this master thesis' work consisted in developing solutions to three problems – bee health classification through images and audio, and bee and Asian wasp detection. Results obtained for the bee health classification through images were significantly satisfactory, exceeding those reported by the baseline work found during literature review. On the case of bee health classification through audio and bee and Asian wasp detection, these obtained results were modest and showcase potential applicability of the respective developed methodologies in the target problems. It is expected that stakeholders of this thesis obtain adequate information, methodologies and insights into the development of AI solutions that can be integrated in a bee health monitoring system, including inherent costs and challenges that arise with the implementation of the solutions. Future work of this master thesis consists in improving the results of the bee health classification through audio and the object detection models, including publishing of papers to seek validation by the scientific community.

Keywords: Computer Vision, Deep Learning, Bee Health Monitoring, Signal Processing

Acknowledgements

First and foremost, I would like to thank my advisor PhD Ana Madureira for all the support and mentoring given throughout both my masters and this thesis. Her availability, expertise and guiding were a pillar of support for the completion of my masters and for the development of this thesis.

I would also like to thank my family and loved ones for the entire support, education and love they have given me throughout my entire life, especially in my times of need. You are the ones that helped me keep sane whenever I lost my cool, and I am forever thankful to be lucky enough to have you as a part of my life.

To my friends, I would like to thank them for the support, good laughs, and companionship they have given me throughout my life, both professional and personal.

Finally, I would also like to thank Dr. Ofélia Anjos for the help provided in obtaining the required knowledge of beekeeping and apiculture needed for this thesis, as well as the Calouste Gulbenkian Foundation for being the ignitors of this thesis' idea and the feedback given for some of the solutions developed on this thesis.

Index

List of Figures

List of Tables

List of Equations

Acronyms List

AI	Artificial Intelligence
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
AP	Average Precision
API	Application Programming Interface
APL	A Programming Language
BHM	Bee Health Monitoring
BHMS	Bee Health Monitoring System
CCD	Colony Collapse Disorder
CNN	Convolutional Neural Network
COCO	Microsoft Common Object in Context data set
CV	Computer Vision
DL	Deep Learning
FAIR	Facebook AI Research
FCIS	Fully Convolutional Instance-aware Semantic Segmentation
FETL	Feature Extraction Transfer Learning
FFE	Fuzzy Front End
FPS	Frames Per Second
FTTL	Fine-Tuning Transfer Learning
GPS	Global Positioning System
GPU	Graphical Processing Unit
HoQ	House of Quality
IoU	Intersection over Union
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MFCC	Mel Frequency Cepstral Coefficients
ML	Machine Learning
MNC	Multi-task Network Cascade
mRNA	Messenger Ribonucleic Acid
NCD	New Concept Development
NPD	New Product Development
PASCAL VOC	PASCAL Visual Object Classes
QFD	Quality Function Deployment
RCNN	Recurrent Convolutional Neural Network
RNN	Recurrent Neural Network
RGB	Red, Green, Blue
RoI	Region of Interest
SP	Signal Processing
SSD	Single Shot Multibox Detector
SWOT	Strengths Weaknesses Opportunities Threats
YOLO	You Only Look Once

1 Introduction

The purpose of this chapter is to provide the context and motivation of the AI solutions for a honey bee health monitoring system developed in this thesis. The concerns surrounding the numerous threats to honey bees will be provided, as well as their vital importance to the global economy. Furthermore, the objectives of this master thesis will be specified. Finally, the approach and the methodology will be described, as well as the remainder of the document's structure.

1.1 Motivation

Honey bees (i.e. western honey bees *Apis mellifera*) are the world's most frequent pollinators of crops, averaging 13% of all floral visits. Furthermore, around 5% of plant species worldwide are exclusively visited by honey bees (Hung et al., 2018), with 200 economically important plants that require bee pollination for reproduction. In countries where agricultural production is an active role, honey bees are accepted as an important factor in modern agriculture, since they thrive in diverse climates, are domesticated, and can be manipulated by people. Related academic work indicates that crop quality and quantity can be increased by using honey bees to facilitate crop pollination, even in self-pollinating plant species, with some crops' yields depending 100% on honey bees and other pollinator insects (Sirali & Cinbirtoğlu, 2018). As stated by the Food and Agriculture Organization of the United Nations (2018), more than 75% of the world's food crops rely to some extent on pollination for yield and quality. Without pollinators, apples, coffee, tomatoes, and cocoa, among others, would not exist. The European Commission highlights that these specimen provide vital ecosystem services to crops and wild plants, forming a key component of European biodiversity (European Commission, 2015).

However, several threats to the honey bee species can be identified. For example, Colony Collapse Disorder (CCD) is a phenomenon characterized by an unexplained rapid loss (60-90%) of a colony's adult population (Underwood & vanEngelsdorp, 2008; vanEngelsdorp et al., 2017). One odd characteristic of CCD colonies is that they usually have plenty of food stores

and can even contain a queen and a small number of young workers, and the colony's reserves remain untouched by robbing bees or honey bee comb pests for several weeks after the collapse. Furthermore, CCD colonies rarely have the bodies of the dead bees in the hive (Nath, 2016; Underwood & vanEngelsdorp, 2008; vanEngelsdorp et al., 2017). Some of the potential causes for CCD consist in parasites (e.g., *varroa destructor*, *acaparis woodi*), insecticides, genetic mutations, climate conditions and viruses and fungi (Nath, 2016). As shown in *Figure 1*, although there was a large decline in worldwide honey bee population around the 1990s, the global honey bee population seems to be increasing, except for regions of Europe and North America, suggesting that CCD cases are diminishing over the years (FarmMeetsTable, 2016). However, small cases of CCD still exist on specific regions and apiaries (vanEngelsdorp et al., 2017).

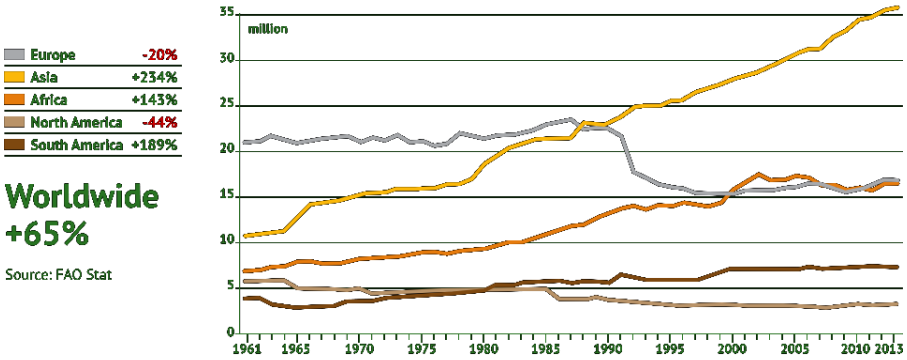

Figure 1 - Worldwide honey bee population between 1961 and 2014 (FarmMeetsTable, 2016).

The effects of CCD in Europe can be seen between 1990 and 1995. Additionally, forest fires, human-induced stress, poor nutrition, pollution, biodiversity loss and intensive agriculture still pose as threats for the survival of honey bees (FarmMeetsTable, 2016; Food and Agriculture Organization of the United Nations, 2018; Jacques et al., 2017; Milius, 2018). Human-induced stress can occur when beekeepers tend to their bee hives, as it is often interpreted as an intrusion by the colony's bees. Another threat to honey bees is the yellow-legged hornet (*Vespa velutina*), commonly referred to as the Asian Hornet. *V. velutina* were accidentally introduced into regions such as Europe from Asia via boat transport (Monceau, Thiery, et al., 2014). The biological invasion of Vespidae raises significant problems for several sectors of Europe, as *V. velutina* prey on domestic and commercial honey bees. Just the presence of *V. velutina* is enough to increase mRNA expression of oxidative stress-related genes, catalase activity and lipid peroxidation, impacting honey bees negatively (Leza et al., 2019). This added stress contributes to events such as CCD. Furthermore, *V. velutina* can be deadly to allergic people and the invaders have rapidly risen an impressive size, especially in the west of Europe (Monceau, Thiery, et al., 2014). Even though signs of these invaders date to records as early as 2003, their expansion to countries such as Portugal, France and the UK was only registered starting from 2012 (Coloss, 2019).

The world population is expected to rise in the upcoming years, creating a demand for higher food production in order to sustain this growth. The western honey bees pollinate crop species that compose up to one-third of the average diet, worldwide (Tiwari, 2018). As such, there is a demand for the development of solutions that can aid preserving honey bees in a healthy state, since these play a vital role in the produces provided to humans by ecosystem services.

1.2 Objectives

On the context of this master's thesis, the goal of the project is to develop solutions for a Bee Health Monitoring System (BHMS) for honey bees. Specifically, the development of Artificial Intelligence (AI) models that can detect signs of bee health issues is intended. The system for which the AI models will be developed has the goal of reducing the frequency at which beekeepers must inspect honey hives in order to assess their current health. The BHMS should be able to perform bee health classification, which consists in the assessment of the overall health status of a hive's colony, including whether the hive has signs of *Varroa Destructor* or other relevant health threats. Furthermore, the BHMS should also be able to detect the presence of *V. velutina*, in order to alert its users to quickly act on their removal. Throughout this thesis, the term "honey bee" and "bee" will be used interchangeably.

For the bee health classification, the goal in this project is to use Computer Vision (CV), Signal Processing (SP) and Deep Learning (DL) techniques in order to monitoring honey bee health using images and audio. Furthermore, in order to provide an AI solution to the BHMS so that it can perform the necessary tasks, the thesis' problem can be further divided into the following 3 subsets of problems and goals:

- Development of a classifier that can predict a honey bee's health status, given a cropped image of the bee. The classification result will then be combined with other classification results to assess the global health of a hive.
- Development of an object detection model that can detect honey bees and *V. velutina*, in order to automatically crop footage of honey bees for the health classifier, and to monitor the presence of *V. velutina*, respectively.
- Development of a classifier that can predict the health of a hive through audio, complementing the image-based classifier, in order to provide a more robust solution, using Machine Learning (ML) techniques for signal processing.

Considering feedback obtained from experts and companies in the field, regarding the object detection model, it must have an accuracy near 80% or more. For both the classification through image and through sound, these should exhibit an accuracy higher than 80%, preferably near 90%. For all classifiers, their speed should be near real-time, i.e. the algorithm should provide a result in a time frame of a second, which corresponds to an elapsed time of less than one second.

On the context of this work, the approach for the development of this master's work will be as follows:

- Review of the state-of-the-art of CV, DL & SP, including recent and relevant techniques that can be used for the development of the system, with adequate analysis and comparison of each technique using review papers.
- Survey of similar BHMS approach conducted both nationally and internationally, either in a commercial or academic environment.
- Comparison of design alternatives for the development of the BHMS, with the respective evaluation of each design's advantages and disadvantages.
- Specification of the experimentation, trials, and testing plans, which will be conducted for the evaluation of the developed predictive models.
- Development of a prototype of the solutions for the BHMS.

1.3 Document Structure

The remainder of this document is structured as follows:

- **Literature Review** describes the review of the state-of-the-art surrounding this thesis' scope, including tools and frameworks, and definitions of several concepts used throughout this document, as to provide a base of understanding.
- **Value Analysis** has the goal of evaluating the opportunity of this thesis' solution, in order to assess if the solution has solid bases regarding its needs and demands that make its development viable. The value of the solution of this thesis is also presented and discussed in this chapter.
- **Design** showcases the requirements of this thesis' project, as well as the alternative designs that were considered for the solution. This chapter also describes the decision process behind the selection of the chosen design.
- **Validation and testing** describes the questions and hypothesis that will be tested, as well as the evaluation plan, including the quality standards for the data to be gathered. Furthermore, the metrics and tests that will be used to evaluate each sub problem's solution are also specified and described.
- **Development & Computational Study** describes the development process of the models for each sub problem of this thesis' problem, including pre-processing steps, general considerations assumed during development, evaluation algorithms implemented and used, results, and the computational study conducted to compare and validate the obtained results of the different models. Furthermore, the chapter also presents the discussion of the obtained results and their implications on the thesis' objectives and stakeholders.
- **Conclusion** summarizes the findings obtained from this thesis, highlighting the contributions of this master thesis' work for the respective stakeholders. Finally, the limitations of the solution developed in this master's thesis, as well as future work, is also described in this chapter.

2 Literature Review

Machine Learning is a subarea of AI which relates to the study of computational algorithms that automatically obtain information and detect relevant patterns from observed data (Bishop, 2006). The learning process is based on acquiring knowledge through example, direct experience and instruction, without human intervention during the process (Schapire, 2008; Shalev-Shwartz & Ben-David, 2014). ML encompasses disciplines such as probability theory, statistics, convex analysis, algorithm complexity theory and approximation theory, among others. Furthermore, it can be stated that the applications of ML span the entire field of AI (J. Liu et al., 2018). In the last decades, ML has been widely applied in fields such as computer-aided disease diagnosis and health monitoring, bioinformatics, psychology and human strategic behaviour prediction, marketing and churn prediction, music, terrorism, CV and computer generated images (Fang et al., 2016; Hartford et al., 2016; J. Liu et al., 2018; Vafeiadis et al., 2015). On the context of ML, the following types of algorithms can be identified (Ayodele, 2010; Dobra, 2017):

- **Supervised learning:** the algorithm generates a representation that maps between input data and an intended output.
- **Unsupervised learning:** a model that groups the entire data set in subsets is created by the algorithm, without providing an intended output.
- **Semi-supervised learning:** a combination of labelled and unlabelled data, i.e. input data which has an intended output and data that does not have any intended output. The expected result is a representation that correctly maps all data to an output.
- **Reinforcement learning:** the algorithm learns a behavioural policy through actions taken in an environment. Each action has an impact on the virtual environment and it provides feedback to the algorithm.

On the context of supervised learning, a further categorization on the types of algorithms can also be identified (Dobra, 2017):

- **Classification:** applies when input data is used to predict a category, such as classifying an image between 'dog' and 'cat'. When the class to be predicted only assumes two values, it is a problem of two classes (or binomial classification). When the class can assume more than two values, the problem is called multiclass classification.
- **Regression:** like classification but applies to real values.
- **Anomaly detection:** applies when the identification of sets of points that do not belong to a normal pattern is intended.

The purpose of this chapter is to describe a review of the literature surrounding this master thesis's scope. A review on the state of the art of CV will be conducted, including its origins and the most recent and relevant techniques that can be used, as well as DL and its advancements towards CV and Signal Processing. Furthermore, surrounding related work in the context of bee health monitoring will be described.

2.1 Artificial Neural Networks

An Artificial Neural Network (ANN) (Rosenblatt, 1958) is an interconnected set of simple processing elements, named after units or nodes, whose functionality is inspired by the current knowledge of the animal nervous system, although it does not have a detailed realistic representation (Gurney, 2004; Nascimento Jr, 1994). The processing capacity of the network is stored in the strength of the connections between the units, also known as weights. These weights are obtained through a process of adapting to, or learning from, a set of data (Gurney, 2004).

The units (or artificial neurons) of the ANN represent the fundamental processing unit of a neural network, representing an artificial simple model of the neuron. Each input terminal (dendrites) and the values received with a certain weight are combined by a mathematical function f_a that is equivalent to the processing performed by the sum. The output value corresponds to the neuron's response to the input values, so several functions can be used (e.g. sigmoid, hyperbolic tangent, etc.). The combination of the input values can be described by the following equation (Gama et al., 2015):

$$u = \sum_{j=1}^{n} x_j w_j \quad (1)$$

Where:

- u - total input received by the neuron
- X_j - value present in the input connection j
- W_j - value of the weight present in the input connection j
- n - number of entries in the neuron

As shown in *Figure 2*, ANN can have several layers, with different types of connections, varying the problem-solving capacity, so they can be used for regression and classification problems. Formally, the neural network's first and last layers correspond to the input and

output layer, respectively. The input layer is responsible for receiving the input values to process, whilst the output layer is responsible for converting the network's response to the desired format. All layers between the input and output layers are called hidden layers, which perform extra data processing (Gama et al., 2015).

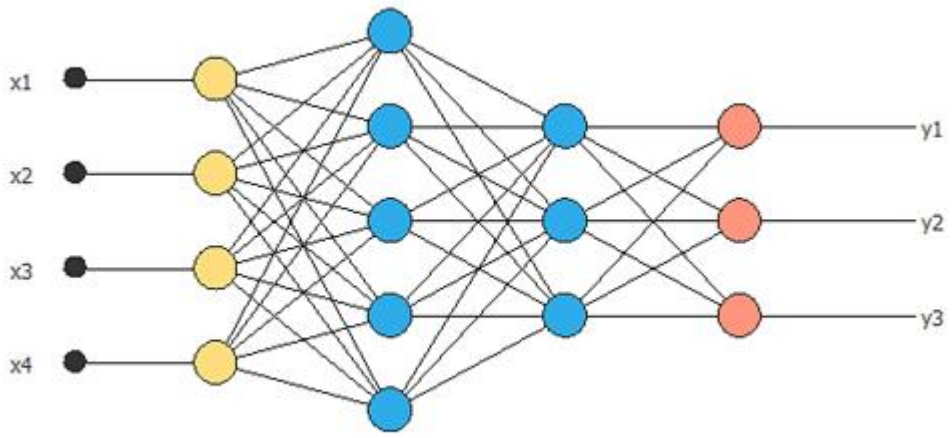

Figure 2 - Example of the architecture of an ANN (NeuralDesigner, 2020). *The yellow, blue, and red neurons represent, respectively, the input, hidden and output neurons, in their respective layers. As can be seen, the input data (x1, x2, x3, x4) gets converted into a desired output (y1, y2, y3).*

2.2 Deep Learning

In the late seventies, Fukushima (1980) had performed work on a type of neural network that would allow the computer to extract a set of features that optimally represent the data for a specific problem. This type of architecture (Neocognitron), which is referenced as the ancestor of what latter became Convolutional Neural Network (CNN), was initially applied to medical image analysis by Lo et al. (1995). In the 1990s, the "era of Convolutional Neural Networks" began when LeCun et al. (2015; Voulodimos et al., 2018) displayed the network's success in a real-world application by publishing the first paper using the architecture for document recognition. One of the key disadvantages presented at the time was that CNN were inefficient for training. However, due to the evolution of ML and deepening of Artificial Neural Networks, the concept of DL was introduced.

DL can be defined as the field of ML which concerns the development of computational models that are composed of multiple processing layers, which can learn representations of data with multiple levels of abstraction (LeCun et al., 2015). Each processing layer is composed by a determined amount of non-linear modules that transform a specific representation at one level, granting models the main advantage of automatically extracting the necessary features in order to complete a given task. This is due to each hidden layer

being able to represent the original input in a higher abstract-level concept than the previous hidden layer preceding it, enabling the model to learn very complex functions from the composition of enough of these transformations and concepts (LeCun et al., 2015).

While the author of the first paper about DL is Geoffrey Hinton (2007b), whom referred to DL as “Deep Networks”, the term was first coined by Andrew Ng. et al. (2013), whose work was developed with a team at Google research. DL has drastically improved the state-of-the-art in domains such as speech recognition, visual object recognition, object detection, natural language processing and others. Since DL can discover intricate structures in high-dimensional data, it can be applied in many domains of science, business and government (LeCun et al., 2015). It can be stated that DL has beaten several records in image recognition and speech recognition, outperforming traditional ML techniques in tasks such as activity prediction of potential drug molecules, analysis of particle accelerator data, brain circuit reconstruction, prediction of the effects of mutations in non-coding DNA on gene expression and disease, among others (LeCun et al., 2015). In DL, it is common to use Rectified Linear Units (ReLU) and similar activation functions. ReLU is a piecewise linear function that outputs the input value received if it is positive and zero otherwise, as shown in *Figure 3* (Brownlee, 2019a).

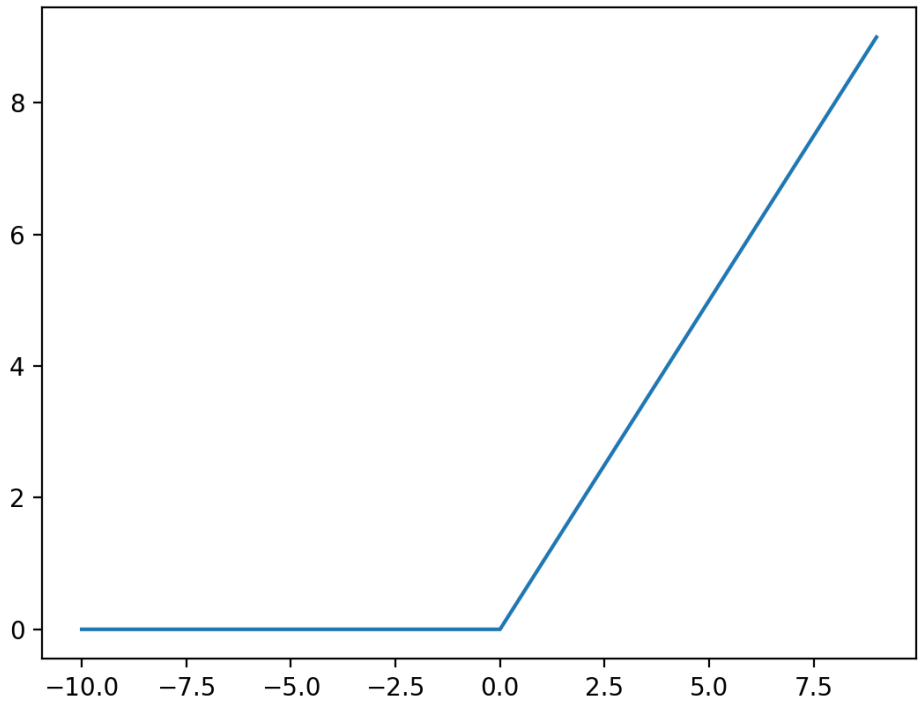

Figure 3 - Line plot of a ReLU activation function

On the context of DL, there are several algorithms that can be identified. For example, in case of natural language processing and speech processing fields, RNN and LSTM are architectures that are commonly applied. CNN are also used in the former fields, as well as in CV.

Generative Adversarial Network are used in unsupervised learning and game-theoretical frameworks, with a common application of generating images like the ones present in each data set. In the next sections, some of the previously mentioned architectures will be described in the context of their common areas of application.

2.3 Computer Vision History

One of the first applications of DL in image processing was in the medical field, since one of the first works with CNN was about medical image analysis (Lo et al., 1995). Therefore, when DL appeared as a solution to efficiently training deep networks (Litjens et al., 2017), the technique was naturally joined with CNN to perform classification tasks in medical images and exams, becoming one of the first areas in which DL provided major contributions to medical image analysis. Exam classification can be defined as a type of task where the goal is to analyse one or multiple images and perform a single diagnostic variable as an output (such as a binary output variable for the presence of a disease) (Litjens et al., 2017). It is important to note that the application of DL with CNN also had aid of techniques such as transfer learning to help overcome the data set size barrier associated with medical applications – the comparatively small data set size when compared to typical CV applications (thousands vs millions of samples).

Transfer learning is a technique where pre-trained networks are used to try to work around the requirement of a large data set for deep network training, having two types of strategies: (1) pre-trained network for feature extraction (Feature Extraction Transfer Learning, FETL), which has the advantage of not requiring to train a network, as the features can be plugged in onto an existing system, such as, an image analysis pipeline, and (2) fine-tuning (Fine-Tuning Transfer Learning, FTTL), a pre-trained network on a given data set (Litjens et al., 2017). However, as far as the best transfer learning strategy is concerned, it can be stated that there is not a clear certainty of which strategy is overall better. Despite some studies have hinted at FTTL being the best strategy, due to having higher results than any approach with FETL, extensive and exhaustive work done by Kim & Hwang (2016) and Antony et al. (2016) have conflicting results in regards to which strategy is best. Furthermore, Litjnes et al. (2017) compared studies that used a pre-trained CNN with other studies that use a CNN trained from scratch. Although pre-trained have better results, the increase is not significant enough to perform any generalized conclusions. Still, it can be stated that there are some classification tasks where FTTL exhibits better results than using FETL, as is the case with recent work done by Esteva et al. (2017) and Gulshan et al. (2016), published in high-ranking journals.

When comparing the evolution of the types of deep networks commonly applied in exam/image classification, it can be stated that the timeline is similar to the one of CV (Litjens et al., 2017). The latter timeline starts with Fukushima's (1980) work of creating a self-organizing neural network model that could perform pattern recognition (and, sub sequentially, feature extraction) without being affected by shift in position. This model was labelled as Neocognitron (Voulodimos et al., 2018). Afterwards, between 1985 and 1990,

came the use of Boltzmann Machines and respective variants (Restricted Boltzmann Machines), which can be defined as a network of symmetrically connected neuron-like units that make stochastic decisions, designed to reconstruct the input (generative approach). During this time, RNN also appeared, which correspond to a structure of a neural network adapted to learn time series, alongside the discovery of Autoencoders, which is a technique composed of simple learning circuits that transform inputs into outputs with the least possible amount of distortion (Baldi, 2012; Hinton, 2007a; Voulodimos et al., 2018). As stated previously, in the 1990s, the work of LeCun (2015; Voulodimos et al., 2018) and LeNet started the "era of Convolutional Neural Networks", followed shortly by the appearance of Long-Short Term Memory (LSTM) neural networks, which are RNN designed to have a forget gate and a block gate, in an attempt to mimic the way the human brain works by forgetting some of the events learned. LSTM have an advantage in that they perform better in modelling temporal sequences with long-range dependencies when compared to RNN (Sak et al., 2015). In 2006, Deep Belief Networks appeared with the work of Hinton (Hinton et al., 2006), which ushered the "age of deep learning" (Voulodimos et al., 2018) and also resulted in the appearance of Deep Boltzmann Machine in 2009 (which represent an architecture of Boltzmann Machines with many layers of hidden variables) (Salakhutdinov & Hinton, 2009). Both were the first signs in CV of the usage of DL.

Finally, in 2012, the work of Hinton et al. (Krizhevsky et al., 2012) on AlexNet[1] started the age where CNN and DL are used for ImageNet classification, establishing a predominant use of CNN, due to the published results being marginally better than the state-of-the-art at the current time (Krizhevsky et al., 2012). ImageNet is an image database that follows the WordNet hierarchy organization where each node of the hierarchy is depicted by hundreds and thousands of images (Princeton University et al., 2019). Hinton et al. (Krizhevsky et al., 2012) applied CNN to classify 1.2 million high-resolution images on the ImageNet LSVRC-2010 contests into 1000 possible distinct classes, having submitted a model that achieved a top-5 winning test error of 15.3% (compared to 26.2% achieved by the second-best entry). In their work, the authors also used an efficient GPU implementation and regularization methods such as Dropout, which are typical techniques in DL implementation.

In the last decade, advances have been made to the CNN architecture, depending on the goal of its application (i.e. image classification, object detection, image segmentation). Some of the most significant contributions are (Khan et al., 2019): AlexNet, which won the 2012-ILSVRC competition (one of the most difficult challenges for image detection and classification, at the time); GoogleNet[1], which won the 2014-ILSVRC competition (introduced the concept of split, transform and merge blocks); ResNet[1], proposed by Microsoft, for the net training of 150 layers deep networks and DenseNet[1], which uses the idea of cross-channel connectivity.

[1] AlexNet, GoogleNet, ResNet and DenseNet correspond to names given to specific configurations of certain architectures that won challenges or achieved state-of-the-art results.

2.3.1 Classification Task, Segmentation and Object Detection

In the context of ML, classification can have two different meanings: (1) establishing the existence of classes or clusters in a given set of observation or (2) establishing a set of rules where one can classify a new observation into one of pre-established existing possible classes. The first definition refers to the Unsupervised Learning, whilst the second definition, which can be considered the standard definition of classification, refers to Supervised Learning, associated in statistical literature to discrimination problems (Michie et al., 1994). In the context of CV, classification has a similar definition to the one given in ML, as it is defined as the process of categorizing a vector of stimuli or features into a finite set of classes, meaning that classification in CV usually involves recognizing the dominant content in a scene (Cooley et al., 2017). It is of importance to note that in CV, detection differs from classification, since the former requires knowing the location and/or count of a certain type of objects in a scene, essentially making detection a process composed of both classification and localization.

The term segmentation (or image segmentation) consists of the partitioning of an image into a set of regions that cover it, with the goal of obtaining regions that represent meaningful areas of the image, such as tumours or polyps in a medical image, or crops, cars, urban areas and forests in satellite images (Shapiro & Stockman, 2000). These regions can either be a border of pixels surrounding the area of interest or a shape, such as a circle, ellipse, or polygon. The goal of segmentation is simultaneously to decompose the image into parts for further analysis and to perform a change of representation of the regions into more meaningful concepts and/or efficient units for further processing and analysis. Image segmentation encompasses a wide range of fields, such as object recognition and detection, which consist in the detection and recognition of one or more objects in a specific scene, depending on the type of problem. Semantic image segmentation is regarded as the more fine-detailed case of image segmentation, where each pixel is classified into belonging to a specific class, essentially clustering the pixels in accordance to their classes (A. Brown & Nvidia, 2017; X. Liu et al., 2018). Although object detection is encompassed by image segmentation, it differs from the former in that object detection has the goal of detecting the bounding shape (usually a box) of the target object, unlike segmentation, which includes the region shape of the detected object.

2.3.2 Convolutional Neural Network

As stated previously, CNN are a type of architecture that can automatically extract a set of features that optimally represent the data for a specific problem. Formally, CNN are composed of two major layers: Feature Learning and Classification/Fully Connected (Bouvrie, 2006; Chatterjee, 2019a; Wu, 2019).

The Feature Learning data is responsible for the automatic extraction of the set of features that optimally represent the data. In this layer, the goal is to find a set of optimal convolutions (i.e. kernels) that can be applied to the given input data. As CNN were developed for medical

image analysis, this input data usually corresponds to images (Bouvrie, 2006; Wu, 2019). The kernel corresponds to a small matrix that can be applied to a given input data, which produces an intended output result, as shown in *Figure 4*. In the case of CNN, the kernels usually produce a new set of features for a given input, as shown in *Figure 5*.

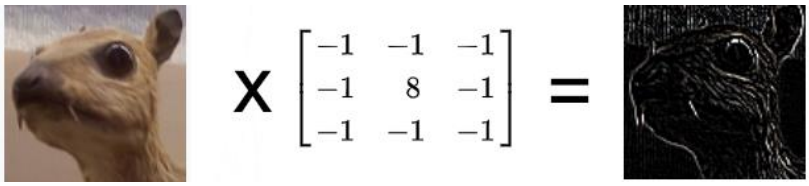

Figure 4 – Example of the application of an edge detection kernel in an image. The original image (on the left) results with the right image when applying the kernel.

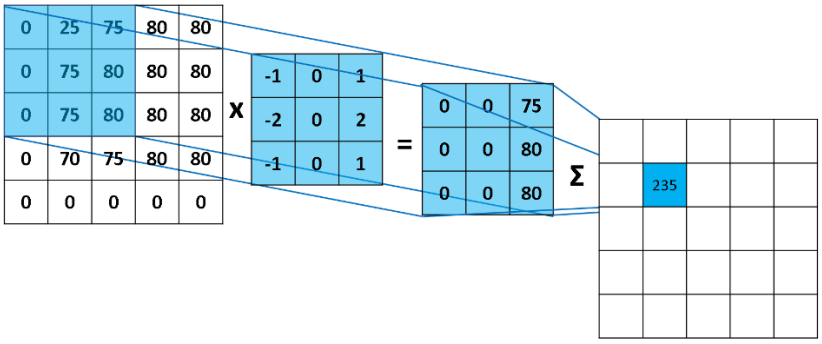

Figure 5 – Example of a kernel producing a new feature (Machine Learning Notebook, 2017).

In addition to finding the kernel on each convolutional layer, pooling is also performed. Pooling layers serve the function of progressively reducing the spatial size of the original input's representation, operating on each feature map independently (Bouvrie, 2006; Wu, 2019). The goal is to reduce the impact of small movements of features on the image in the resulting feature map (i.e. down-sampling) and the number of parameters and computation in the network. Pooling is a more common and robust version of down-sampling (Bouvrie, 2006; Machine Learning Notebook, 2017; Wu, 2019). The two most common pooling methods are as follows:

- **Average pooling:** computes the average for each patch on the feature map.
- **Max pooling:** computes the maximum value for each patch of the feature map.

The Feature Learning layer's features are then flattened and passed onto the Fully Connected layer, which usually consists of a standard ANN that uses the flattened features to perform the classification task, as shown in *Figure 6*.

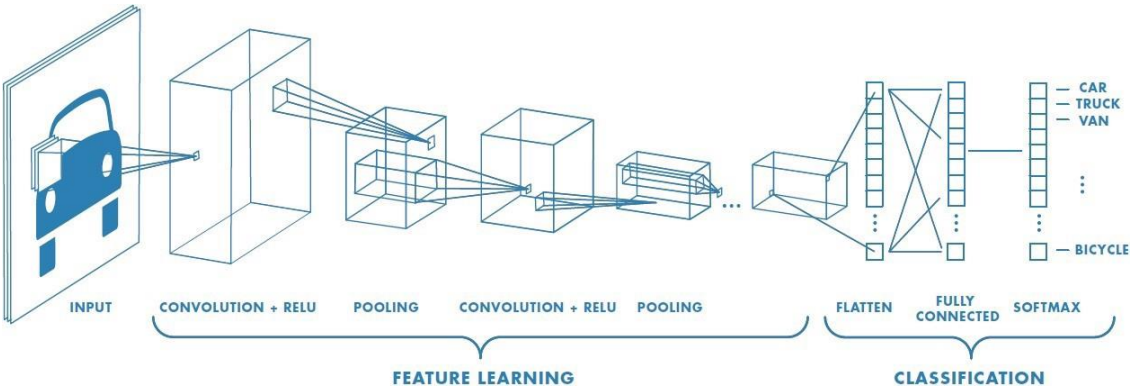

Figure 6 – CNN architecture visualized (Chatterjee, 2019a).

On the context of CNN, the explanation of concepts such as dropout, padding and strides is important, as these are commonly used when working with CNN. Dropout can be defined as an operation where neural units are ignored (hence the word "dropout") during the training phase (Budhiraja, 2018; N. Srivastava et al., 2014). These units are usually chosen at random, according to a given probability *p*. Essentially, at each training stage, individual nodes have a 1-*p* probability of being dropped out of the network, with their incoming and outcoming edges also removed, as seen in *Figure 7*.

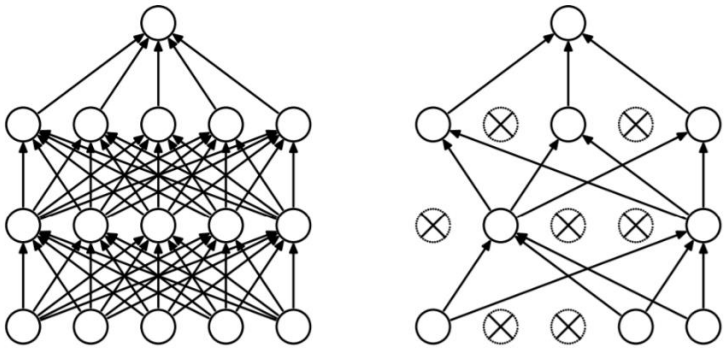

Figure 7 - Example of a standard neural network (left) and the same network after applying dropout (right) (N. Srivastava et al., 2014).

This operation is used to prevent over-fitting of the network, as it can be seen as a form of regularization (which consists in reducing over-fitting by adding a penalty to the loss function). As a fully connected layer occupies most of the parameters, during the training process neurons develop co-dependency among each other, curbing the individual power of each unit and leading to over-fitting. The following observations can be obtained from the application of dropout (Budhiraja, 2018; N. Srivastava et al., 2014):

- Forces a neural network to learn more robust features which are useful when paired with the different random subsets of neurons.

- Roughly doubles the number of iterations required to converge, but the training time for each epoch is less, and almost always improves the performance of the neural network.

In the case of padding, it can be stated that it consists of an operation that adds a border of features to an input object (e.g. a border of blank pixels to an image), with a specific width (which is stated by the padding value), so that all features have the chance of being the focus of the kernel (Brownlee, 2019b). For example, considering an 8x8 image and a 3x3 kernel, it can be stated that the original image will be shrunken down to a 6x6 image, since the kernel can only be applied 6 times both horizontally and vertically. In this example, the pixels at the border of the image (i.e. those that were excluded) were not processed as being the focus (centre) of the kernel, and only as the neighbouring cells of a given cell. This type of behaviour can cause issues when stacking multiple convolutional layers or using big kernels. In order to compensate for this behaviour, padding adds an amount of white space cells so that these also get the chance of being the centre of the kernel, and allowing for a non-lossy convolutional, as can be seen in *Figure 5*, where the left and bottom rows are padded with zeros.

On a similar fashion, stride(s) refers to the amount of movement between applications of the kernel to the image (Brownlee, 2019b). Typically, the kernel is moved one unit to the right and bottom, represented as (1,1) (1 horizontally, 1 vertically). Strides are typically symmetrical and changing these values can cause certain effects such as shrinking (down-sampling) of the image (e.g. by applying a kernel with (2,2) strides, it will shrink an image of 8x8 to 4x4).

Additionally, CNN typically have a SoftMax layer at the output of the network, which corresponds to a layer that performs a SoftMax function on the input values. The SoftMax function transforms a vector of *k* real values (positive, negative, zero, etc.) into another vector of *k* real values that sum up to 1. This function is particularly useful to convert a set of inputs into values between 0 and 1, so that these can be interpreted as probabilities. As several neural networks usually have a penultimate layer which outputs real-valued scores that are not conveniently scaled, the SoftMax function is used to convert the scores into a normalized probability distribution. However, it must be stated that the SoftMax function can only be used in classifiers whose classes are mutually exclusive, due to the defined behaviour of the SoftMax function (Wood, 2019).

Finally, on the topic of the CNN, it can be stated that although significant improvements have been made to the CNN, it still possess the following set of limitations, among others (Hosseini et al., 2017; Lipton & Priya, 2017):

- Do not encode the position and orientation of the object into their predictions.
- When feeding the negative of an image, the classification output can change significantly and unpredictively.
- Hyper-parameter tuning is non-trivial.
- Require a large data set to have desirable performance.

- As convolutions are significantly less efficient than operations such as max pooling, if the network is significantly deep, each training step is going to take much longer.

2.3.3 Recurrent Convolutional Neural Network

Due to the way Recurrent Convolutional Neural Networks (RCNN) work, the explanation of what Recurrent Neural Networks (RNN) must also be provided, apart from the description of the CNN conducted previously. Although the term was not coined yet, the architecture itself was already present as early as the 1980s on the work done by Jordan (1986). This first work described a theoretical treatment of neural networks that learned actions. The term seems to have been first coined in the work published by Pearlmutter (1989) about *"State Space Trajectories in Recurrent Neural Networks"*, with another paper published in the same year by Cleeremans et al. (1989) about "*Finite-state Automata and Simple Recurrent Networks*". Finally, Elman (1990) popularized the simple RNN, wide spreading the term for the architecture.

RNN can be described as a supervised learning architecture composed of artificial neurons with one or more feedback loops, which correspond to recurrent cycles over a given sequence (e.g. time) (Pinheiro & Collobert, 2014). As such, training RNN in a supervised fashion requires a data set of training input-target pairs, where the goal is to minimize the difference between the output and the target pairs (i.e. the loss value). This is achieved by optimizing the weights of a network, much like ANN. Formally, RNN have three types of layers, of which two are like ANN: an input layer and an output layer (Pinheiro & Collobert, 2014). The hidden layers, in the case of RNN, are called recurrent hidden. The input layer has the particularity of being composed of N input units, whose inputs represent a sequence of vectors through a time series, such as *{…, X_t, X_{t1}, X_{t2}, …}*, where each X_t will have a vector of *N* values like *(x_1, x_2, x_3, …, x_n)*. As shown in *Figure 8*, the recurrent hidden layers are connected through each other through a temporal sequence with recurrent connections.

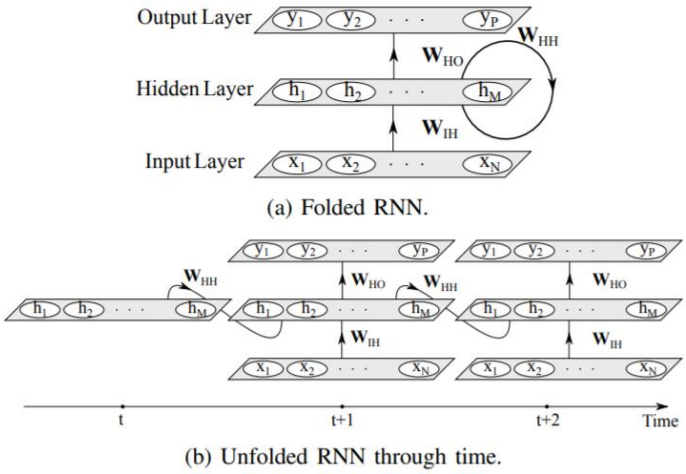

Figure 8 – Recurrent Neural Network architecture visualized (Pinheiro & Collobert, 2014).

The hidden layer defines the state space or "memory" of the system as per below:

$$h_t = f_H(o_t) \tag{2}$$

Where:

$$o_t = W_{IH} x_t + W_{HH} h_{t-1} + b_h \tag{3}$$

- $f_H(\cdot)$ is a function for the hidden layer's activation
- b_h is the bias vector of the hidden units

A RCNN can be essentially described as a regular RNN that uses the CNN architecture type instead of the standard ANN's (Kalchbrenner & Blunsom, 2013). As shown in *Figure 9*, the RCNN architecture has a first component that is essentially a CNN, which is followed by an RNN, instead of the standard flattening and ANN process.

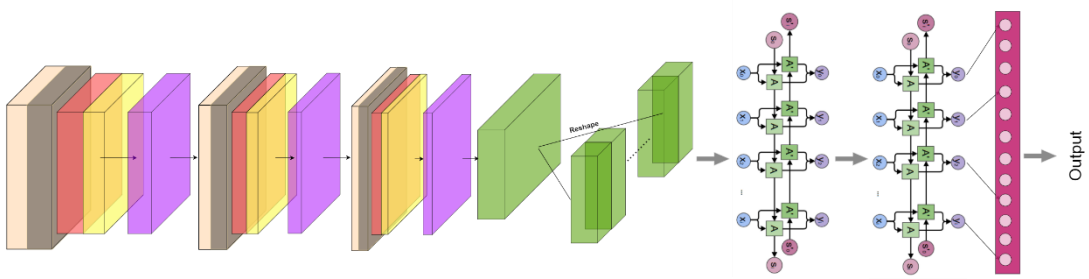

Figure 9 - RCNN architecture. In the picture, it can be seen that the architecture starts with a CNN and ends with a RNN (Chatterjee, 2019b).

RCNN have the benefit of combining the advantages provided by CNN and RNN, which consist on the joint image/label embedding and label co-occurrence capabilities that allow the proper modelling of label co-occurrence dependency in joint label embedding space (Kalchbrenner & Blunsom, 2013; Pinheiro & Collobert, 2014; Wang et al., 2016). However, RCNN have difficulties in detecting small objects in an image, due to limited discriminativeness of the global visual features of such small objects.

2.3.4 Mask-RCNN

On the topic of object detection, one of the most significant contributions is the Mask-RCNN architecture, which is a Recurrent Convolutional Neural Network that extends the Faster R-CNN – one of the best architectures for object detection and image segmentation (He et al., 2017). Developed by the Facebook AI Research (FAIR), Mask-RCNN adds a branch for predicting an object mask in parallel with the branch that performs bounding box recognition, allowing for a much simpler training process with a small over-head, capable of running at 5 Frames-Per-Second (FPS).

Formally, the Faster R-CNN consist of a first stage, called Region Proposal Network (RPN), and

a second stage, mostly based on Fast R-CNN (He et al., 2017). The RPN has the goal to propose candidate object bounding boxes from a given image of any size, resulting in a set of rectangular object proposals with a value that indicates how likely the rectangular region belongs to the given object being detected (Ren et al., 2016). This is done by sliding a small network over the convolutional feature map output by the feature map produced by the convolutional layers, as shown in *Figure 10*. The second stage extracts features using a RoIPool (Region of Interest Pool) from each candidate bounding box, as shown in *Figure 11*. These features are then used to perform the classification and bounding box regression.

Mask-RCNN adopts a similar strategy to the Faster R-CNN model, working essentially as an extended version of the latter (He et al., 2017). The first stage of Mask-RCNN is identical to Faster R-CNN's (an RPN). However, for the second stage, as stated previously, Mask-RCNN outputs a binary mask for each RoI in parallel to predict the class and box offset, unlike most recent systems, where classification depends on mask predictions. During training, Mask-RCNN has a multi-task loss on each sampled RoI as follows:

$$Loss = Loss_{cls} + Loss_{box} + Loss_{mask} \tag{4}$$

Where the classification loss ($Loss_{cls}$) and the bounding-box loss ($Loss_{box}$) are identical to the ones defined in the Fast R-CNN (Girshick, 2015; He et al., 2017). The mask branch has a Km^2 dimensional output for each RoI, which encodes K binary masks of a target m x m resolution, one for each of the K classes, by applying a pixel-by-pixel sigmoid and defining the mask loss ($Loss_{mask}$) as the average binary cross-entropy loss. With this definition, the Mask-RCNN can generate masks for every class without competing with the remaining classes, as it relies on the dedicated classification branch to predict the class label used to select the resulting mask. This essentially decouples mask and class prediction, allowing for good instance segmentation results.

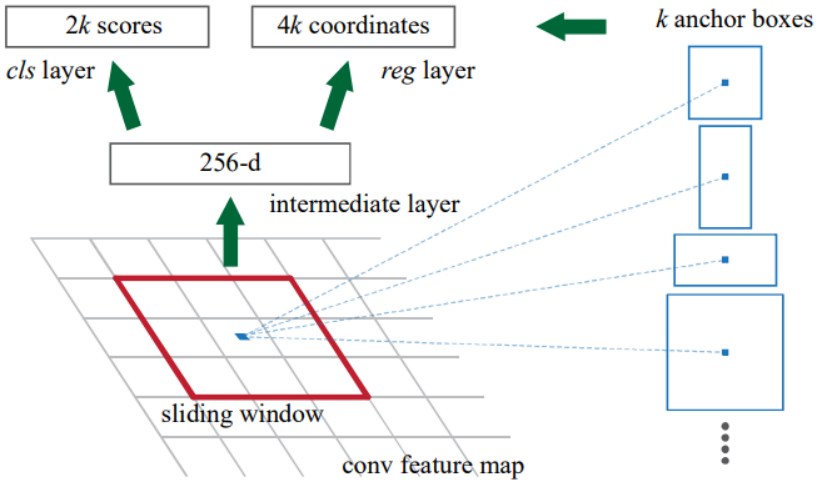

Figure 10 - Region Proposal Network Architecture (Ren et al., 2016).

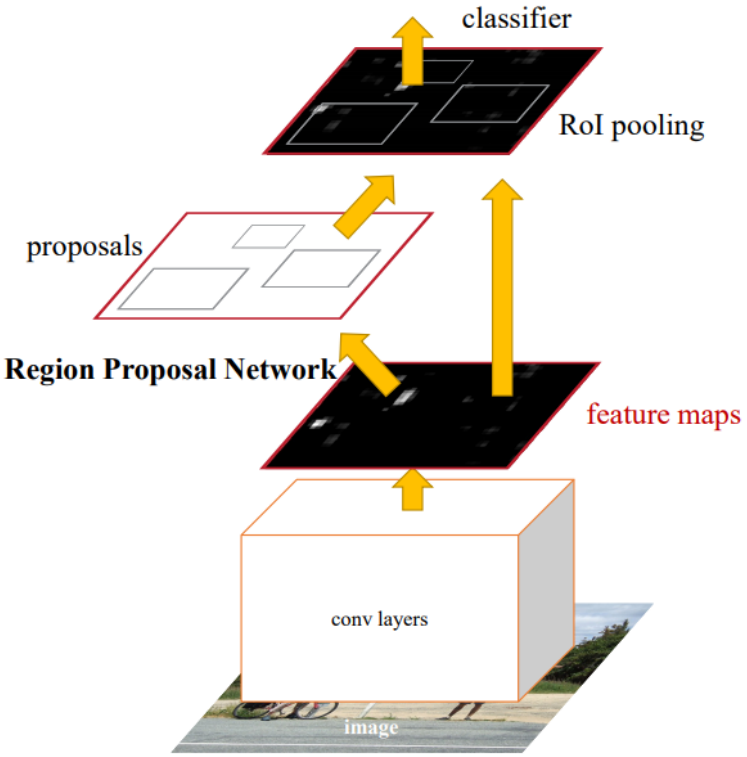

Figure 11 - Faster R-CNN Model Architecture. In the image, feature maps are extracted from the convolutional layers and then fed into an RPN, whose output values are combined with the original feature maps on a RoI pooling layer, in order to produce the final bounding box (Ren et al., 2016).

The significant advantages of using Mask-RCNN, despite its low FPS, are its robust training process. As stated by He et al. (2017), the architecture is fast to train and can outperform baseline variants of previous state-of-the-art models, including MNC and FCIS, which were the winning architectures of the image segmentation challenges COCO 2015 and 2016, respectively. Furthermore, Mask-RCNN can achieve good results even under challenging conditions, as it has a reduced artefact count on overlapped instances, as shown in *Figure 12*.

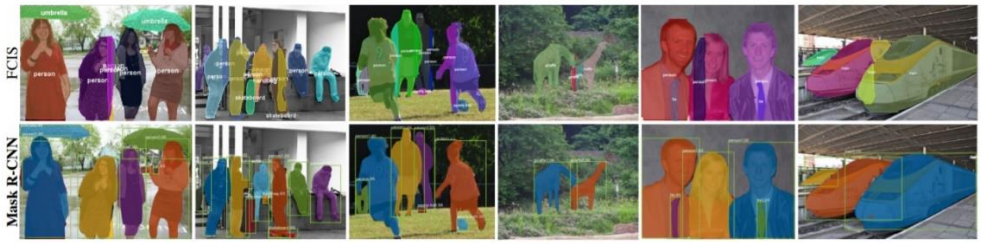

Figure 12 - Comparison of artefact presence in Mask R-CNN and FCIS. As can be seen, FCIS shows some artefacts on images where overlapping instances exist (He et al., 2017).

2.3.5 You Only Look Once

You Only Look Once (YOLO) can be defined as a framework that predicts both confidences for multiple categories and bounding boxes using the whole topmost feature map (Redmon et al., 2016). The motivation behind the development of YOLO is the fact that, at the time, approaches such as RCNN were deemed to have complex pipelines that were slow and hard to optimize because each component had to be trained individually. As such, Redmon et al. (2016) developed an architecture of an object detection model that would treat the task as a single regression problem, starting with the original image pixels and ending straight into the bounding box coordinates and probabilities. Simply, YOLO has a single convolutional network simultaneously predicting multiple bounding boxes and respective class probabilities.

Formally, YOLO divides an input image into an *S x S* grid, and, if the centre of an object falls into a grid cell, the cell is responsible for predicting the object centred in that grid cell, using convolutional and fully connected layers (Redmon et al., 2016). As such, each grid cell produces *B* bounding boxes and respective confidence scores that reflect the model's confidence regarding the presence of the object in the cell, as shown in *Figure 13*. Furthermore, the confidence score also serves as an accuracy metric of the model for the given cell. The confidence score is computed as per below:

$$conf(Object) = Pr(Object) * IOU_{pred}^{truth} \quad (5)$$

Where:

- *Pr(Object)* is the probably of the object
- IOU_{pred}^{truth} is the Intersection Over Union (IOU) of the ground truth mask with the predicted mask

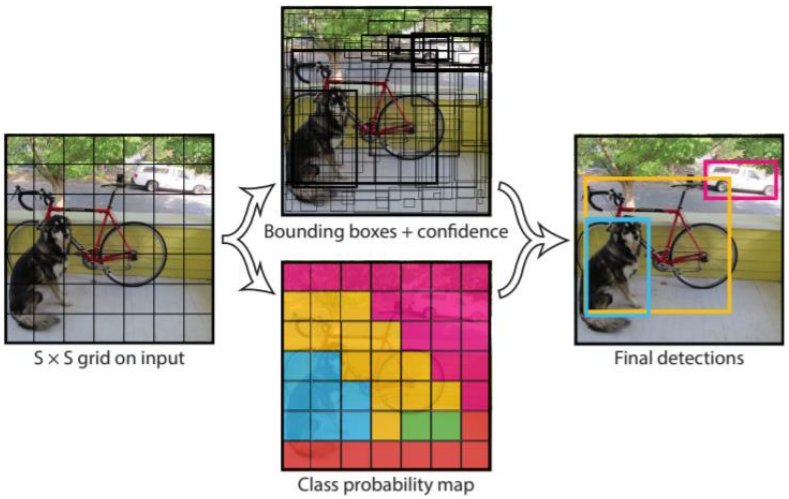

Figure 13 - Example of YOLO's model performing on an image, where each cell of an SxS grid predicts the bounding box and classification of the object (Zhao et al., 2019).

The predictions are combined in order to produce the final confidence score of the predicted class and respective bounding box (Redmon et al., 2016; Zhao et al., 2019). YOLO's network design can be seen in *Figure 14*, which is inspired by the GoogLeNet's design.

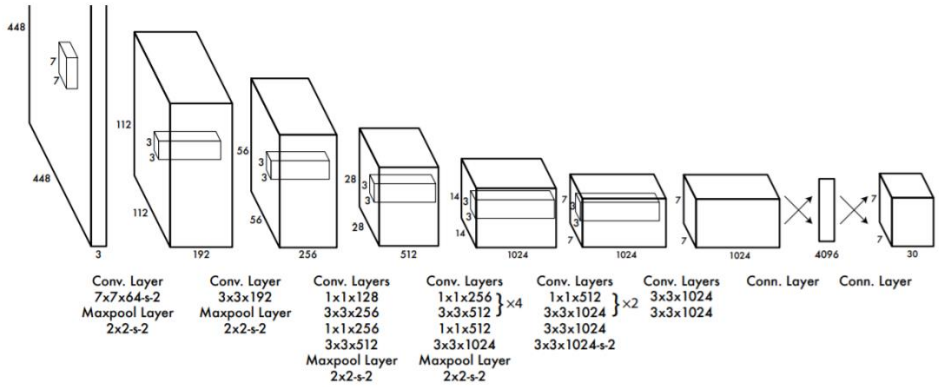

Figure 14 - YOLO's network design, inspired by GoogLeNet (Redmon et al., 2016).

One of the advantages of YOLO is that it can process at 45 FPS. Furthermore, if one uses its simplified version (Fast YOLO), it can reach 155 FPS. However, YOLO has difficulties in dealing with small objects in groups and generalization of objects in new or unusual aspect ratios (Redmon et al., 2016; Zhao et al., 2019).

2.3.6 Single Shot Multibox Detector

The Single Shot Multibox Detector (SSD) architecture is inspired by the anchors adopted in Multibox, RPN and multi-scale representation (W. Liu et al., 2016). Even though SSD functions similarly to YOLO, the former uses a set of default anchor boxes with different aspect ratios and scales, instead of fixed grids, as shown in *Figure 15*. This approach is used to discretize the output space of the bounding boxes and by fusing predictions from multiple feature maps with different resolutions, the network can handle objects of various sizes (W. Liu et al., 2016; Zhao et al., 2019).

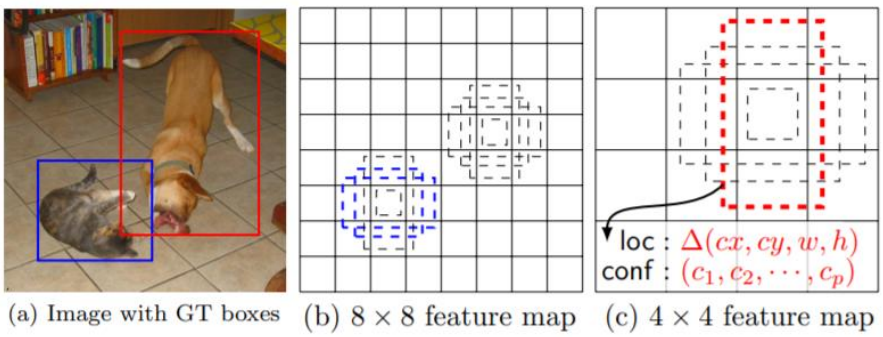

Figure 15 – Example of the anchor box construction of SSD (W. Liu et al., 2016).

Formally, as SSD is based on YOLO, it uses its standard architecture for high quality image classification as base network, excluding any classification layers, as shown in *Figure 16*. To this base network, the authors add 3 key features (W. Liu et al., 2016):

- **Multi-scale feature maps for detection**, which consists in the addition of convolutional feature layers to the end of the base network, which decrease in size progressively as to allow predictions of detections at multiple scales.
- **Convolutional predictors for detection**, where each added or existing feature layer can produce a fixed set of detection predictions using a set of convolutional filters. The resulting bounding box values are measured relative to a default box position, which in turn is relative to each feature map location.
- **Default boxes and aspect ratios**, where each feature map cell has a set of default bounding boxes associated. This is present for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional fashion, so that the position of each box, relative to its corresponding cell, is fixed. Specifically, for each box in *k* at a given location, SSD computes *c* class score and the 4 offsets relative to the original default box, resulting in a total of *(c+4)k* filters that are applied around each location on a feature map. In total, for a *m x n* feature map, *(c+4)kmn* outputs are produced.

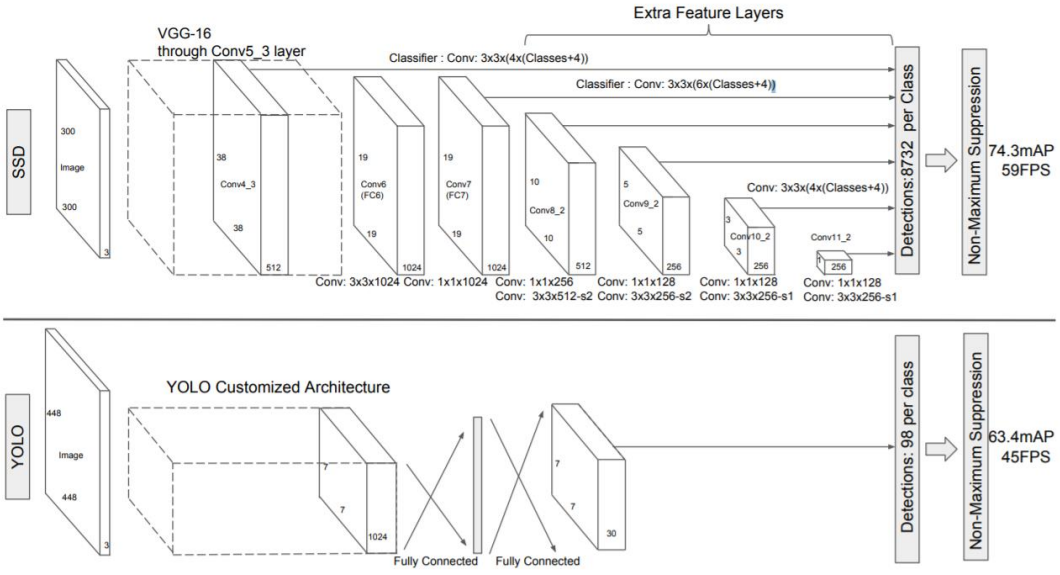

Figure 16 - SSD architecture. As shown in from the image, the architecture is based on YOLO's (W. Liu et al., 2016).

Comparing SSD to YOLO applied to images of 300x300, SSD outperforms YOLO with better accuracy at an average of 59 FPS. With tuned settings, SSD can even outperform Faster R-CNN. However, SSD cannot handle small objects by default, requiring a better feature extractor backbone (such as a ResNet101) in order to achieve this goal (Zhao et al., 2019).

2.3.7 Comparison of Relevant Frameworks

From the reviewed literature, the work published by Zhao et al. (2019) has a particularly detailed and exhaustive comparison of the several object detection frameworks such as Mask-RCNN, Faster-RCNN, SSD and YOLO, as well as insights on how these are implemented. *Table 1* summarizes the authors' findings when comparing the models on the VOC 2007, VOC 2012, and Microsoft COCO data sets.

The PASCAL Visual Object Classes (PASCAL VOC) data sets are a collection of standardised images for object class recognition. The project makes the data from the VOC challenges available for the public and is used as a reference for architecture and model performance comparison, as it provides a common set of tools for accessing the data sets and annotations. The project ran challenges from 2005 to 2012, as it is currently considered a finished project (The Pascal VOC Project, 2020).

The Microsoft COCO (Common Object in Context) is a data set like PASCAL VOC, which has the goal of advancing the state-of-the-art in object recognition. This is done by setting object recognition in the context of a scene's understanding (i.e. to also perform semantic segmentation). Objects in the COCO data set are labelled using per-instance segmentations, as to aid in precise object localization. COCO contains a total of 2.5 million labelled instances in 328 thousand images (Lin et al., 2015).

Table 1 - Comparison of different object detection architectures. The table depicts the ranking of each architecture in the tested data set. Multiple rakings represent different configurations of the architecture (Zhao et al., 2019).

Architecture	**VOC 2007 Ranking**	**VOC 2012 Ranking**	**Microsoft COCO**
Faster R-CNN	$3^{rd}/4^{th}/5^{th}$	7^{th}	13^{th}
Mask-RCNN	8^{th}	12^{th}	$3^{rd}/4^{th}$
SSD	$1^{st}/2^{nd}$	$2^{nd}/3^{rd}$	$2^{nd}/10^{th}/11^{th}$
YOLO	N/A	4^{th}	12^{th}

As shown in *Table 1*, it can be stated that SSD seems to be the overall best architecture when compared to the remaining architectures, having achieved top rankings. Furthermore, it can be stated that although Mask-RCNN had poor performance in the VOC data sets, it seems to have a satisfactory performance in the Microsoft COCO data set. Finally, since one of the caveats of the SSD architecture is its requirement of fine tuning the backbone in order to detect small object groups, special attention should be taken to assess its true feasibility for the object detection tasks encompassed by this master thesis' scope. From this analysis, SSD and Mask-RCNN architectures seem good candidates to be included in the development process of the models.

2.4 Signal Processing

In Signal Processing (SP), a signal can be defined as any phenomenon which transports information, such as human voices, smoke signals and sign language. Formally, it can be defined as a function composed of one or more independent variables (NPTEL, 2017). When the function is dependent on only one variable, as is the case of temperature & time, the signal is unidimensional. Likewise, when it depends on more than one variable, as is the case with images, the signal is multidimensional (NPTEL, 2017; Puolivali, 2013). A signal can be continuous when its domain is a set of real numbers (analogic signal), or, otherwise, discrete (digital signal).

SP is a technological area which consists in the discovery and application of signal processing techniques, which encompasses concepts such as the creation and modification signals, with the purpose of extracting information from the signals. SP tasks can consist of filtering, encoding, estimation, detection, analysis, recognition, synthesis and recording, and directly benefit from improvements in mathematics, statistics and areas of engineering (Nebeker, 2000; NPTEL, 2017; Puolivali, 2013). Starting from the 1960s, SP has been predominantly using digital techniques, and the area is present in several others, such as communication, information processing, control systems, medical diagnosis, and seismology.

As stated previously, SP can concern areas such as image processing, since images are an example of a multidimensional signal. However, since image related architectures are already described in 2.3, SP will be used in audio and temporal signals. As such, only audio-related DL techniques will be described in this section.

Regarding ML, the first applications of its techniques in SP consisted in an essential step: the extraction of audio features. Usually, the extraction of these features would be considered a separate problem to the classification/regression problem, as the features would have to be appropriately designed for the task at hand (Purwins et al., 2019). However, a manual approach of audio feature extraction could often yield non-optimal features for the task. For decades, the Mel Frequency Cepstral Coefficients (MFCC) where used as the dominant acoustic feature representation for audio analysis tasks. MFCC can be described as a reduced set of frequency bands obtained from a magnitude spectra projection, i.e. the representation of the short-term power spectrum of an audio signal. These frequency bands are usually converted to logarithmic magnitudes, and approximately whitened and compressed with a discrete cosine transform, which DL models have shown that the latter is unnecessary and/or unwanted, as it may remove significant information (Purwins et al., 2019).

With the advancements of DL, the audio feature extraction began to be considered alongside the task at hand (and not as a separate problem), as DL models like CNN can use a sequence of either frames of raw audio or human engineered feature vectors (Purwins et al., 2019). This particular trait eases the audio feature extraction task, making the extraction of some of the most known audio features not required for a classification or regression task. The evolution of DL algorithms in SP started with the use of CNN and RNN (as CNN have a limited effective

context size), proceeding to the use of LSTM and Sequence-to-Sequence models (Purwins et al., 2019). Sequence-to-Sequence is a problem setting (i.e. learning type) where both the input and outputs of the model represent sequences, such as the case of language translation (Sutskever et al., 2014).

Having into context the evolution and history of SP and DL, it is important to provide a description of Elman RNN's and LSTM's concept, including respective advantages and changes to the architectures. Elman Recurrent Neural Networks (ERNN) are known as *Simple RNN*, as they are defined as the most basic version of RNN, where their structures are simple and do not have any special features or characteristics, when compared to an ANN, apart from the fact that ERNN are prepared to deal with sequential data (Bianchi et al., 2017). ERNN have been applied in several natural language problems, such as learning grammar by generating successive words in a sentence. However, this type of architecture suffers from the exploding and vanishing gradient problems, making it often unsuitable for certain tasks where long range dependencies between sequences are needed to be learnt (Chang, 2018).

Long Short-Term Memory networks are a type of RNN architecture that address the vanishing and exploding gradient problems and can learn long-term dependencies, where the network is trained using backpropagation through time. The main concept of a LSTM is that instead of having simple neurons, it uses the idea of memory cells/blocks that are connected through layers (Brownlee, 2019c; Hochreiter & Schmidhuber, 1997). The block possesses components that make it smarter than a classical neuron, as well as a memory (cell state vector) for recent sequences.

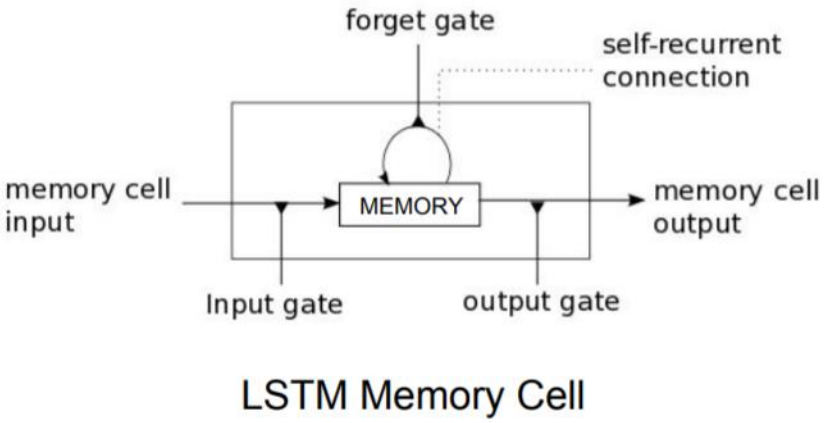

Figure 17 - Simplified architecture of a LSTM memory cell, with focus on the gates (Sood, 2017).

Furthermore, LSTM have the concepts of gating units, which regulate the flow of information into and out of the memory. Each gate within a block uses the sigmoid activation units to control whether they are triggered or not. Formally, a LSTM can have the following gates, as shown in *Figure 17* (Brownlee, 2019c; Hochreiter & Schmidhuber, 1997; Sood, 2017):

- **Forget Gate:** conditionally decides what information to discard from the block, essentially "forgetting" the information.
- **Input Gate:** conditionally decides which values from the input will update the memory state.
- **Output Gate:** conditionally decides the resulting output, based on the memory block and input.

As can be seen from the description, each unit essentially acts like a small-scale state machine where the weights of the gates of each unit are learned during the training process. The full architecture of the memory cell of a LSTM is shown in *Figure 18*.

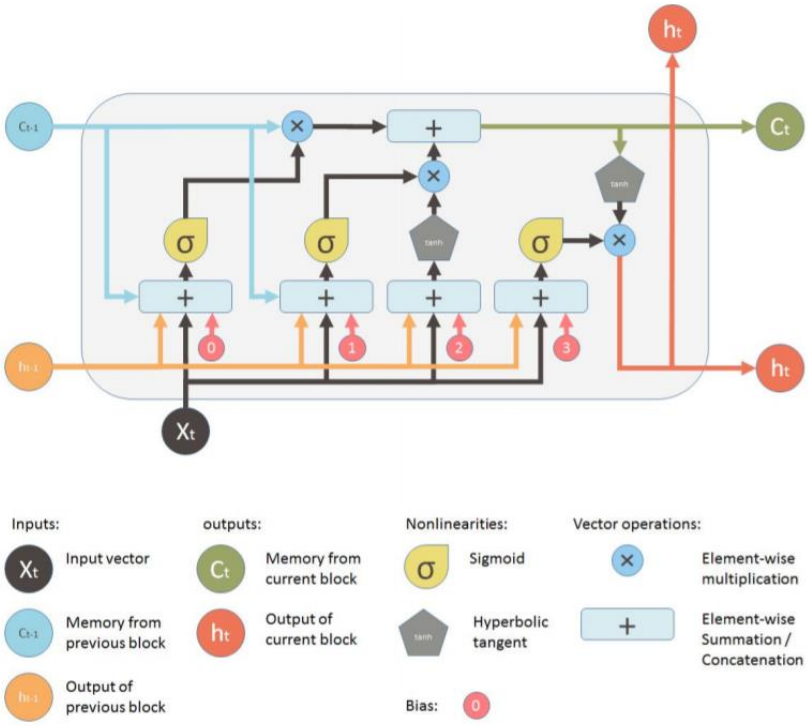

Figure 18 – Detailed architecture of the LSTM memory cell, including the inputs and outputs of the cell (Sood, 2017).

Due to its architecture, LSTM have allowed state-of-the-art results in several complicated problems where the learning of long range dependencies is required (Fischer & Krauss, 2018; S. Liu et al., 2019). Furthermore, even on simpler problems where ERNN and other types of RNN can be applied, LSTM still has either the best performance, or close to performance of the best model for the given task (Bianchi et al., 2017). Recently, LSTM-CNN architectures have also been deployed onto several sequence learning problems, as they take advantage of the automated feature extraction of the CNN and the LSTM's ability for long range dependency learning. Essentially, by combining both architectures (LSTM+CNN), the hybrid solution can obtain better results than traditional LSTM. Some applications of this hybrid architecture include the automated extraction of information and semantics in text (Sainath et al., 2015; X. Zhang et al., 2018).

2.5 Related Work

For this subsection, a review of ML in contributions related to honey bees will be described, as to assess the amount of related work available surrounding honey bee protection and preservation. However, the amount of work found surrounding BHMS is quite reduced when compared to other areas of research. Furthermore, from the survey conducted, no relevant literature was found surrounding the detection of *V. velutina* in images or through audio. The only work of interest found was the one conducted by Keeling et al. (2017), which depicts the expansion of the *V. velutina*. As such, all contributions that use ML will be reviewed, regardless of whether the contribution uses video, audio, or other data for the approach. Moreover, due to the resemblances between bees and wasps, the part of the review that concerns bees mentioned previously will be used to assess the features and data set conditions that can be used for the detection of Asian hornets.

In the literature, recent contributions have been made involving ML algorithms and bee health monitoring. The work done by Kulyukin et al. (2018) used 9110 audio samples, equally distributed by “Bee Sound”, “Noise Sound” and “Cricket Sound”, in order to monitor a bee hive. The approach used a CNN-based architecture, and was tested on the BUZZ1 and BUZZ2 (Utah State University, 2019) data sets against other types of ML & DL algorithms. The study concluded that DL can be used to monitor bees in a bee hive, with the CNN-based approach having obtained an accuracy of 95.21% and 96.53% on the BUZZ1 and BUZZ2 data sets, respectively. The study conducted by Amlathe (2018) analysed standard ML techniques (Random Forest, SVM, KNN and Logistic Regression) to perform classification of bee audio between 3 classes (bee, noise and cricket). The work concluded that ML based techniques can aid in the classification of bees using audio, as well as help monitoring a bee hive’s health. Nolasco & Benetos (2018) compared DL (DCNN) and ML (SVM with linear, radial basis function and 3rd order polynomial kernels) approaches for bee hive sound recognition. The study used an annotated data set of 78 recordings, comprising approximately 12 hours of audio. From the compared approaches, the study concluded that the SVM outperformed CNN. However, the authors also recognized that CNN showed room for improvement, highlighting the impact that samples with large context and the size of training data have on CNN. Florea (2013) used video and CNN for the automatic detection of honey bees in a hive, using approximately 11 hours of video footage, where bees have been placed with a tag. The best F1-score value achieved in the study was of 0.686 and the author concluded that the manual labelling provided by the tags may not suffice for bee detection. Both Chazette et al. (2016) and Schurischuster et al. (2016) used CNN and ML algorithms to perform bee hive monitoring, aiming at the detection of mites and *varroa destructor*. The first study obtained an accuracy of 93% detection of *varroa destructor* using a training data set of 5000 artificially generated images, tested with different CNN configurations (Chazette et al., 2016). The second study used different configurations of lights in a special camera setup, using 1920x1080 images recorded at 50-60FPS. The authors highlighted the challenge in detecting mites on moving bees or on bees where wings occlude the mites (Schurischuster et al., 2016). The work done by Pukhov (2018) provides a thorough explanation and exploration of the data set publicly available in (J. Yang, 2018a). The approach consists of 2 CNN models to perform the

classification of a bee's health and subspecies. The author provides several methods to analyse and remove bias from the data set, as well as a solid pipeline for the modular training of the CNN models (Pukhov, 2018). Furthermore, using the data set provided by Yang (2018), Pukhov was able to obtain a best accuracy of 84.92% and 86.54% on the health and subspecies models, respectively.

Finally, some projects about bee health monitoring can also be identified in the literature. Although these do not have specifics regarding the implementation details nor the methodology or approach used, they still represent acknowledgeable related work. These projects also use AI techniques and some even possess connections with the European Commission (IoBee, 2020; The Portugal News, 2019).

2.6 Technologies and Frameworks

As stated previously, the goal of this masters' thesis is to develop a prototype of the solutions for a BHMS, using images and audio of honey bees and related relevant data. From the reviewed architectures, it is important to assess the technologies that are currently available which allow for the architectures' proper implementations, as well as the development of models based on the algorithms and respective evaluation. As such, the following subsections will provide insights to the different languages, tools and frameworks that are available for the context of this thesis' work, as to assess which should be used.

2.6.1 Python

Python (Python Software Foundation, 2020), originally developed by Guido van Rossum in 1991, is a modern and powerful programming language with similarities to the Fortran programming language, despite being much more sophisticated. Its use and distribution are free, and it is also possible to study and make changes to the source code of the language (Doty, 2008). Python is not a strongly typed language, allowing the use of variables without declaring them and determining the type implicitly. It also uses indentation as a control structure, so braces are not used to define blocks of code. The Python programming language also does not require the definition of classes, although it is possible to define them whenever needed or convenient to do so (Doty, 2008). This language is considered good option for mathematical calculations, since the syntax is like the way mathematical ideas are expressed in the literature and the language allows you to write and test the code quickly and easily. It is also possible to identify the applicability of Python in web projects (Doty, 2008). Furthermore, Python is one of the most widely used languages in the scientific field due to its ease of learning and use (Cass, 2016). The increase in popularity of this programming language seems to stem from Google's investment in it (L. Kim, 2015). Python is also the most used language for ML and DL, with 57% of data scientists and ML developers using it worldwide, where 33% prefer it over other alternatives. The language assures its leader position primarily due to the release of TensorFlow, PyTorch and a wide selection of other libraries (Bansal, 2019).

2.6.2 R

R (The R Foundation, 2020) is defined as a statistical computing and graphics system, with features such as a programming language, high-level graphics, interfaces with other languages and debugging capabilities (Palik, 1998). This programming language is a dialect of the S language, originally designed in the 1980s, being possible to identify a great use in the statistical community since its creation. The syntax of the R language has superficial similarities with the C language, however the semantics follow the variety of functional programming languages, with great affinity to Lisp and APL (A Programming Language) (Palik, 1998). R allows the user to write functions that accept expressions as an input parameter, being very useful in statistical modelling and graphics. Certain level of objectives can be completed using only the interactive R, through the execution of simple expressions. For the use of new features and systematization of repetitive work, it is necessary to develop functional packages and functions for the intended purpose, respectively (Palik, 1998).

2.6.3 Keras

Keras can be described as a high-level neural networks API which enables its users to develop and implement several neural networks in a high-level fashion. Keras is written in Python and can use several low-level frameworks as a backend for the deployment of models, such as TensorFlow and Theano (Chollet, François et al., 2015). As stated by its slogan *"Being able to go from idea to result with the least possible delay is key to doing good research"*, Keras' development exhibits special focus on enabling fast experimentation. Models can be described either through files or in pure Python, and since these are understood as sequences or graphs, they are highly modular and easily extensible, whilst being user friendly at the same time. Keras is the second most popular framework and is generally viewed as an "API designed for humans, not machines" (Hale, 2018).

2.6.4 TensorFlow

TensorFlow is an end-to-end source platform for ML, developed by Google. It has a comprehensive and flexible ecosystem of tools, libraries and community resources that enables researches to push the state-of-the-art in ML even further (Google, 2020). TensorFlow is viewed as the most popular framework. From an analysis of *Figure 19* and *Figure 20*, it can be concluded that although Keras is the most searched API on Google, it has been majorly sought out by users in Indonesia, Brunei and others. In other countries, TensorFlow is the most searched API (Hale, 2018). This popularity arises because of its integration with Keras, but also due to the following facts (Forsvarets Forskninginstitutt, 2018; Google, 2020):

- Considered to have the right level of abstraction, which is good for both research and production purposes.
- TensorFlow can run on different devices without extra work for adaptation (cross-platform).

- Has a lot of functionality, although sometimes it has too many functionalities, many of which end up not getting used, cluttering the API.
- Has a light-weight version called TensorFlow Lite, which can be run on small embedded devices. TensorFlow itself can also run on huge clusters.
- Resource availability and a lot of examples and pretrained models.
- TensorBoard, which is a visualization tool and platform to view metrics, progress, and trials of run models, as well as the architecture of the model.
- Models are built using a graph notation, allowing for an easy model construction.
- Can run on GPU.
- Can be used with several languages.

Figure 19 - Evolution of the interest on Google's search engine for the terms "keras", "tensorflow" and "pytorch" (Google Trends, 2020).

Figure 20 - Interest of the terms "keras", "tensorflow" and "pytorch" on Google's search engine, by region. From the image and the graph shown in Figure 19, Indonesia has a lot of users using Keras, but worldwide, TensorFlow is the most used (Google Trends, 2020).

2.6.5 PyTorch

PyTorch can be defined as an open source ML library for Python, developed by Facebook's artificial intelligence research group (TutorialsPoint, 2019). It also contains Uber's Pyro software for the probabilistic programming on it. PyTorch describes itself as *"An open source machine learning framework that accelerates the path from research prototyping to production"* (PyTorch, 2020), and it is currently the third most popular ML & DL framework worldwide (Hale, 2018). PyTorch has the following set of features and characteristics (PyTorch, 2020; TutorialsPoint, 2019):

- Easy tool for toy and research projects.
- Is viewed as a clear "pythonic" API, i.e. follows the natural Python way of developing.
- Provides a platform for implementing models as dynamic computational graphics, enabling the user to alter them during runtime (useful when memory constraints are unknown beforehand).
- Has three levels of abstraction:
    - Tensor – Imperative n-dimensional array which runs on GPU
    - Variable – Node in the computational graph that stores data and gradient
    - Module – Neural network layer which stores state or learnable weights
- Includes lots of loss functions and functionality.
- Can run on GPU.

2.6.6 Technology Selection Decision Process

From the reviewed technologies, the following set of frameworks, languages and tools were selected for the development of this thesis' work: Python and Keras with TensorFlow backend. The reasoning behind the decision process for the language resides in the fact that most frameworks and API for DL development were originally developed in Python. Thus, more documentation, examples and support are provided and available for Python, as opposed to R. Furthermore, Python is seen as the go-to language for the development of DL, as all models developed in this language can be easily integrated into production. Regarding the framework chosen, Keras and TensorFlow are the most popular API for development of ML. Since Keras supports TensorFlow by using it as a backend, advantages from both architectures can be obtained. Furthermore, due to their popularity, the number of examples, documentation and support available is quite significant, which is beneficial for the development of complex models and deployment of these models into production environments.

2.7 Summary

In this chapter, a review of the literature surrounding the thesis' scope was described. DL concepts and contextualization were provided, as to establish a common-ground of knowledge for the next chapters. Related work surrounding bee health monitoring and bee

keeping using AI was provided, with examples of projects that are currently being developed and can act as direct competitors for the solution's opportunity of this work. Finally, a review of the languages and tools to be used was provided, as well as the rationale behind the decision process of which languages and tools to use.

3 Value Analysis

One of the most significant metrics to measure corporate success is a company's value creation. Innovation allows companies to gain competitive advantage and generate shareholder value. However, strategic innovation is a very complex task and it is essential that innovation is properly designed and launched in a timely fashion, as to generate customer value (Špacek & Vacík, 2016). Usually, companies adopt one of the following strategies: **demand-pull**, which seeks to find challenges from clients and develop new solutions for their needs, and **supply-push**, which starts by developing a product or innovative solution and then finds the target market for it (Cardia & Andrade, 2020). In these approaches, the company can find that the required solution has constraints on its development or that there is not a target market for the specific product developed. The company will effectively waste resources if any value creation ends on any of the previously mentioned scenarios.

In this chapter, the value analysis of this master thesis is described, in order to assess the potential value of this work in the problem's context. The various concepts encompassed by value creation will be described, including the innovation concept development and value of the solution. Furthermore, Quality Function Deployment will be described and used in order to aid in the design decision process on different possible solutions.

3.1 Innovation/New Concept Development

Innovation can be described as the result of a change that adds value to a specific target. As defined by the Oxford (Oxford Learners Dictionaries, 2020a), innovation is *"the introduction of new things, ideas or ways of doing something"*. Yet, when relating to the definition of the term from a business perspective, it has a more detailed definition. According to the third edition of the Oslo Manual (OECD, 2005), innovation can be defined as *"(…) the*

implementation of a new or significantly improved product (good or service), or process, a new marketing method, or a new organizational method in business practices, workplace organization or external relations". The definition provided by the Oslo Manual highlights the important aspects of innovation in business: for an idea or concept to become an innovation, an analysis of the added value, perceived by the targeted customers, and a proper plan for its implementation and concretization are required. In sum, an idea or concept becomes an innovation when it answers its customers' needs or wills and is viewed, by the customer, as a positive trade-off between the added value and the cost for obtaining the solution, generating benefits to the client and the organization. In this thesis, in order to conduct the value analysis of the thesis' work, the model New Concept Development (NCD), proposed by Koen et al. (2002), will be used.

On the topic of the NCD, a brief overview of its origins is necessary in order to understand the reasoning of its creation. According to Koen et al. (2002), the innovation process can be divided into three main areas, as shown in *Figure 21*: Fuzzy Front End (FFE), New Product Development (NPD) process and commercialization. According to Koen et al. (2002), FFE is generally viewed as one of the greatest opportunities for improving the overall innovation process, as the structured nature of NPD has already been dramatically improved by many companies.

Table 2 - Main differences between FFE and NPD. Adapted from (Koen et al., 2002).

Metrics	**Fuzzy Front End (FFE)**	**New Product Development (NPD)**
Nature of Work	Experimental, often chaotic. "Eureka" moments. Can schedule work—but not invention.	Disciplined and goal-oriented with a project plan.
Commercialization Date	Unpredictable or uncertain.	High degree of certainty
Funding	Variable—in the beginning phases many projects may be "bootlegged," while others will need funding to proceed.	Budgeted.
Revenue Expectations	Often uncertain, with a great deal of speculation.	Predictable, with increasing certainty, analysis, and documentation as the product release date gets closer.
Activity	Individuals and team conducting research to minimize risk and optimize potential.	Multifunction product and/or process development team.
Measures of Progress	Strengthened concepts.	Milestone achievement

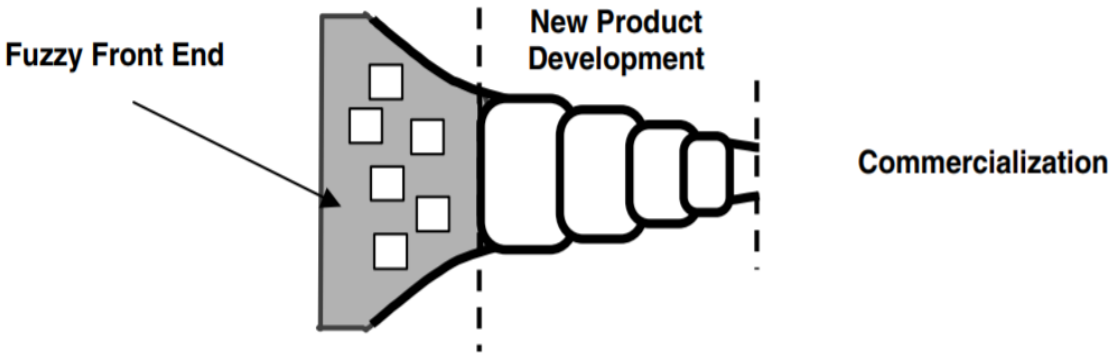

Figure 21 – Architecture of a standard innovation process. As can be seen, the division between FFE and NPD is often less than sharp, as technology development activities may need to be conducted at the intersection (Koen et al., 2002).

As shown in *Table 2*, FFE is unstructured in nature, making it hard to compare FFE approaches in different companies. Furthermore, due to this nature, the typical sequential approach used in NPD cannot be properly applied to the FFE (Koen et al., 2002). In order to increase the value, amount and success probability of high-profit ideas that are entering the NPD and commercialization, Koen et al. (2002) devised a set of methods, tools and techniques for managing the FFE of companies.

From the proposed methods, Koen et al. (2002) developed the NCD, which intends to solve issues such as the lack of common language and vocabulary (by creating a common terminology) and the application of a sequential model to an unstructured problem (by providing a nonsequential relationship model), among others. On this scope, Koen et al. (2002) provide the following set of definitions:

- **Opportunity:** *"A business or technology gap, that a company or individual realizes, that exists between the current situation and an envisioned future in order to capture competitive advantage, respond to a threat, solve a problem, or ameliorate a difficulty."*.
- **Idea:** *"The most embryonic form of a new product or service. It often consists of a high-level view of the solution envisioned for the problem identified by the opportunity."*.
- **Concept**: *"Has a well-defined form, including both a written and visual description, that includes its primary features and customer benefits combined with a broad understanding of the technology needed."*.

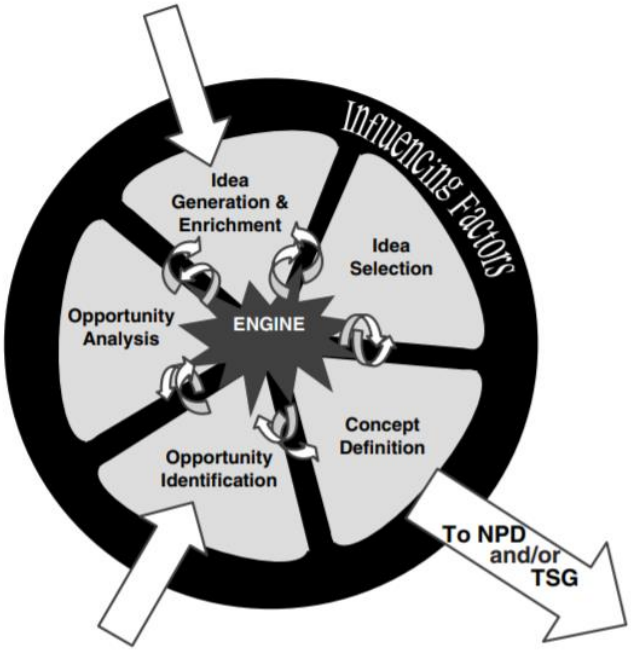

Figure 22 – The New Concept Development (NCD) construct proposed by Koen et al. (2002). As can be seen, it is a relationship model that provides a common language and definition of the key components of the FFE.

The NCD, as seen in *Figure 22*, is a relationship model, unlike the standard linear process, and is composed of the following three main sections (Koen et al., 2002):

- **Engine (Bull's-Eye portion):** corresponds to the *"leadership, cultural and business strategy of the organization that drives the five key elements that are controllable by the corporation".* Essentially represents senior and executive level management support and powers the five elements of the NCD model.
- **Inner spoke area:** defines five controllable activity elements of the FFE, which are the opportunity identification, opportunity analysis, idea generation and enrichment, idea selection and concept definition. The inner parts are called elements, as opposed to processes, since the later demand a structure which may not be applicable. Further insights on elements relevant to this thesis are provided in subsequent sections.
- **Influencing factors:** mainly composed of organizational capabilities, the outside world (i.e. distribution channels, laws, government policies, customers, competitors, and political and economic climate), and enabling sciences (internal and external) that may affect the entire innovation process through to commercialization. Influencing factors are relatively uncontrollable by the organization.

Furthermore, the NCD model has a circular shape, as to indicate the circular and iterative flow of ideas between all five elements.

3.1.1 Opportunity Identification

This element has the organization identifying the opportunities that it may want to pursue. Examples of an opportunity in the context of this element include a near-term response to a competitive threat, a "breakthrough" possibility aimed at capturing competitive advantage, a simplification of operations with an aim at increasing speed, reducing costs, or others or a new direction for the business, among others (Koen et al., 2002). Essentially, this element defines the sources and methods used to identify opportunities to pursue, defining the market or technology arena the organization may want to participate in.

On the context of this master thesis' scope, the opportunities identified are as follows:

- Need for automation of the hive inspection and check-up process (Beesource Beekeeping Forum, 2017).
- Necessity for increased frequency in the health inspection of hives, in a remote fashion.
- Need for solutions that allow a more sustainable and scalable evolution of the core activity of a bee-related business (e.g. honey production) – sustainable survival of bees.

The opportunities defined above arise around the fact that, for most companies and beekeeping enthusiasts, the main activity developed is honey production, which in turn depends entirely on the success of honey bee hives. Any threats to these hives can severely impact the expected yield of a hive, demanding constant monitoring of vital health signs. However, as manual human check-ups induce stress in bees, too many check-ups can cause a decrease in expected yield, due to increased stress and overall health population decline. As such, there is a high demand for monitoring solutions that can help with the assessment of a hive's health, without inducing stress. Additionally, from discussions held with companies and experts in the field, it can be stated that the AI solutions must have an accuracy around 80% or higher due to the volume of the data processed, as well as the time for which these solutions will be operating. For example, in the case of detecting bees and Asian wasps in a threat detection context, it is expected for a hive to have around 200 bees. Having an accuracy of 80% means that the AI solution would fail to detect 40 out of 200 bees, multiplied by the amount of time it has been operating. Realistically, such an AI may fail to detect well over 2000 bees or Asian wasps in one day. Similarly, the elapsed time of the AI solutions should also be less than second due to the volume of data processed. Considering the previous example, if the AI solution takes one second to process a single bee, then the AI solution will take 200 seconds (3 minutes and 20 seconds) to process all bees in a single hive. Due to the volume of data and potential implementation cost of such systems, it only becomes profitable for a company if the AI solution reduces or optimizes a workload significantly (i.e. the solution should optimize the process, instead of complicating it with further necessity of validating the produced results). These conditions led to the opportunity stated in this thesis, which is the development of AI solutions that can be used in the development of a BHMS using video data along with SP and CV techniques.

3.1.2 Opportunity Analysis

In this element, opportunities are assessed in order to confirm that they are worth pursuing, requiring additional information for the translation of the opportunity identification into specific business and technology opportunities. In this element, early and often uncertain technological and market assessments are involved, and extensive effort maybe be committed for focus groups, market studies and scientific experiments, depending on the value of the information associated with reducing uncertainties about the attractiveness of the opportunity. Opportunity analysis can be part of a formal process or occur iteratively.

For this element, a SWOT analysis was used (Dyson, 2004). Even though the SWOT analysis is typically applied on a strategy formulation context, it generally describes the advantages and disadvantages of a target concept, from an internal and external perspective, i.e. advantages and disadvantages that are dependent and independent of the concept. The SWOT analysis, shown in *Table 3*, highlights the following topics regarding the BHMS that could be developed using this thesis' solutions:

- **Strengths:** taking into consideration the opportunity identified in the previous element, it can be stated that a system can be developed for the automatic detection of signs of health issues in bee hives through video. This system will be using cameras so that it can perform monitoring of a set of hives in a remote fashion, with increased frequency. As the system will be using video data, it can be reviewed by experts if needed, as to assess whether a given health issue is accurately true.
- **Weaknesses:** the proposed system would have a set of costs, both to the company and to the customer. Regarding the company, it would require development resources, such as hives, electronic boards (e.g. Arduino), cameras, bees for testing, planning, developers, and data sets. Regarding the customer, the solution would always have inherited costs in terms of acquiring the correct camera setup for the target bee hives, and it would also require user training in order to properly use the system.
- **Opportunities:** the potential of the solution can be increased using new state-of-the-art techniques that show promising results in object detection and SP. Additionally, the contributions made towards the understanding of fundamental bee health signs, and how these impact a colony's health, is also beneficial to the solution.
- **Threats:** factors that may pose as a threat to this opportunity include projects that exploit a similar opportunity (Pollenity, 2020; The Portugal News, 2019). Furthermore, on the lower chance of occurring, the creation of new technological advancements that may make the current solution's design obsolete also presents itself as a threat, as is the case with robotic pollinators and artificial honey production, which can make beekeeping obsolete (HealthyWithHoney, 2018; The Guardian, 2018).

Table 3 - SWOT analysis of the master thesis work's opportunity.

Strengths	Weaknesses
• Ability to perform remote inspections • Higher frequency of inspections • Uses video and audio, which can be manually reviewed by experts if necessary	• Solution price • Development costs and resources • Possible user training required in order to understand and work with the system
Opportunities	**Threats**
• New state-of-the-art technologies that provide potential for the system's development • Consideration of contributions towards the deep comprehension of bee health issues and respective impacts	• Development of competitor projects that aim towards the same opportunity • Low chance regarding the creation of new technologies that make the system obsolete (such as robotic pollinators and artificial honey production)

From the conducted SWOT analysis, it can be stated that the opportunity, created by the needs described in the previous element, is worth pursuing, as it is likely for it to provide a significant contribution towards the sustainable survival of honey bee colonies. This in turn validates the necessity for the development of AI solution that help interested developers and researchers to develop the BHMS, as is the case of this thesis' proposed solution.

3.1.3 Idea Generation and Enrichment

This element concerns the birth, development, and maturation of a concrete idea. It is expected for the idea to undergo many iterations and changes (e.g. built up, torn down, reshaped), as it is examined and developed in conjunction with other elements of the NCD. This element can use a formal process, including brain-storming sessions and idea banks (Koen et al., 2002).

For this element, taking into consideration the literature review conducted in Chapter 2, as well as the work conducted in the previous elements, the following set of idea enrichments were identified:

1. Implementation of DL architectures from scratch.
2. Use of open-source libraries that contain the implemented architectures for the specified tasks.
3. Implement a system based on weights scales.
4. Use GPS equipped cameras.

3.2 Value of the Solution

This section aims at providing definitions and context for the value of the solution provided in this thesis. For this end, the value, customer value, perceived value and longitudinal perspective value will be defined on the context of this master thesis' scope.

3.2.1 Value

The term "value" can exhibit different meanings depending on the context and area it is being used. Oxford defines the term as "*how much something is worth in money or other goods for which it can be exchanged*" (Oxford Learners Dictionaries, 2020b). On the context of value analysis and business, value refers to the same concept, but can have different meanings depending on whether it is referring to the producer or consumer side (Lindgreen & Wynstra, 2005). On the producer's side, the value can be derived from factors such as the value of the loyal customers, which often corresponds to a beneficial view to the business. On the consumer's side, the value can be dependent on factors such as the relationship with the producer/seller, the quality of the acquired product/service, social reactions from the acquisition of the product and even scenario dependent factors (e.g. place, time, projected objectives, etc.). For the value of a product to be improved, it must consider the following three elements (Nicola, 2020):

- **Use value**, which concerns the useful/functional aspect of the product.
- **Esteem value**, which comes from the ownership of the product, whose characteristics are attributed by the user. These are usually aesthetic and subjective in nature and do not directly contribute to utility.
- **Market value**, which describes what market would pay for the product and can be viewed as the sum of utility and esteem values.

Regarding the value of this thesis, it can be defined according to different perspectives, which will be detailed in the next sections. Broadly defined, the value of this thesis' contribution is the discovery, description and development of reliable AI models that help towards the development of a system for bee health problems monitoring, as well as a suggestion of an architecture for the implementation of the BHMS, as to aid in the sustainable survival of bee health colonies and depending ecosystems.

3.2.2 Customer Value

Customer value can be defined as the concept that provides answers to questions such as "What do customers really want and how does the producer meet their demands?" and "What do customers really value?". Therefore, customer value refers to the ability of organizations in creating and adding value to good and services, especially to those offered to customers, or service aspects of the organization's business. The delivery of customer value is based on four components (Mcfarlane, 2013):

- **Service:** intangible value offered to customers.
- **Quality:** client perception that the company's products and services meet expectations.
- **Image:** customer perception of the organization they interact with.
- **Price:** the cost that the organization can command for their goods and services and that their customers are willing to pay.

On the scope of this thesis, the customer value is described by the potential uses and problem solving of the developed prototype and architecture. From the literature review, the currently proposed solutions for bee health monitoring do not have a high reliability for their implementation in a real-life scenario. Furthermore, the many threats bee colonies face demands the development of a remote, near real-time monitoring system. The solution proposed in this thesis' work aims at developing a system prototype that meets these expectations.

3.2.3 Perceived Value

The customer perceived value can be summarized as the difference between the prospective customer's evaluation of all benefits and all costs of an offering, when compared to perceived alternatives (Kotler & Keller, 2012). Therefore, perceived value is the difference between the value that the customer expects to acquire (customer value) and the costs that the customer believes the product/service is worth, as per below:

$$CustomerPerceivedValue = Total_{CustomerValue} - Total_{CustomerCosts} \quad (6)$$

For the solution proposed in this thesis, the perceived value is defined as the difference between the benefits obtained by the proposed solution, i.e. effective and remote bee health monitoring near real-time, and the costs associated with the development and implementation of the solution, such as the human resources, knowledge of frameworks used, implementation testing on the desired context and computational environment required, among others.

3.2.4 Longitudinal Perspective of Value

Longitudinal perspective is defined regarding the customer value, with the goal of describing the latter in a temporal and cumulative fashion. The perspective can be divided into four phases, where each contributes for the overall value attributed by a customer to a product. As it is cumulative, the success of previous phases can affect the remaining. The phases are as follows (Messner, 2013; Woodall, 2003):

1. **Ex Ante (Pre-purchase):** ideas that the customer may have about the value that they expect to experience, before purchasing the value.

2. **Transaction (At the point/moment of trade):** when a sense of value is experience at the point of sale (e.g. buying an expensive car might feel valued through good treatment in the showroom).
3. **Ex Post (Post-purchase):** the moment when subject-based use value is experienced, after completing the purchase decision and when the product/service is used.
4. **Disposition (After use):** occurs when the product/service is decommissioned. The experience value may continue to last.

Table 4 provides the benefits and sacrifices of the proposed solution associated with each phase defined by Woodall (2003). As can be seen, there are several distinct benefits and sacrifices associated to each phase, with some directly related to the development and implementation of a BHMS based on the solutions provided in this thesis' work. As the stakeholders correspond to researchers, developers and all those who are interested in deploying a BHMS, the sacrifices are majorly associated to the efforts in understanding, developing, and implementing the models provided in this thesis. Regarding the benefits, special note is taken on the Disposition phase. In case that the models and technologies provided in this solution are obsoleted by another innovation, the solution can still be used as a starting point or comparison for future innovations, as is the case with most researchers' work. Essentially, the solution can be beneficially used to compare or review the approach and results of future innovations.

Table 4 - Benefits and sacrifices of each phase of the longitudinal perspective value.

Phase	Benefits	Sacrifices
Ex Ante (Pre-purchase)	Expected increased efficacy and efficiency in the bee health monitoring process	Efforts in searching and understanding the proposed solution, as to take full advantage of its implementation
Transaction (Moment of trade)	Acquisition of a solution that can be modified as needed and can provide high quality and reliable information, as well as proper documentation on the proposed solution	Efforts and costs related to the development and implementation of the solution in the customer's specific context and environment
Ex Post (Post-purchase)	Competitive advantage regarding competitors who do not have technology like the one found in the proposed solution, as well as a boost in performance for apiculture related activities	Costs related to the upkeep of servers or related computational environment needs, as well as management of generated data
Disposition (After use)	Proposed solution can be reviewed to gain insights on new and improved approaches, through comparison of technical details (e.g. advantages and disadvantages of certain technologies)	Costs of upgrading or changing infrastructure and/or system, in the case the technology is made obsolete

3.3 Value Proposition

The value proposition can be defined as an explicit promise made by an organization to its customers, where it will deliver a particular bundle of value creating benefits, which can be viewed by the customers as superior, equal or inferior to alternatives (Hassan, 2012). The value proposition canvas is a tool that helps assessing the underlying value proposition of a solution, by providing a simple graphical way to describe customer needs, benefits and pains (The Warren Johnson Society, 2020).

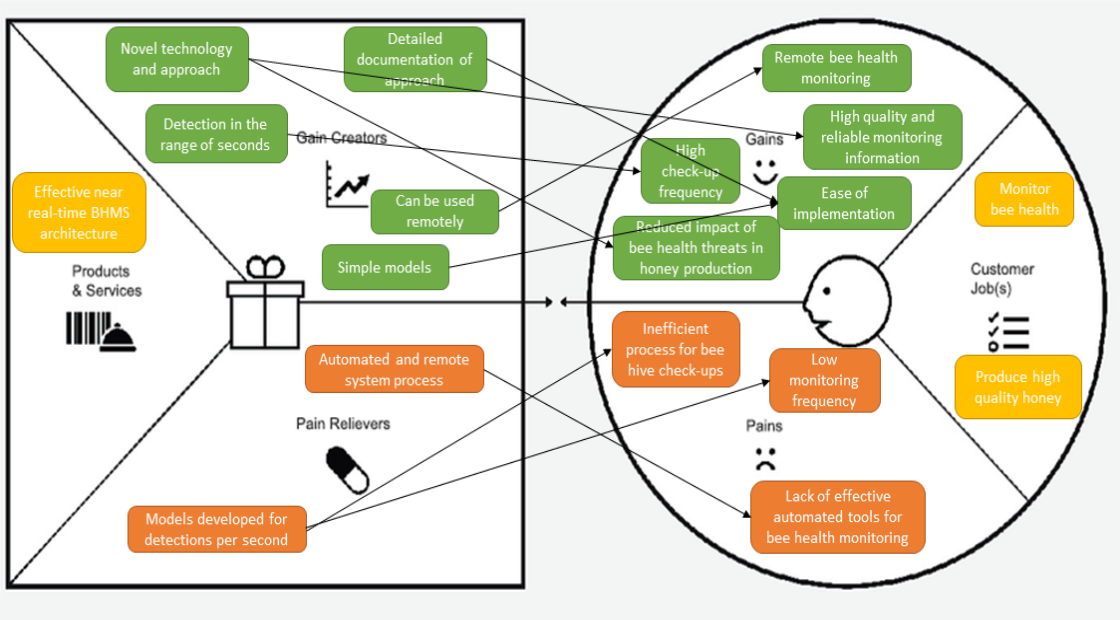

Figure 23 – Value proposition canvas of the proposed solution. Adapted from (Strategyzer, 2016).

Considering the value proposition canvas of this thesis' solution shown in *Figure 23*, it can be stated that the value proposition of the thesis' proposed solution is an effective near real-time BHMS architecture that can help the customer monitor bee health and boost the quality of honey. The solution does this by relieving the pain associated with the inefficiency in bee hive check-ups, low monitoring frequency and lack of effective automated tools for BHM through the models that are developed to output detection in less than a second, in an automated and remote fashion. The customer benefits from the high check-up frequency, reduced impact of bee health threats in the honey production, ease of implementation and high quality, remote and reliable health monitoring provided by the solution's novel technology and approach, and respective detailed documentation.

3.4 Quality Function Deployment

Quality Function Deployment (QFD), first developed in Japan by Yoji Akao in the late 1960s while working for Mitsubishi's shipyard, can be defined as a structured approach to effectively define customer requirements and translating them into detailed engineering specifications and plans to produce products that meet those needs (Hauser et al., 2010; Kenneth, 2011; Quality-One, 2020). The customer requirements are usually called Voice of the Customer (VOC), and these requirements can be captured in several ways such as direction discussion, interviews, surveys, focus groups and observation (Kenneth, 2011; Quality-One, 2020).

For this thesis' work, the VOC was obtained from interviews with experts in the field, as well as through observation of beekeeping forums and other publicly available information about the opinions and requirements of the customers. The VOC can be summarized as follows: clients find that the most important requirements are the quality and speed of the detection. However, the quality of the detection is deemed as far more important than the speed of the detection, as a product that is fast but does not produce relevant information is seen as useless (the common statement used for to describe such as product would be "Even a broken clock is right two times a day"). On the context of the solution, the VOC relates to the following technical requirements:

- Elapsed time (amount of time between each detection)
- Accuracy (percentage of correctly classified examples over total examples)
- Computational Environment to Run Models (the hardware and software requirements to run the models)
- Model Complexity (how complex the model is)

3.4.1 House of Quality

Within the QFD, cross-functional teams implement it by creating a series of one or more matrices. The first is called the House of Quality (HoQ). It is a tool that allows for direct comparison of the way the design or product stacks up to the competition in meeting the VOC and benefits the design decision process (Kenneth, 2011; Quality-One, 2020).

Figure 24 illustrates the QFD's HoQ developed under this thesis' scope, alongside appropriate legends of symbols used. It is also important to note that the competition sections of the HoQ were cut and blacked out. The reason behind this is that under this thesis, the evaluation of the competition, in accordance to the solution provided, would not be executable, as the solution/methodology has yet to be developed. Since there is not a solution to showcase to any potential customers, it is not accurate to performance a competitive analysis and comparison between the approach/solution developed in this thesis and ones found on other projects. Despite this fact, the HoQ is still used to highlight and assess important features of this master thesis' solution.

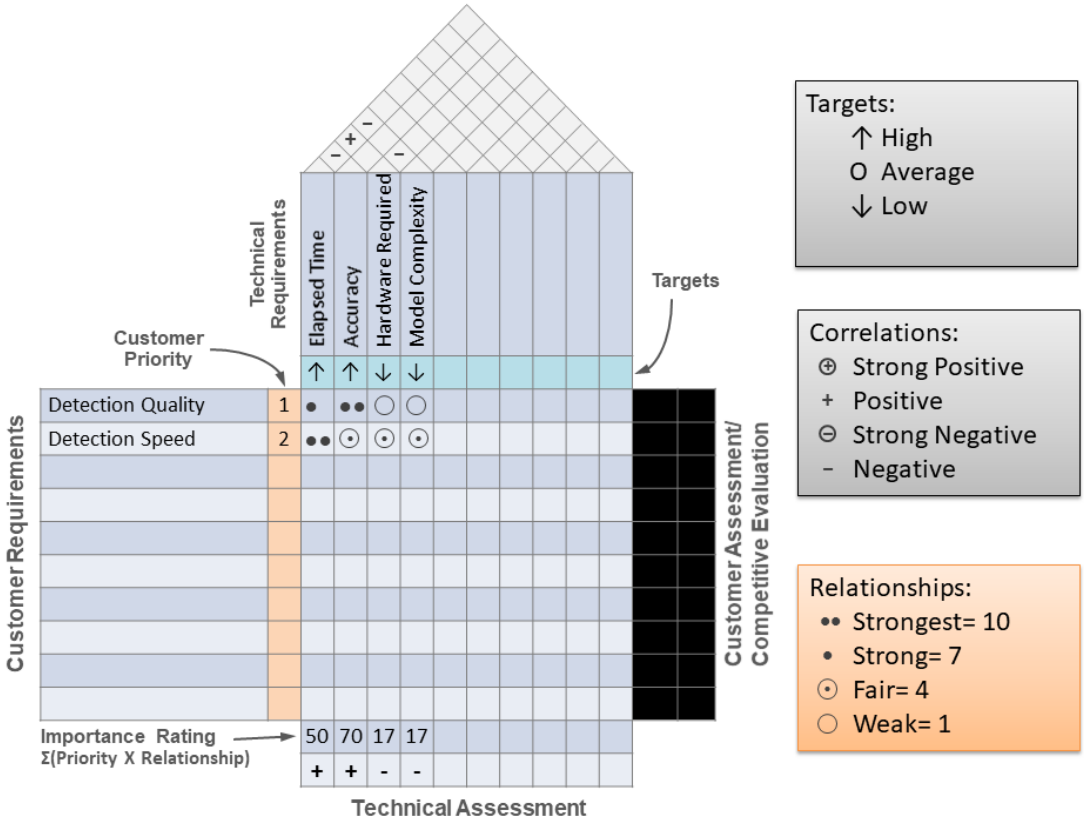

Figure 24 - House of Quality of the solution. Competitive analysis was blacked out as it was not feasible and reliably possible to do so under this thesis' scope.

From the VOC gathered and the developed HoQ, it can be stated that the top priorities for the target clients are the quality and speed of the detections. Furthermore, having into account their relationship with the technical requirements, it can be stated that all design choices of the architecture and models to be developed under this thesis should prioritize accuracy and elapsed time over the hardware requirements and model complexity. However, the best solution would be one that can maximize accuracy whilst reducing elapsed time, model complexity and hardware requirement.

3.5 Summary

In this chapter, the analysis of the value of this thesis' solution was described. The process of innovation was described, as to contextualize the value analysis under this thesis' scope, with the specification of the opportunity and its respective analysis. Furthermore, the value proposition of the solution was defined. Finally, using QFD it was possible to assess the most important features to consider during the design and development of the solution's models.

4 Design

In software engineering, design can be considered as the act of creating artefacts in order to reach goals. As all definitions of software engineering explicitly or implicitly imply a concern for the production, use and maintenance of software, the design activity can be seen as the one that best fits the general characterization of software engineering, being the central integrating activity that ties others together. System analysis, specification, detailed design, programming, cost estimation, management and other software development activities directly involve the design of programs or demand the engineer to have a clear understanding of the design process behind the programs they are developing (Freeman, 1976).

In this chapter, the design process for this thesis' solution will be described, considering the requirements and objectives of this thesis' work. A specific listing of the requirements of the system will be provided, as well as alternatives found for the implemented solution, including the final selected design. Furthermore, although the goal of this thesis is not the development and deployment of a software/product, but rather an architecture and modules that help with the development of a solution, design alternatives for the potential system that could be implemented with this thesis' solutions will still be provided, as well as the recommended best architecture, as to provide a reference for any stakeholders.

4.1 Requirement Gathering

System requirements describe the functionality, goals, and constraints of a system, based on the conditions which stakeholders define for the system. The requirements engineering process has an important role in system engineering (Silhavy et al., 2011). Loucopoulos & Karakostas (2001) define requirements engineering as *"the systematic process of developing*

requirements through an iterative cooperative process of analysing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of the understanding gained". This iterative cooperative process is divided into the following three processes, as seen in *Figure 25* (Loucopoulos & Karakostas, 2001):

1. **Elicitation:** concerns the phase of collecting knowledge regarding several aspects of the problem at hand, through different sources. The goal is to produce a formal specification of the necessary prerequisites to solve the target problem. In cases where the problem is novel, the first step consists in understanding it.
2. **Requirement Specification:** the phase where formal specifications of the requirements are defined, as to allow their use for future development stages. These specifications can be viewed as a bridge between users and software developers, often not describing the methods that should be used in order to achieve the target functionality.
3. **Requirement Validation:** considered as an ongoing process, this phase is activated whenever new information is obtained and incorporated. Works to guarantee that incorrect information does not get included in the project's requirements. Usually obtained through meetings with both the development team and the project's client.

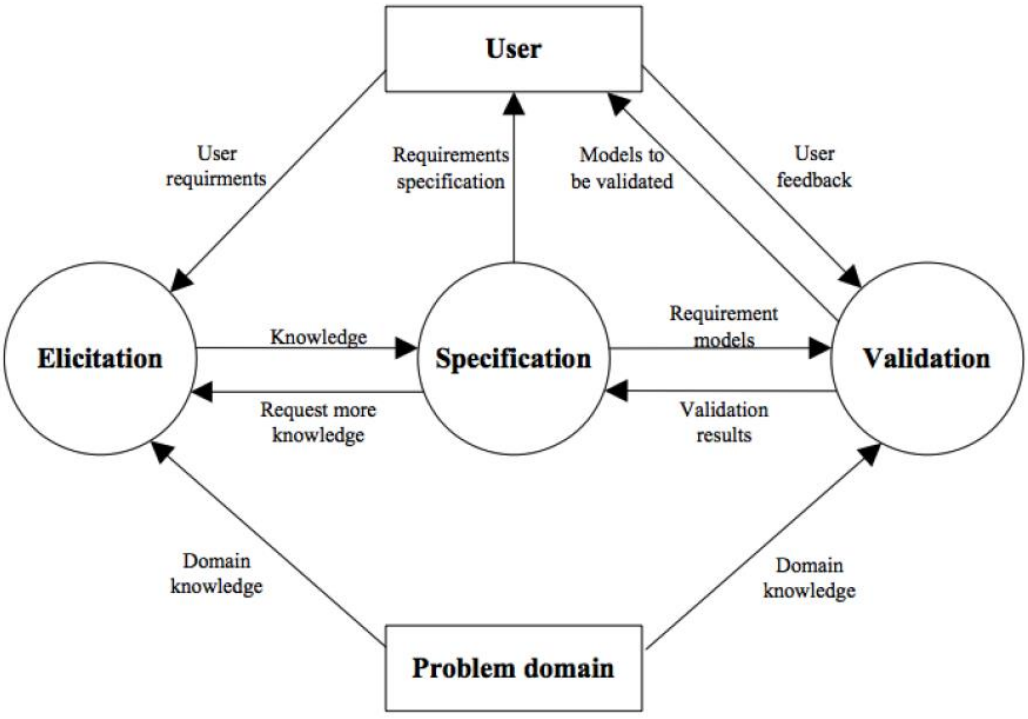

Figure 25 – Sub processes and respective relationships of the requirements engineering (Loucopoulos & Karakostas, 2001).

From the concepts introduced above, the following sub sections will describe each process in the context of this thesis' work.

4.1.1 Elicitation

For the elicitation process, the information was mostly obtained through the conducted state-of-the-art review, as to assess the current needs of AI in beekeeping. Furthermore, meetings with experts, such as National Foundation of Portugal Apicultures (*Fundação Nacional de Apicultores de Portugal,* FNAP), were held as to assess real-life scenario needs and validate findings (FNAP, 2020). The prerequisites can be summarized by the objectives of this thesis and other key findings in chapter 2. These can be briefly summarized as follows:

- The solution must enable remote monitoring.
- The result produced by the solution must have high enough reliability and quality for real usage (starting around 80% accuracy).
- The solution must be able to provide results in a near real-time fashion, as some cases (like *V. velutina* detection) demand fast action for successful minimization of consequences. For these intents, the solution should provide results in less than a second.
- As the solution is expected to serve multiple hives, it should naturally scale with the amount of hives.

4.1.2 Specification

The specification can be used as a tool to capture non-functional requirements (documentation, support, licensing, etc.) and functional requirements that are not captured by use cases. Functional requirements pertain to behaviours a system must be able to perform, whilst non-functional requirements concern the way which a given system will perform a function or behaviour (e.g. performance, reliability, etc.) (Silva, 2015). For this thesis, the requirements are as follows:

- **Functional Requirements:**
    - The system should be capable of processing images of bees and output the corresponding health issue present, if any.
    - The system must be able to detect and automatically crop (through a provided area) images containing bees and/or *V. velutina.*
    - The system should be capable of processing audio and detect health conditions based on the audio, as well as detect whether the audio is coming from the inside or outside of the hive.
- **Non-functional Requirements:**
    - As the system should be applied to real-world scenarios, it must naturally scale without significant degrading of the performance.
    - For the system to be compatible with all scenarios, it should output responses in less than a second, i.e. possess near real-time speed.
    - The accuracy of the overall system should not be lower than 80%.
    - The system must be able to handle images from multiple cameras.

- All results must be accompanied with the respective confidence score.

4.1.3 Validation

For the validation process of the requirement gathering, the requirements stated previously were validated through meetings with an expert in the field – Dr. Ofélia Anjos, teacher at Polytechnic Institute of Castelo Branco and integrator of the International Honey Commission (Anjos & IPCB, 2020). The requirements were also partially validated through opinions and reviews found on beekeeping forums, as far as the needs are concerned (Beesource Beekeeping Forum, 2017).

4.2 Solution Design

The current section has the goal of describing the different design alternatives considered for the development of the classifiers that provide a solution to the thesis' problem. Considering that this thesis' work does not concern the development and implementation of a system for BHM, but rather the development of the classifiers/models, and respective architecture, that enable such implementation, concepts like the Domain Model and Data Modelling will not be described. The reason behind this decision has to do with the fact that the Domain Model and Data Modelling will depend on the context of the developer/engineer's problem and environment. For example, the engineer might see fit to store information in another manner than one that could be suggested in this thesis, due to contextual constraints. As such, only the architectural aspects of the solution were considered and proposed.

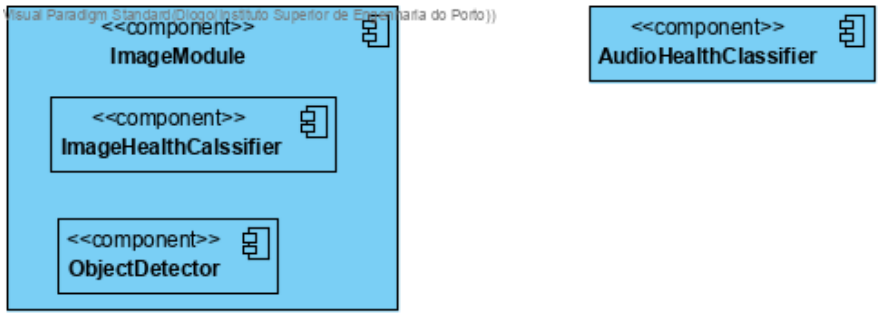

Figure 26 - Component diagram for the classifier alternative design MA.

Concerning the development of the classifiers, two different alternative designs were identified. Regarding the first alternative design – MA – shown in *Figure 26*, it consists of developing only two classifiers: one that deals with image classification and another for audio classification. The image classification would be responsible to do both the object detection and health classification. This can be achieved by treating the image health classification as an object detection task, where the intended object to be detected is the respective image class. This design benefits from the increased speed, as there is less overhead: the image only must be passed once to the classifier in order to obtain the final output. However, this design may

imply sacrifices to the accuracy of the image module, as the number of classes for the object detection task would be significantly higher, which, in turn, can reduce the accuracy (it is easier to distinguish between 2 types of butterflies, than between 10 types).

Regarding the second alternative design – MB – as shown in *Figure 27*, it consists of treating each problem as a separate classifier, totalling 3 classifiers. In this case, object detection, image and audio health classification image would be handled by distinct classifiers. This design has the exact opposite benefits and sacrifices of MA. For the benefits, it has a potentially increased accuracy, as there are fewer classes to distinguish from. For the sacrifices, it has more overhead, as a single image will have to pass through the object detection classifier and the resulting cropped images will have to pass through the image health classification classifier.

Figure 27 - Component diagram for the classifier alternative design MB.

Comparing both alternative designs, as well as the requirements for the master thesis' solutions, it can be stated that design alternative MB is the one that best fits these requirements. As described in the QFD developed in Section 3.4, the quality of the information produced by the solution has a higher priority than the speed of the information. From this priority, design alternative MB was chosen.

4.3 Recommended System Architecture

As stated previously, the goal of this thesis is to provide the classifiers/models for the development of a BHMS, as well as the architecture of these models. The development and implementation of the BHMS is not within this thesis' scope. Despite this fact, it is possible to state that the stakeholders of this master thesis' work would benefit from a recommendation of the architectural design to use for the development and implementation of the BHMS. For this reason, the current section will also describe alternative architectural designs for the potential implementation of a BHMS, as well as provide the recommended best design.

Before design alternatives can be considered, the detection process must be specified, as to assess the potential performance impacts of specific architectures. *Figure 28* describes the recommended detection process, which simply consists of a sequential and ordered execution of the models, considering that the object detection module must be executed before the image health classification. Special note should be taken that with the current design of the models, the object detection and/or image health classification can be executed in parallel to the audio health classification. As such, even though the process will be considered sequential

for easier understanding, it is recommended that a stakeholder who wishes to implement the system parallelizes the classifiers if possible.

Figure 28 - Process diagram and dependency for the recommended detection process.

Figure 29 shows the first alternative for the architectural design – DA. As can be seen, the design consists on having cameras provide images to a component that serves as a scheduler and controller for the entire detection/monitoring process. The controller has the role of orchestrating the detection process by providing the necessary data to each module, in the correct order. The scheduler counterpart has the role of determining which camera is processed next. The following advantages and disadvantages can be identified in design DA:

- **Advantages:**
  - It is easier to develop the management and orchestration of the camera's inputs and the module's outputs (i.e. mapping the detection results to the respective cameras)
  - Less overhead (as the data only needs to be passed to one scheduler)
- **Disadvantages:**
  - It is inefficient in terms of module usage per time unit, since each camera will have to wait for all steps of the detection process to finish (as opposed for each step of the process only having to wait for its specific module)
  - Can only use one scheduling policy, which can be suboptimal for the different types of tasks at hand

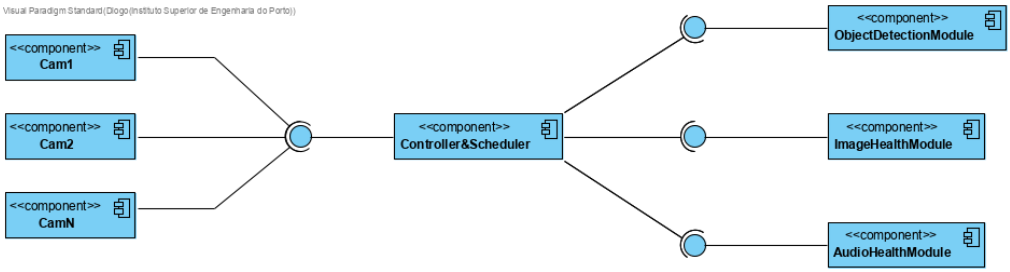

Figure 29 - Component diagram for the architectural design DA of an implementation of the system.

Figure 30 shows the second alternative for the architectural design – DB. As can be seen, the design is like DA, with the difference that each module has its own scheduler. In this design, each scheduler is responsible for scheduling the input for the respective module, according to the policy that is best suited for the type of task. The controller has the role of orchestrating the detection process by providing the necessary data to each scheduler, in the correct order, whilst associating the respective output of each module to the correct camera. The key

advantage of this design is the ability to break down the detection process into its core steps and execute them in parallel, increasing the efficiency of each module. The following advantages and disadvantages can be identified in design DB:

- **Advantages:**
  - Has a higher efficiency per module, as each camera's data only must wait for its respective module to finish, instead of the entire detection process.
  - Can use different scheduling algorithms that can optimally schedule the data for a given module's conditions.
  - Due to the increased efficiency that results from the parallel execution of steps, the architecture has increased speed.
- **Disadvantages:**
  - It is inefficient in terms of module usage per time unit, since each camera will have to wait for all steps of the detection process to finish (as opposed for each step of the process only having to wait for its specific module).
  - Has slightly higher overhead due to the multiple schedulers.
  - The different schedulers may cause additional bottlenecks if their scheduling policies, along with the logic of the controller, generate incompatibilities with each other. An example of this would be the case of priority scheduling and a limited-size buffer for treating cameras in the controller, which can cause the hold of new cameras because a scheduler is constantly choosing cameras with high priority, ignoring the low priority ones. In this case, the users of the system would notice that some cameras would not be getting any feedback from the BHMS.

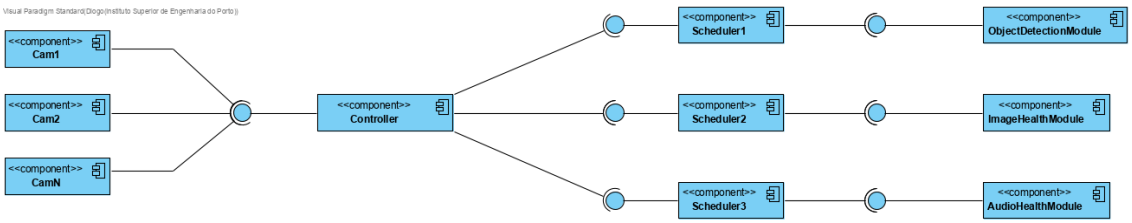

Figure 30 - Component diagram for the architectural design DB of an implementation of the system.

Considering the advantages and disadvantages of designs DA and DB, it can be stated that the recommended architectural design for the BHMS would be DB, as although it has a slightly higher overhead and orchestration complexity, the way that it can split the process in steps allows the system to have a higher usage of each classifier, which in turn should increase the overall speed of the system, when compared to DA.

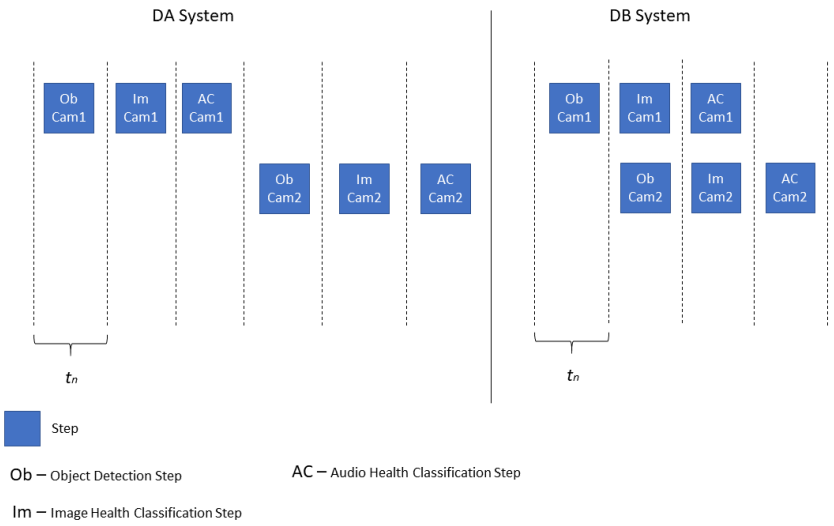

Figure 31 - Comparison of the detection process in DA and DB systems.

This fact should be especially noticeable in the case that each module has the same processing time ($t_{module1}$=$t_{module2}$=t_n). On the context of the recommended detection process, as shown in *Figure 31*, in DA each camera would have to wait $3t_n$ times the number of cameras ahead in the queue, just to be processed. On DB, this wait time would correspond to the worst-case scenario, as each step would be processed in parallel. For example, once the object detection step of camera 1 was finished, camera 2's object detection step could begin. In this case and assuming t_n as the average processing time of each module, the wait time between camera 1 and camera 2 would be of t_n instead of $3t_n$.

4.4 Summary

In this chapter, the design for the proposed solution was described. The requirement gathering was conducted, with both functional and non-functional requirements of this thesis' work listed, as to guide the remainder of the design and development process. Furthermore, design alternatives were shown and described, with a final alternative chosen as the best for the classifier development, with adequate reasoning. Finally, a recommended architectural design, and respective detection process, was described, as to provide help to any stakeholders that may want to implement and develop a BHMS based on the developed solutions of this thesis.

5 Validation & Testing Plan

Programming can be described as problem solving activity. As with any activity like this, determination of the validity of the solution is a vital part of the process. When the severity of the consequences of a program's malfunctioning is high, the validation efforts should be equally high, as to minimize the chance of a malfunctioning occurring (Adrion et al., 1982). For example, software used to control plane landings requires a higher confidence in its performance than the one found in a car-pool locator project. Due to these concerns, validation and testing are needed as to ensure the quality and validity of the solution. In order to do so, validation requirements must be clearly set, as to specify the techniques for validation, verification and testing (Adrion et al., 1982). In sum, software testing can be described as any activity that aims at detecting the differences between existing and required behaviour.

In the case of ML and DL, validation and testing are especially important, as the prevalent applications of techniques from these domains raise natural concerns about their trustworthiness, particularly when applied in safety-critical context, such as medical treatments and self-driving systems (J. M. Zhang et al., 2019). In example, DeepXplore (Pei et al., 2017), which is a differential white-box testing technique for DL, has revealed thousands of incorrect case behaviours in autonomous driving learning systems. Recently, as interest and activity has been rising rapidly, testing has been shown to be an effective way to expose problems and potentially help improving the trustworthiness of ML systems (J. M. Zhang et al., 2019).

This chapter describes the validation and testing techniques used to assess the quality of the developed solutions, in respect to the established objectives and performance requirements. The hypothesis to be tested under this thesis' scope will be described, as well as any relevant techniques, tools and metrics that can provide answers to the hypothesis.

5.1 Hypothesis & Relevant Questions

According to the Oxford Dictionary, the term "hypothesis" can be defined as *"a supposition or proposed explanation made on the basis of limited evidence, as a starting point for further investigation"* (Lexico, 2020). Statistically speaking, a hypothesis can be described as a tentative statement about the relationship between two or more variables, essentially being considered as a specific, testable prediction about the expected behaviour to occur in a study. For a hypothesis to be considered complete, it must specify three components: the variables, the relationship between them and the population (Kabir, 2016).

Hypothesis can help providing answers to questions created by the requirements set for the project. As an example, in the case of this thesis' context, the following set of questions can be extracted from the objectives, requirements, and approach:

1. Do the systems meet the expected accuracy?
2. Compared to other approaches, does the system/solution perform better at detecting bees and *V. velutina*?
3. Compared to other approaches, does the system/solution perform better at detecting a colony's health through audio?
4. Comparing to the baseline, has the proposed system/solution improved the accuracy of the target labels?
5. Does the system meet the expected speed (referenced hereafter as efficiency) performance, under the given computational environment?
6. What architecture type is best at the object detection task, in terms of accuracy and efficiency?

The hypotheses to be tested on this context will help providing answers to the questions above, as to keep focus of the validation and testing of the proposed system. In sum, for each problem, the hypothesis will be focused on the accuracy and efficiency relationship of the developed model, as well as any necessary sub-relationships required. For example, a hypothesis to provide answers to question 4 (described above) is as follows:

- Hypothesis: "Is there a significant difference between the average elapsed time of the baseline system and the one from the proposed solution?"
    - H_0: "The average of the baseline is equal to the average of the proposed solution"
    - H_1: "The average of the baseline is different from the average of the proposed solution"

5.2 Evaluation Methodology

The evaluation methodology can be defined as a tool that aids in the understanding of the steps needed to conduct a quality evaluation through a computational study. It guides readers to learn what must be done and known in order to determine the level of quality of a

performance, product or skill (Baehr, 2004). The evaluation methodology is usually defined in four steps:

1. **Definition of the parameters of the evaluation:** essentially describes the need for the evaluation, how the results will be used and what decisions must be made.
2. **Designing the methods used for the evaluation:** determining how data should be collected, including the decision of the form of evidence (e.g. performance, test questions, etc.).
3. **Settings standards and collecting evidence from a performance or outcome:** consists essentially in defining the scale that describes how the quality of the collected data is judged.
4. **Reporting and make decisions:** the reporting of the results obtained and decision making by the respective authorized decision-makers.

In the context of this thesis' scope, the next subsections will describe the evaluation methodology as per the structured defined above. However, the 4th step does not require a detailed description. The reporting will be performed on this thesis document and the decisions will also be documented on this thesis.

5.2.1 Parameters of the Evaluation

As stated previously, the goal of the testing and validation is to provide a system with trustworthiness. For the system to be trustworthy, there are a set of questions that must be answered. On the context of this thesis, the following set of questions and needs are identified:

1. From the reviewed object detection frameworks, i.e. Mask-RCNN, YOLO and SSD, which provide the best results in terms of detection accuracy and efficiency?
2. Comparing to the baseline approach for the bee health classification through images and through audio, is it possible to improve the results? If so, what is the best hyper-parameter configuration for such results on each problem?
3. Considering the best models obtained from the trials, do they meet the required performance for the proposed system, in terms of accuracy and efficiency?

The above questions are all encompassed by a major main question in the context of this thesis, which is as follows: is it possible to develop a bee health monitoring system, based on detection modules, that has a near real-time performance, with an accuracy starting around 80%?

The results obtained from the testing and validation process will be used to provide answers to the above needs and questions. From these answers, decisions will be made on the type of models to be used for the development of the prototype of the system, as well as any possible impacts on the architectural design of the system and the viability of the system in its application to a business-like environment.

5.2.2 Data Collection

For the objectives of this thesis, three kinds of data sets will be needed for the tasks: one for object detection; another for health classification via image; and a final one for health classification via audio.

Regarding the bee health classification through images, the data set must be composed of cropped images of bees with a variety of health conditions, since using an image with other background objects can introduce errors to the classifier (i.e. the model may think that a bee is healthy because of a background object). Specifically, each image should contain a bee as the focus, as per *Figure 32*. Additionally, since the target object can have low proportions, the image should not have a total resolution lower than 24x24 pixels, as images of bees with lower resolution may not have enough detail for an effective evaluation. Likewise, the image should not have a total resolution higher than 520x520, as it would represent an image quality too high to be realistically obtained for a bee, i.e. a camera with such settings would amass enormous amounts of data to analyse and prove difficult to enable a near real-time health classification, as well as potential footage storage and transmission issues. Apart from the health label, the data should also encompass at least 2 other health-related labels, such as "Presence of *Varroa Destructor*". Finally, each label should have a minimum of 500 images.

Figure 32 - Example of an image containing a bee for the health classification task.

Regarding the object detection task, the data set must be composed of images of bees and/or *V. velutina* in an environment, where these are not the focus of the scene. The images must be manually annotated. Furthermore, as one of the goals of the object detection model is to provide the image health classifier cropped images of detected bees, the dimensions of the annotations should respect the ones defined for the image health classification data set. Annotated bees and wasps must also exhibit a minimum quality, i.e. these should not be blurry to the point that basic features of the bee/wasp cannot be seen (e.g. the stripes of wasps and bees). Furthermore, bees and wasp should only be annotated as a target if at least 50% of its body is visible. *Figure 33* showcases an example of a bee that does not fit annotation conditions due to blurriness. Additionally, the environment images should have as much variety as possible, regarding the perspective, colour, angle, and size of the bee/wasp. The data should also contain images of habitats where bees and wasps may be present but are otherwise not (e.g. hives, flowers, etc.). These images should be used as to refine the object detection model by providing it with empty scenarios that should have bees/wasps (therefore, reducing the impact of the feature that these scenarios may have on the detection

process), composing at least 50 images of the data set. In total, there should be at least 100 images for bees and wasps, respectively.

Figure 33 - Example of an image of a bee that is too blurry. Bees and wasps under these conditions should not be annotated.

Finally, regarding bee health classification through audio, the data set should contain recordings of bee hive colonies. As the goal of the health classification task is to assess any indicators that signal a problem in the hive, the data set should have examples of audio that does not belong to the hive, as to allow the classifier to filter out irrelevant data, corresponding to the label as "not a bee". Furthermore, the data set must belong to at least 3 bee hives, as to reduce bias of a specific colony's behaviour. The data set should have audio segments manually annotated, as to increase the classifier's potential accuracy for the health classification task. There should also exist at least two different health classes, with one of them being a label corresponding to the "healthy" state of the hive/bee. The remaining classes can be relative to any signs that may suggest a problem with the hive such as lack of a queen's buzz, weakened buzz due to fatigue, possibly caused by a disease, and others. For each class ("healthy", "not a bee" and others), the data set must have a minimum of 30 minutes of audio, as to make sure that the captured audio has chances of containing bees that are returning from foraging. Bees can travel around 15km/h and will fly up to 3.2 km to find nectar and pollen (Canadian Honey Council, 2020; University of Kentucky, 2020). From these values, it can be stated that the worst average time for a bee's travel is of 12 minutes and 36 seconds, resulting in a worst average amount of travels of 2-3 trips per hour. As one trip is equal to two travels, it can be said that one trip can have the worst average time of around 25 minutes. Therefore, as to increase the odds that at least one trip is caught, the audio should contain a total of 30 minutes.

5.2.3 Metrics & Tests

The metrics to be used for the quality assessment of the proposed solutions will have slight variations in each task. However, for all developed models, whenever possible, the elapsed time will be used for evaluation of the efficiency of the model.

For the case of the bee health classification through images, the baseline's metrics shall be used, as not only these are adequate for the classification problem, but are also useful to perform a direct comparison between the approach defined in the baseline and the one

defined on this thesis. Specifically, the accuracy, f1 score, recall, precision, and validation loss will be used.

Regarding the bee and *V. velutina* detection, accuracy and mean Average Precision (mAP) will be used. As the task deals with object detection, all values to compute the previous metrics (e.g. true positives, true negatives) will be obtained using the IoU and a classification threshold of 0.5, i.e. all detections whose IoU is higher or equal to the threshold value are counted as a true positive.

Finally, concerning the bee health classification through audio, accuracy, precision, recall and f1 score will be used to evaluate the model. *Table 5* summarizes all metrics described above.

Table 5 - Metrics to be used per task.

Task	Metrics to be Used
Bee Health Classification (Image)	Elapsed time, accuracy, validation loss, recall, precision, f1 score
Bee Health Classification (Audio)	Elapsed time, accuracy
Bee & Wasp Detection	Elapsed time, accuracy, mAP

In the case of testing, all three major tasks will be evaluated using t-test, as their hypotheses suggest great potential in the application of this type of testing. This is because in the hypothesis, a statistical assessment of the differences in the relationship of the variables described is desired. As such, parametric tests like t-tests will be conducted on results obtained from these models, in respect to the defined metrics. However, for the bee and wasp detection task, ANOVA testing will also be conducted, as to compare differences between the three intended models to be used. These tests will only be applied if their applicability conditions are verified, i.e. if the pre-conditions necessary to execute the tests with reliable results are met. If the conditions are not met (e.g. normality assumption), non-parametric tests will be used, whilst also verifying their respective assumptions. *Table 6* provides a summary of the tests to be conducted per task. All classifiers will also be validated using the holdout cross-validation (train/validation split).

Table 6 - Tests that may be conducted per task. The execution of the tests will depend on their applicability.

Task	Tests
Bee Health Classification (Image)	t-test, Shapiro-Wilk, Wilcoxon, Mann-Whitney U
Bee Health Classification (Audio)	t-test, Shapiro-Wilk, Wilcoxon, Mann-Whitney U
Bee & Wasp Detection	t-test, ANOVA, Kruskal, Shapiro-Wilk, Wilcoxon, Mann-Whitney U

The following sections aim at providing a context of the metrics and tests used, as to establish a common ground of knowledge and definitions for the respective concepts.

5.2.4 Metrics Definitions

The current section aims at providing insights of the meanings behind the metrics used for the validation and testing phase of this thesis' work. However, before the proper definition of the metrics, it is important to define what true positives/negatives and false positives/negatives are. True positives relate to the cases where the intended result was expected to be positive and the result obtained from a given model/algorithm is also positive (therefore, a **true** positive). The same logic applies to true negatives, with the difference of the expected and obtained results being negative (Google Developers, 2020b). False positives occur when the expected outcome is a negative result, but the result obtained from a model/algorithm is positive (therefore, a **false** positive). Likewise, the same logic applies to the false negatives, but with the expected outcome being a positive result and the obtained result as a negative. To demonstrate, the following case can be considered: the development of a predictive model to assess whether a patient has signs of cancer. In this case, a positive and negative results would correspond to the facts "patient has signs of cancer" and "patient does not have signs of cancer", respectively. In the case of the true positive, these would correspond to a test where the expected outcome is "patient has signs of cancer" and the model correctly outputted "patient has signs of cancer". However, an example of a false negative would be the case where the expected outcome is "patient has signs of cancer" and the model wrongly outputted "patient does not have signs of cancer" (Google Developers, 2020b). *Table 7* provides a brief summary of true/false positives/negatives examples. This example will be used on the following subsections a demonstration is required.

Table 7 - Examples of true/false positive/negative cases, as well as their relationship between the model and the expected outcome.

	Expected Outcome Positive	**Expected Outcome Negative**
Model Output Positive	True Positive • The patient has signs of cancer and the model also states this	False Positive • The patient does not have signs of cancer, but the model stated otherwise
Model Output Negative	False Negative • The patient has signs of cancer, but the model stated otherwise	True Negative • The patient does not have signs of cancer and the model also states this

5.2.4.1 Accuracy

Accuracy can be defined as the number of tests where the output was correct over the total number of test cases (Caruana, 2003; Louis, 2017). Formally, accuracy can be defined as the

sum of the total number of true positives and true negatives, over the total number of cases, as per below:

$$Accuracy = \frac{n_{truePositives} + n_{trueNegatives}}{n_{totalCases}} \quad (7)$$

Although accuracy is the simplest and most widely used metric to measure the performance of a classifier, it is not a proper metric that can always be applied to all types of data set, as it is insensitive to bias (Louis, 2017), since accuracy assumes the same cost for both positive and negative classes of a task (Caruana, 2003). Using the previous example, the following scenario can be described: the goal is to assess a classifier's performance in correctly diagnosing signs of cancer in 100 patients, and the classifier correctly outputted the response of 95 candidates. From the previous information, it could be concluded that the accuracy of the classifier is 95%, which, in turn, can be described as a high accuracy value. However, assuming the following evolution of the previous scenario: all 95 correctly outputted cases correspond to cases where patients did not have signs of cancer. In cases where patients did have signs of cancer (the remaining 5), the classifier failed. Having this scenario into context, it is possible to understand how accuracy can fail in biased data sets – on the example, accuracy cannot distinguish between a classifier built with good practices and intentions, and a simple algorithm that always outputs "false" to whatever input is given. Furthermore, on the context of the previous example, it can be stated that the cost of false negatives is quite higher than the cost of false positives. This is because if the classifier outputs a false positive, further testing can be conducted in order to fully assess whether the patient has cancer or not. However, if the classifier outputs a false negative, the patient's cancer will not be treated early on and can cause great impact on their lives and treatment success. Nevertheless, in cases where data is correctly balanced and the costs of the classes are approximately equal, accuracy can still be used to obtain reliable results.

5.2.4.2 Recall & Precision

On the context of ML (excluding the are encompassed by Natural Language Processing, where the definitions differ slightly), precision can be defined as the metric that provides the proportion of positive identifications that were correct, whilst recall provides the proportion of actual positives that were identified correctly (Google Developers, 2020a; Louis, 2017; Shung, 2020). Formally, precision and recall can be defined as per below:

$$Precision = \frac{n_{truePositives}}{n_{truePositives} + n_{falsePositives}} \quad (8)$$

$$Recall = \frac{n_{truePositives}}{n_{truePositives} + n_{falseNegatives}} \quad (9)$$

Although the definition of both precision and recall may seem similar, they aim to test different things. In the case of precision, it aims at testing how many true positives were obtained from the classifier, from all positives outputted by it. So, on the previous example, precision aims to know, based on the classifier's outputs, the percentage of patients that

actually had signs of cancer, over the total cases that the classifier said that the patient had signs of cancer. In this example, if the precision value were to be 0.5, it could be stated that the classifier is correct 50% of the time when it predicts that a patient has signs of cancer (Google Developers, 2020a).

In the case of recall, it aims at testing how many true positives were obtained from the classifier, out of all cases that should have been positive. From the running example, recall would give the percentage of cases where the classifier stated the patient had signs of cancer correctly, over the total number of patients that have signs of cancer. If the classifier were to have a recall value of 0.22, it could be stated that the classifier correctly identifies 22% of all cases where patients have signs of cancer (Google Developers, 2020a).

In order to properly evaluate a model, the combined use of recall and precision is required, as these metrics alone are not enough to fully evaluate a model's effectiveness. Furthermore, recall and precision have an indirectly proportional relationship, i.e. improving one metric reduces the other metric (Google Developers, 2020a).

5.2.4.3 Dice/F1 Score

The Dice score (also known as F1 score) can be defined as a harmonic mean of the precision and recall metrics. This metric aims at combining both precision and recall metrics in a balanced manner, as to serve as an overall measure of a model's performance (Nicholson, 2019; Powers, 2020; Sasaki, 2007; Shung, 2020). Formally, the F1 score is defined as per below:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

Although F1 score might be better at evaluating a model when the goal is to seek a balance between Precision and Recall, the work done by Powers (2020) highlights some pitfalls associated to the metric, such as the fact that the metric is biased towards the majority class.

5.2.4.4 Validation Loss

On the context of ML, logarithmic loss (usually referred to as log loss) can be defined as a function that serves the purpose of evaluating how good are the predicted probabilities of a given classifier. Validation loss is the loss obtained for the validation set when training a given classifier (Carvia Tech, 2019; Godoy, 2019; T. Srivastava, 2019). Multiple functions can be used to compute the log loss, such as binary cross entropy. Log loss can be viewed as a general evaluation metric of the classifier's predicted probability, where the lower the log loss value, the higher or more confident the classifier is in its predicted result. However, the use of log loss as an evaluation metric is case dependent (T. Srivastava, 2019). Regardless, using the validation loss, it is possible to assess whether the model is overfitting or not, as shown in *Table 8*. This assessment will also be conducted on the developed models whenever possible, to further provide insights on their performance.

Table 8 - Overfitting behaviour when analysing validation loss and accuracy (Carvia Tech, 2019).

Validation Loss	Accuracy	Possible Outcome
increasing	Decreasing	no learning
increasing	Increasing	overfitting or diverse probability values (e.g. SoftMax being used as output layer)
decreasing	Increasing	normal and expected behaviour

5.2.4.5 Intersection over Union

In CV, intersection over union measures the overlap between 2 boundaries, on an interval of [0,1]. It is used to assess how much of the predicted boundary of an object detection model overlaps the ground truth provided (i.e. the real object boundary). IoU is defined as the area that results from the intersection of the two boundaries, over the area that results from the union of the two boundaries, as per below:

$$IoU = \frac{GroundTruth \cap Prediction}{GroundTruth \cup Prediction} \quad (11)$$

This value can be used with a threshold value in order to estimate true/false positives. Usually, the value of 0.5 is used for the threshold as it is a value that allows the classifier to obtain reasonably good boundaries without being too strict, as shown in *Figure 34*.

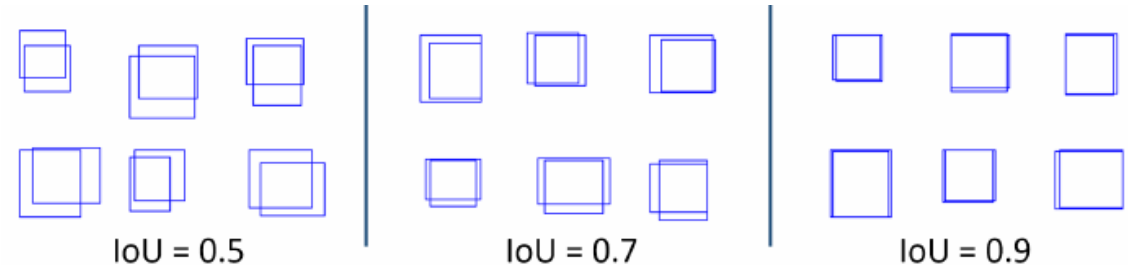

Figure 34 - Comparison of different IoU thresholds for object detection (Santos, 2020).

5.2.4.6 Mean Average Precision

In CV, Average Precision (AP) can be defined as the area under the precision-recall curve, which corresponds to a curve where precision (y axis) is plotted dependent on recall (x axis), as shown in *Figure 35*. The precision and recall values are computed based on the IoU values and a given threshold. The mAP metric is the average of the several AP values obtained for each class. However, there are some scenarios (as is the case of the COCO data set) where AP and mAP are used as synonyms (Hui, 2019).

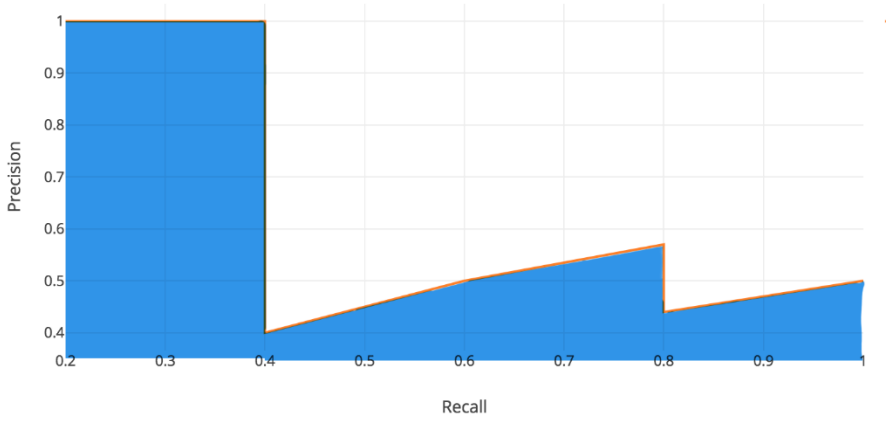

Figure 35 - Plot of the precision-recall curve. The area under the precision-recall curve is highlighted in blue. Adapted from (Hui, 2019).

5.2.5 Tests Definitions

The current section aims at providing context of the tests to be used in the testing and validation phase. Specifically, the several possible tests that may be used will be described, as to provide context of their use and assumptions in this thesis.

5.2.5.1 t-test

t-test can be defined as a parametric hypothesis test which compares the averages of two samples, with the goal of assessing if there is a significant statistical difference between the averages, at a given confidence level. t-test can also be used with only one sample, in which case the sample's average will be compared with a target average value, as to verify a hypothesis (T. K. Kim, 2015). As t-test is parametric, in order to use it, the data must follow a normal distribution.

5.2.5.2 ANOVA test

The Analysis of Variance (ANOVA) test is a parametric hypothesis test which evaluates two or more groups' averages in order to assess whether these are different by chance variation or by a difference in their underlying model (Sawyer, 2009). The ANOVA test can provide answers to the hypothesis of "Is there a significant difference between the averages of *n* groups?". In ANOVA, the null hypothesis is considered as the hypothesis where all averages are statistically equally, and the rejection of the null hypothesis (i.e. accepting the alternative hypothesis) means that there is at least one average that differs (not necessarily all of them). For the ANOVA test to be applicable, the following assumptions must be validated (Sawyer, 2009):

1. The variance of each group being tested must be the same, i.e. all groups must have equal variance between each other.
2. The error for any group must follow a normal distribution (usually addressed by the central limit theorem).

3. The observations of each group must be independent.

5.2.5.3 Normality Assessment – Shapiro-Wilk Test, Kurtosis and Skewness

In statistical testing, the assessment of the data's normality is a common prerequisite, as it is an underlying assumption for all parametric tests. This assessment can be performed graphically (through the use of QQ Plots and histograms, for example), or quantitatively.

The Shapiro-Wilk test can be used to assess if a given sample comes from a normal distribution. Although the test is more suitable for small sample sizes (i.e. lower than 50 samples), the Shapiro-Wilk test can also handle sample sizes up to 2000 (Laerd Statistics, 2018d; Stephanie, 2014). The test provides a W value which can be used to assess normality. If it is a small value, it can be stated that the sample does not follow a normal distribution.

In regard to kurtosis and skewness, it can be stated that these two metrics can aid in determining if a given data's distribution is normal (S. Brown, 2016; McNeese, 2016; SmartPLS, 2020). In the case of skewness, it relates to the distribution's symmetry, where values close to 0 indicate a normal distribution. Significantly high positive values (around 1 or higher) state that the symmetry is right skewed. Likewise, significantly high negative values (around -1 or lower) state the symmetry is left skewed. *Figure 36* showcases an example of left and right skewed distributions. In the case of kurtosis, it corresponds to a measure of whether the distribution is too peaked or too flat. A kurtosis of less than -1 suggests the data's distribution is too flat, whilst if the value is higher than 1, the data's distribution is substantially peaked (S. Brown, 2016; McNeese, 2016; SmartPLS, 2020).

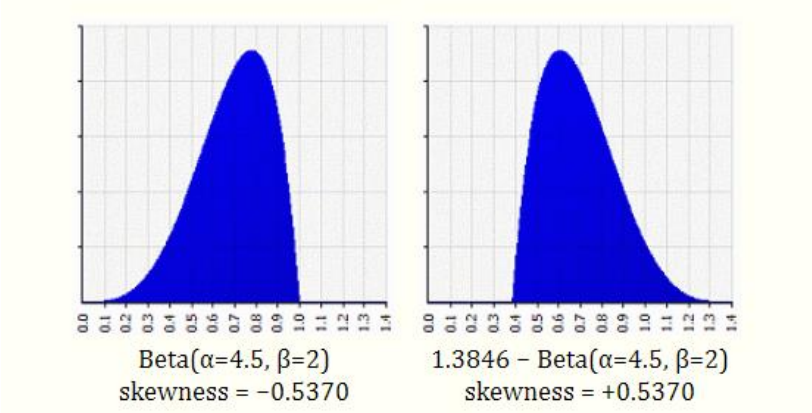

Figure 36 - Skewness example. On the left, a left skewed distribution. On the right, a right skewed distribution (S. Brown, 2016).

5.2.5.4 Kruskal-Wallis Test

The Kruskal-Wallis test can be defined as the non-parametric alternative to the One Way ANOVA (being sometimes called the "one-way ANOVA on ranks") (Laerd Statistics, 2018a; Stephanie, 2016). As it is a non-parametric test, the Kruskal-Wallis test does not assume that the data comes from a distribution, suitable for cases where data has a non-normal distribution. The test uses the ranks of data values instead of the actual data points and determines whether the medians of two or more groups are different (Laerd Statistics, 2018a;

Stephanie, 2016). Like most statistical tests, Kruskal-Wallis uses the H test statistic, which is based on the definition of the following hypothesis:

- H_0 – population medians are equal.
- H_1 – population medians are not equal.

Furthermore, like ANOVA, the Kruskal-Wallis test is an omnibus test, which means that the test can only tell if there is a difference in the tested groups and cannot tell which group is different.

To use the Kruskal-Wallis test, the following assumptions must be previously verified (Laerd Statistics, 2018a; Stephanie, 2016):

1. The dependent variable must be measured in ordinal, interval, or ratio scale (Laerd Statistics, 2018e).
2. The independent variable should have two or more categorical, independent groups (two or more levels). Typically, the Kruskal-Wallis H test is used for three or more independent groups, with the Mann-Whitney U test being more commonly used for two independent groups.
3. The observations should be independent, meaning that there should be no relationship between members of each group. If this assumption fails, the Friedman test is recommended.

The shape distribution of each group must be known to properly interpret the results. If these are the same, the Kruskal-Wallis H test can be used to compare the medians of the dependent variables. However, if the shape is different, the test can only be used to compare mean ranks.

5.2.5.5 Wilcoxon Test

The Wilcoxon signed-rank test (also called the Wilcoxon signed-rank sum test) is considered as the non-parametric test equivalent to the t-test, which assumes that the population data does not follow a normal distribution (Laerd Statistics, 2018f; Stephanie, 2015). Due to this reason, it can be used when the normality assumption is violated, rendering the t-test inappropriate. It can be stated that there are two slightly different versions of the test:

- **Wilcoxon signed-rank test**, which compares a sample's median against a hypothetical median.
- **Wilcoxon matched-pairs signed-rank test**, which computes the difference between each set of matched pairs, and then performs the standard procedure as the signed-rank test. The specifics of this test will be further defined below.

Despite the above, the term "Wilcoxon" is used interchangeably for either test. Furthermore, the data samples should be paired (i.e. the two sets of scores must come from the same participants). To conduct the Wilcoxon test, the following assumptions must be verified (Laerd Statistics, 2018f; Stephanie, 2015):

1. The dependent variable must be measured in ordinal, interval, or ratio scale.

2. The independent variable must have two categorical, paired groups (i.e. non-independent).
3. The distribution of the differences between the two paired groups needs to be symmetrical in shape. If this assumption is not verified, the data can be transformed to achieve a symmetrically-shaped distribution (although not a preferred option, as it can change the meaning of the test's results) or a sign test can be run instead of the Wilcoxon test.

Finally, the Wilcoxon test allows one to specify whether the test is two-sided, greater or less, which means that the alternative hypothesis states that the scores of the groups are different, the first groups' scores are greater than the second's, and the first groups' scores are lower than the second's, respectively.

5.2.5.6 Sign Test (*Wilcoxon matched-pairs signed-rank test*)

The paired-samples sign test, commonly referred to as sign test, can be used to assess whether there is a median difference between paired observations (Laerd Statistics, 2018c). It is considered as an alternative to t-test and the Wilcoxon signed-rank test, when the data normality or symmetry assumptions are not verified, respectively. To apply the sign test, the following assumptions must be validated (Laerd Statistics, 2018c):

1. The dependent variable must be measure in ordinal, interval, or ratio scale.
2. The independent variable must have two categorical, paired groups (i.e. non-independent).
3. The paired observations need to be independent (i.e. one participant's values cannot influence another's).
4. The difference scores are from a continuous distribution. If this assumption is not met, then there is a risk that a unique median does not exist.

5.2.5.7 Mann-Whitney U

The Mann-Whitney U test can be used to assess whether there is a median difference or mean rank difference between two independent groups of observations, if these do not follow a normal distribution (Laerd Statistics, 2018b). Although not always the case, due to the different possible interpretations of the Mann-Whitney U results depending on the data's shape, the test is often considered the non-parametric alternative to the t-test. To apply the Mann-Whitney U test, the following assumptions must be validated (Laerd Statistics, 2018b):

1. The dependent variable must be measure in ordinal, interval, or ratio scale.
2. The independent variable must have two categorical, independent groups.
3. The observations must be independent (i.e. one participant's values cannot influence another's, and both groups must also be independent).
4. The two groups' data must not follow a normal distribution. Furthermore, if these have the same shape, then the test refers to the median of the groups. Else wise, the test refers to the mean ranks.

5.2.5.8 Holdout Cross-Validation

The holdout cross-validation method can be defined as a simple validation method where the training and testing data are separated into two sets: the training and testing sets. The split can is usually performed considering a certain percentage (e.g. 66% of the data is training data and the remaining 33% is validation data) and it can be split in accordance to class distribution, i.e. making sure the class is equally distributed in both the training and validation sets, percentage wise (Schneider, 1997). The main advantage of this method is its reduced computational time due to its simplicity, which is desirable in the case of CV & DL classifiers (A. Ramezan et al., 2019).

5.3 Summary

In this chapter, the validation and testing techniques used to assess the quality of the developed solutions were described. The potential hypothesis and relevant questions were specified, as well as the requirements for the testing plan, with a detailed explanation of the quality of the data that must be obtained. Finally, the tools and metrics that will be used were described.

6 Development & Computational Study

In this chapter, the development process of the solution to this thesis' problem is described. Firstly, general considerations about some of the development choices taken are provided, with the goal of further detailing the development process. As this thesis' problem can be divided into 3 distinct tasks (i.e. health classification through image, object detection of bees and Asian wasps, and health classification through audio), the solution provided by this thesis is composed by 3 sub solutions. Each solution will be described in a depth-first fashion, including the model's development process and the comparison of the different models and architectures used, using statistical tests whenever adequate. Furthermore, the different approaches defined for a given task, and respective relevant pre-processing steps, evaluation algorithms and used data sets will be described.

6.1 General Considerations

During the development process of the models for each problem previously identified, some considerations were had in regard to the assessment of performance metrics. The goal of these considerations consists in providing a robust comparison between different architectures and context problems, so it is possible to assess the impact in using each of the provided solutions in the implementation of a system. Essentially, these considerations focused around two key points: measurement of the efficiency and quality of the solutions.

Regarding the efficiency measurement, all solutions' metrics were obtained using the same algorithm, as to provide a way to compare each solution, if desirable. The algorithm consists in measuring the elapsed time (in seconds) of the processing of a single input object (i.e. an image or an audio sample). Specifically, the time measurement does not consider neither the setup time of the model/solution nor the time spent obtaining the input object (e.g. the amount of time it takes to load a frame from an image source). Instead, it only considers the elapsed time of the call of the model's "detect" function, as can be seen in the *Code Snippet 1*. The reason behind this decision is that assessing only the "detect" function's elapsed time grants the best possible performance that a model can obtain, enabling a fair comparison of

each model when considering the best choice for a real-life scenario implementation. Furthermore, all time measurements used a sample of 1000 input objects, as to guarantee a large enough sample for proper testing and comparison. During time data collection, all possible processes on the system were terminated and network connection was disabled, as to minimize the possibility of residual errors in the measurements conducted due to the computational environment handling other processes in parallel. Furthermore, the data collection regards the elapsed time of a given model obtained using a subset of the data set used in the corresponding task. The same subset is used for all time measurements of the same task's models. Therefore, it can be stated that the data sample collected for the models follows a continuous distribution (as it corresponds to the elapsed time in milliseconds, up to the microseconds). Furthermore, the elapsed time data will be considered as independent, since although the subset is the same, the models (i.e. the groups) that generated the data sample of elapsed times are different and have no correlation with each other (different weights, often different architecture, among others). From this assumption, it can also be stated that any tests which require paired groups (such as the Wilcoxon test) will not be used nor considered for the elapsed time comparison. Instead, tests such as Mann-Whitney U will be considered.

```
1.  Def elapsed_time_solution(model,loaded_data):
2.     # Any setup that occurs
3.     ...
4.     start_time = 0
5.     end_time = 0
6.     for input_object in loaded_data:
7.        # Measure time only for detect function
8.        start_time=time.time()
9.        model.detect(input_object)
10.       end_time = time.time()
11.       measurements.append(end_time-start_time)
12.    ...
```

Code Snippet 1 - Time measurement algorithm.

Like the efficiency measurement, the assessment of the quality of object detection models also used the same algorithm, only with modifications to the way confusion matrix values were obtained, which were model data dependent (i.e. binary or bounding-box masks). On the topic of object detection metrics, it can be stated that although there are some metrics that are more popular than others (e.g. mAP), some differences still exist on the definitions and implementations of the these metrics, including ones used in major competitions like PASCAL VOC and MS COCO (Bruintjes, 2019; Manal El Aidouni, 2019; Mao et al., 2019; Zeng, 2018). As such, in order to provide a comparison between models, a custom approach was defined to assess the quality of any model, so that there is a common base for comparison. Specifically, the following approach was defined: implementation of a custom algorithm to compute the confusion matrix of a given model, for a given data set, where the only difference in each model is in regard to the way results are obtained (i.e. the way a 'hit' or 'miss' is assessed). In the case of the confusion matrix, it allows to obtain metrics such as accuracy, recall and precision, which can be useful to assess a model's performance using common, non-specific ML metrics.

On the topic of assessing a "hit" or "miss" of a given example, further considerations and decisions were made for the implementation of the custom algorithm. In object detection, the output of a given model is considered correct if the area of the predicted mask overlaps a desirable amount of the truth mask. As stated previously, the IoU provides the intersection of the masks (between 0 and 1), and the threshold commonly used is 0.5. However, the complex part of the evaluation algorithm resides in choosing the predicted mask that should be compared with the truth mask, as well as the expected behaviour in the case of multiple overlapping predicted masks. On the context of choosing the best candidate predicted mask, the custom algorithm chooses the predicted mask closest to the truth mask, using the Euclidean distance. The position used for the distance computation is obtained from the calculation of the centre of the mask (i.e. if the masks are bounding boxes, the centre is computed using the corner coordinates; if the masks are Boolean values, the centre is computed considering that it is a centroid). On the context of the expected behaviour in the case of multiple overlapping predicted masks, the following two alternatives were identified:

- **Consider multiple predicted masks as TP if their IoU is above the desirable threshold**, which means the model predicted "correctly" when the intended detected object was in the given area, but did this prediction multiple times (possibly meaning that the model fails to remember a given prediction's location).
- **Consider only one predicted mask as a TP, with the rest being FP**, which translates to considering that the model incorrectly predicted the extra masks and assumed that other objects/information close to the TP were the incorrectly target object (e.g. considering that a bee's leg is a bee).

Comparing both alternatives, the first alternative was concluded to provide a more optimistic assessment of the model's quality performance, albeit possibly less realistic. Likewise, the second alternative was considered to provide a less optimistic assessment, but more realistic. Taking into consideration the differences between the approaches, the potential impacts (both positive and negative) in obtained results, and the overall goal of this thesis, the second alternative was implemented in the custom algorithm. The reasoning behind this decision is that if an algorithm is good enough to meet real-life implementation standards, it will have good results under the harshest conditions, such as a pessimistic assessment of the model's quality performance. As such, as can be seen in *Code Snippet 2*, for each target class, the custom algorithm compares the predicted mask closest to the truth mask and if the IoU is higher than the threshold, the mask is considered a TP. This process is repeated until there are no remaining truth masks. Any remaining predicted masks are automatically labelled as a FP. Additionally, apart from the case where the IoU is lower than the threshold, the algorithm also considers that a FN occurred whenever the classifier provides any detection for a given class *X* and the current image being analysed does not contain any examples of class *X*. Whenever such occurs, the number of detections that were incorrectly labelled as class *X* will be considered FN. Furthermore, in the case that the image contains less objects of a given class (e.g. class *Y*) than the number of predicted masks, all remaining masks for that given class will also be labelled as a FN (e.g. if the image only has 2 bees, but there are 3 predicted masks of bees, at least 1 of these must correspond to a FN).

```
def compute_confusion_matrix_image_data(self,truth_class_array,truth_mask_array,pred_class_array,pred_mask_array):
    ground_truths = self.pair_class_names_masks(truth_class_array,truth_mask_array)
    predictions = self.pair_class_names_masks(pred_class_array,pred_mask_array)

    unique_classes = unique_list(truth_class_array)
    for clazz in unique_classes:
        has_gt_masks = False
        if clazz not in ground_truths:
            if clazz in predictions: #If mask does not exist in ground truths, but predictions has it, then it's a false positive
                self.matrix.add_value(clazz, 'fp', float(len(predictions[clazz])))
            else:
                #If not, then it's a true negative
                self.matrix.add_value(clazz,'tn',1.0)
        else:
            masks = ground_truths[clazz]
            if clazz not in predictions:
                #If predictions do not have any class, then it's a false negative
                self.matrix.add_value(clazz, 'fn', float(len(masks)))
            else:
                pred_masks = predictions[clazz]
                for i in range(0,len(masks)):
                    has_gt_masks = True
                    # if predictions has no targets for a class but there are still ground truths, then it's a FN
                    if len(pred_masks)==0:
                        #False Negative
                        self.matrix.add_value(clazz,'fn',float(len(masks)-i-1))
                    else:
                        index = self.closest_prediction_to_truth(masks[i],pred_masks)
                        pred_mask = pred_masks.pop(index)
                        iou = self.compute_intersection_over_union(masks[i],pred_mask)
                        if iou>self.iou_threshold:
                            #True Positive
                            self.matrix.add_value(clazz,'tp',1.0)
                        else:
                            #False Negative
                            self.matrix.add_value(clazz,'fn',1.0)
                #if predictions still have masks (extra), but ground-truths have not: FP
                if has_gt_masks and len(pred_masks)>0:
                    # False Positive
                    self.matrix.add_value(clazz,'fp',float(len(pred_masks))) # pred_masks should have the remaining masks
```

Code Snippet 2 - Custom algorithm for confusion matrix calculation of object detection models.

Additionally, on the topic of the impact (weight) of FP and FN when developing a classifier, it can be stated that different weights can be applied to the importance of FP and FN. For example, in the case of detecting if a patient has cancer, it is usually considered that a FN has a lot more impact than a FP, as it would imply failing to diagnose someone who needs treatment. As such, in cases like this, the minimization of FN is prioritized over the FP. However, on the context of this thesis' work, from feedback obtained by experts in the field and by companies, FP and FN are assumed to have the same impact. Considering the detection of bees and Asian wasps, it was argued that, although the algorithm should have a higher accuracy detecting Asian wasps (as these are a high threat for the hive, making their FN have higher weight than FP), it is also equally important not to miss-detect a bee or another object for an Asian wasp, as it would require manual check-up every time such an event occurred in a significant frequency (therefore, diminishing the added value of the solution to a company, for example). This applies to the health conditions of honey bees as well, as any issues with the hive will require manual inspection by a beekeeper. As both pros and cons of FP and FN outweigh each other in the defined tasks, FP and FN will be considered to be equally important.

Finally, in regard to the physical and virtual setup, all models were ran on the computational environment specified in *Table 9*, using the SciPy Python package, which is an open-source software for mathematics, science, and engineering, widely used in the Python language (SciPy, 2020). Furthermore, in all statistical tests used to assess differences in results obtained from the developed models, the confidence degree used was 95% (significance level α=0.05), as per literature review conducted (Siegle, 2015).

Table 9 - Computational environment details, discriminated by physical setup and virtual setup.

Graphics Card	GeForce RTX 2070 8GB
CPU	Intel i7 Hexacore 8700 3.20GHz
RAM	32GB
LANGUAGE	Python
ENVIRONMENT	Keras with Tensorflow GPU backend; tensorflow v1.13.1; keras v2.2.4; mask-rcnn v2.1; cuda v10.0.130; cudnn v7.6.0;

6.2 Bee Image Health Classification

As stated previously, the goal of the bee image health classification task is to correctly identify health problems in images of cropped bees. The developed model should be able to detect different health problems, as well as if a bee is healthy, so that the information can be used by a monitoring system to inform beekeepers and other stakeholders of any issues that may be occurring in the bee hive.

6.2.1 Data Description

The data used for this task is from Yang (2018), which corresponds to a data set of cropped honey bee images, each annotated with several attributes, publicly available on Kaggle (J. Yang, 2018b). The health, subspecies and pollen carrying (*pollen_carrying*) attributes are particularly relevant to this problem's context, as they correspond, respectively, to the current state of the bee hive's health, the subspecies of the honey bee and whether it is carrying pollen or not. Apart from the health attribute, the subspecies and pollen carrying attributes were selected for the solution, as they can be used to further determine health problems in a bee hive. For example, the subspecies attribute can help beekeepers know if and when a bee hive is being robbed (by having the system identifying different subspecies in the same hive). In the case of the pollen carrying attribute, the number of bees detected to have this characteristic in a given time span can help beekeeping experts assess if the colony may be suffering from poor nutrition or scarcity of food. Due to these examples, it can be stated that these attributes provide potential insights into useful information. Regarding the health attribute, it represents the current state of the bee hive, as annotated by experts of the field, when the image of the honey bee was taken. The reasoning behind this annotation is centred around the fact that the state of bee hives can reflect changes in honey bees and vice-versa (Meikle & Holst, 2015). *Table 10* highlights the different values that each selected attribute can have.

Table 10 - Possible values of each attribute of the bee image health classification data set. As can be seen, some of the attributes (like subspecies) account for unknown examples.

Attribute	Values
Health	• Varroa, Small Hive Beetles • Ant Problems • Few Varroa, Hive Beetles • Healthy • Hive Being Robbed • Missing Queen
Subspecies	• 1 Mixed Local Stock 2 • Carniolan Honey Bee • Italian Honey Bee • Russian Honey Bee • VSH Italian Honey Bee • Western Honey Bee • Unknown
Pollen Carrying	• True • False

The data set is composed of 5,172 RGB[2] images of honey bees, with varying sizes (width between 27 and 392 pixels; height between 24 and 520 pixels), annotated by experts in the

[2] Red, Green & Blue (RGB) is a colour system to represent images using Red, Green and Blue channels.

field. The work done by Pukhov (2018) was used as the baseline for this solution. In the Kaggle kernel, the author implemented a basic pipeline to load the data set in an efficient manner and to train classifiers using a balanced data set. When any data set is loaded, the pipeline checks how many instances belong to each class and proceeds to balance the data by splitting in accordance to class distribution, forcing all classes to have the same distribution, minimizing bias. Additionally, the pipeline also implements an image data generator (i.e., the standard Keras *ImageDataGenerator*) which randomizes image attributes by applying operations such as scaling, skewing, translation, flipping and zooming, among others. The baseline work's learning algorithm is DCNN. In the baseline work, the author also created methods to visualize the kernel matrices produced by the developed DCNN. Furthermore, classifiers for the health, subspecies, and pollen carrying attributes were also developed by the author (among other classifiers), which will be used for comparison of the results obtained. All images were scaled to have the same size (i.e., 100x100 pixels).

From the gathered data set, it can be stated that the specific goal of the problem is to classify cropped images of bees according to their health, to their subspecies and to whether they are carrying pollen or not. Characteristics such as subspecies and pollen carrying are meant to be used as additional meta information to assess problems such as hive robbing and poor nutrition, respectively. Furthermore, as the work done by Pukhov (2018) will be used as a baseline, the goal is to build upon the provided architecture and improve the results of the classifiers.

6.2.2 Approach & Configurations

From literature review and analysis of the work conducted by Pukhov (2018), in order to improve the obtained results, the following approaches can be defined:

- Application of image filters and mathematical morphology operators, such as opening and closing.
- Hyper-parameter optimization of the classifiers and respective architecture.

On the topic of mathematical morphology, it can be stated that it is a theory which provides a number of useful tools for image analysis. Specifically, it is a theory that concerns the analysis and processing of geometrical structures, through the analysis of planar and spatial structures (C A Glasbey & G W Horgan, 2010; Eidheim, 2007; Hlaváč, 2018). In this theory, pixel intensities are regarded as topological highs instead, allowing for the manipulation of an image in several ways, such as noise removal and edge detection. These effects are usually obtained through the combination of mathematical morphology operators. The most common ones are erosion and dilation. In the case of erosion, it can be defined as the operation where the value of a given pixel is the minimum value of its surrounding pixels, and can be perceived as the removal of small objects and islands, leaving only substantive objects (Mathworks, 2020). In the case of dilation, it can be defined as the operation where the value of a given pixel is the maximum value of its surrounding pixels, being perceived as the operation that makes objects more visible and fills small holes in objects. Opening and closing

operators correspond to an erosion followed by a dilation and a dilation followed by an erosion, respectively (Mathworks, 2020). The opening operator allows the removal of small objects whilst maintaining the shape and size of objects, and the closing operator fills small holes whilst maintaining the shape and size of objects. *Figure 37* and *Figure 38* show examples of the erosion, dilation, opening and closing operators. Due to the impacts mathematical morphology operators can have on images, it can be stated that there is potential in the use of these operators to enhance image features for the DCNN.

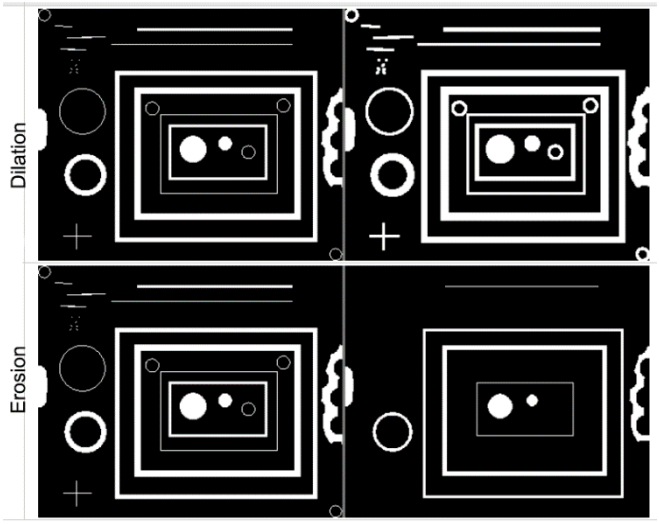

Figure 37 - Example of the erosion and dilation operators. As can be seen, in erosion, the smaller objects disappeared.

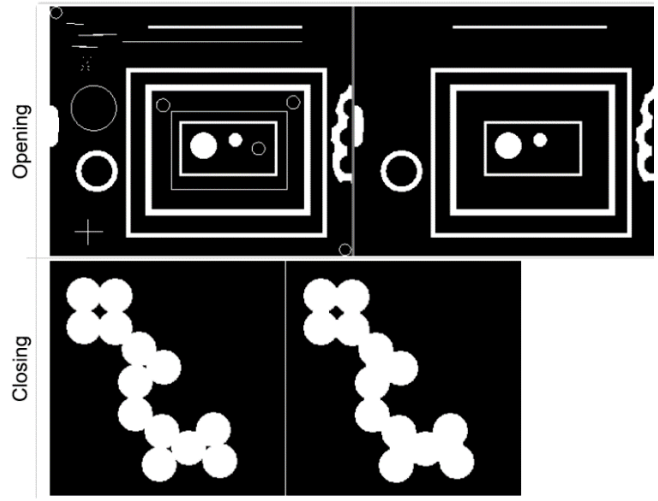

Figure 38 - Example of the opening and closing operators. As can be seen, small objects can be eliminated with opening and holes can be filled with closing, without losing the original size and shape.

Regarding the mathematical morphology, *Table 11* shows the image filters and mathematical morphology operators used to attempt to improve the classification results. Both

mathematical morphology operators and image filters were tested independently of each other. However, the possible values for each of these pre-processing techniques were combined, as to determine their effectiveness at improving the classification results (i.e. applying an opening operator followed by a closing operator). *Figure 39* and *Figure 40* exhibit the application the different image filters and mathematical morphology operators on a bee that is carrying pollen and another that is not, respectively.

Table 11 - Image filters and mathematical morphology operators used on the first task. The goal is to enhance classification results.

Approach	**Techniques**
Image Filters	Colour Dodge, Colour Burn, Hard Light
Mathematical Morphology Operators	Opening, Closing, Dilation, Erosion

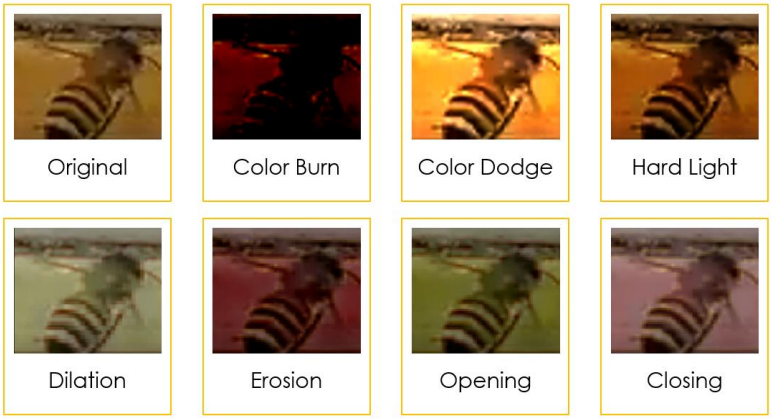

Figure 39 - Application of image filters and mathematical morphology operators in a bee that is not carrying pollen. As can be seen, some filters seem to highlight more information, whilst others seem to make it more difficult to identify visual clues.

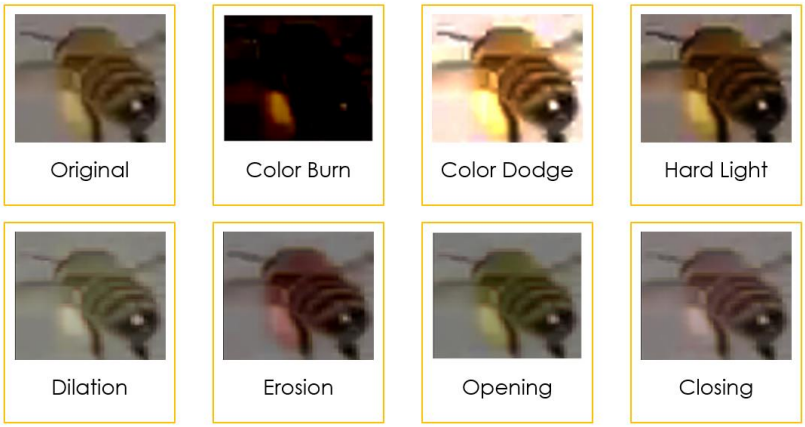

Figure 40 - Application of image filters and mathematical morphology operators in a bee that is carrying pollen. As can be seen, some filters seem to brighten the region where the pollen is

seen (near the bee's legs), whilst others seem to make it more difficult to identify any visual clues.

Regarding the hyper-parameter optimization, the defined approach consists in performing a grid-search of the best values for the architecture, varying the several different parameters it can have. In order to improve the classification results by finding the best combination of values for the CNN architecture of the baseline work, the parameters and respective values shown in *Table 12* were defined. Regarding the variation in the number of layers, preliminary testing showed that there were no significant improvements in increasing the number of convolutional layers past 4. Similarly, the same results were obtained regarding the number of neurons in each layer. Regarding other parameters' possible values, as the goal is to improve the results obtained from the baseline approach, performing an exhaustive search on a wide variety of values is important, as to ensure a desirable confidence on statements about the best configuration for the CNN. Furthermore, several random seeds were used in order to determine which initial configuration of the weights of the CNN showed better results, as well as to assess the overall improvement of the architectural changes to the CNN (i.e. if a change improves the architecture, then it should do so in several random states).

Table 12 - CNN Hyper-parameters and respective values that were optimized. Each parameter was combined with the remaining parameters.

Hyper-parameter	**Possible Values**
Number of Layers	2, 3, 4
Number of Neurons	5, 10, 15, 20, 25, 30, 35, 40
Pooling Methods	AveragePooling2D, MaxPool2D
Activation Functions	ReLU, PReLU (Parametric ReLU), LeakyReLU, SoftMax (last layer only)
Optimizers	Adam, Nadam, Adamax, Adadelta
Loss Function	Mean Square Error, Categorical Crossentropy, Mean Square Logarithmic Error, Poisson, Logcosh, Kullback Leibler Divergence
Random Seeds	42, 100, 200, 300, 400, 500, 600

Furthermore, each of the values of the hyper-parameters were combined with the others in separate phases, as to reduce the total number of combinations tested in one given test setup (which would total to around 516 thousand different runs). After each phase, values that produced significantly poor results were discarded for next setups. This process was repeated until all hyper-parameters were stabilized (i.e. had obtained the best possible results). The last setup was composed of a total of 576 different tested combinations. Furthermore, only the last setup used all random seed values, whereas the others used at least 3.

Each model configuration, for each problem, was run for 20 epochs, with 50 steps per epoch, as to match the same training efforts conducted on the baseline approach. The data set was split into train, test, and validation data, with a split weight of 0.65, 0.25 and 0.10, respectively. The validation set is used during training to provide continuous validation of the model, whilst

the test set is used for the final validation. As mentioned previously, by approaching the same training conditions used in the baseline approach, a further comparison can be made on the beneficial impacts of the defined approaches. Additionally, every time a new configuration of the CNN is executed, the image data generator is reset, as to ensure that any improvements of the classification results happen due to a change in the pre-processing and/or architecture of the CNN, instead of different quality images.

On what concerns both approaches, it can be stated that an increase in the average elapsed time is expected. In the case of the first approach, as the classifier architecture used is the baseline's, with an additional pre-processing step, the elapsed time will always be the one for the baseline plus the elapsed time of applying the pre-processing step. In the case of the second approach, depending on the complexity of the resulting best structure, there can exist a slight increase in the elapsed time. Furthermore, in order to assess whether classifiers were overfitting, plotted graphs with the training and validation loss were used, as can be seen in *Figure 41*.

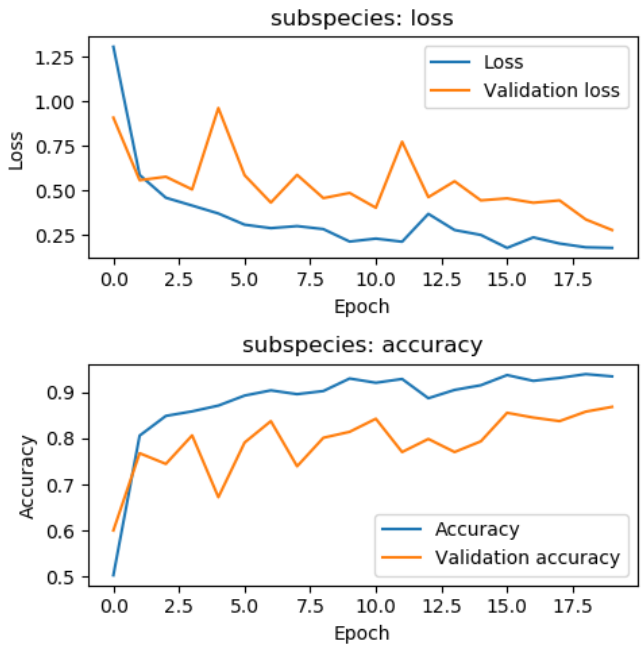

Figure 41 – Example of the loss curves used to assess if a classifier was overfitting.

6.2.3 Results & Discussion

In order to compare the baseline approach's results with the ones obtained during trials, the baseline approach was run for each of the target attributes and was validated with the results stated in the work done by Pukhov (2018), as per *Table 13*. All results shown in this section are relative to the test set.

Table 13 - Best Results obtained using baseline work's optimal CNN (test set).

Measure	***Baseline (Best)***		
	health	*pollen_carrying*	*subspecies*
Accuracy	0.8492	0.9574	0.8654
Precision	0.8200	0.9900	0.9100
Recall	0.8500	0.9600	0.8700
F1 Score	0.8100	0.9700	0.8800
Val. Loss	0.4052	0.1217	0.3319
Elapsed time (ms)	1.02	1.11	1.10

Regarding the first approach, which has the goal of improving the performance of the classifiers using mathematical morphology operators and image filters, each of the effects was applied during the pre-processing phase, when the image is loaded. *Table 14* shows the results obtained from the application of image filters and morphology operators, using the baseline CNN model best configuration present in (Pukhov, 2018).

Table 14 - Best results obtained using image filters and morphology operators applied to the baseline work's optimal CNN (test set).

Measure	***Developed Approach – Filters (Best)***		
	Health	*pollen_carrying*	*subspecies*
Accuracy	0.7092	0.9907	0.6798
Precision	0.7200	0.9900	0.6900
Recall	0.7100	0.9900	0.6800
F1 Score	0.7100	0.9900	0.6700
Val. Loss	1.1614	0.0661	1.3782
Elapsed time (ms)	2.52	2.31	2.31
Operator	Closing	Opening	Closing

Regarding the second approach (hyper-parameter optimization), *Table 15* and *Table 16* show the best results and hyper-parameters obtained, respectively. Additionally, in *Table 16* the random seed used to obtain the results shown is also available, for reproducibility purposes.

Table 15 - Best results obtained with hyper parameter optimization, using the baseline work's optimal CNN (test set).

Measure	***Developed Approach – Hyper-parameters (Best)***		
	Health	*pollen_carrying*	*subspecies*
Accuracy	0.9520	0.9822	0.9126
Precision	0.9500	0.9900	0.9200
Recall	0.9500	0.9800	0.9100
F1 Score	0.9500	0.9900	0.9200
Val. Loss	0.1505	0.4655	0.2420
Elapsed time (ms)	1.47	1.43	1.35

Table 16 - CNN best hyper parameter values for maximizing the classification results, discriminated by attribute. In the table, the random seed that allowed to obtain the best results is also available, for reproducibility purposes (test set).

Attribute Tested	***Best Architecture***	
health	Structure	[15, 35, 30, 10]
	Activation functions:	[ReLU, ReLU, ReLU, SoftMax]
	Pooling method	MaxPooling2D
	Optimizer	adam
	Loss function	categorical cross-entropy
	Random seed	42
pollen_carrying	Structure	[10, 20, 20, 5]
	Activation functions	[ReLU, ReLU, ReLU, SoftMax]
	Pooling method	AveragePooling2D
	Optimizer	adam
	Loss function	categorical cross-entropy
	Random seed	42
subspecies	Structure	[15, 25, 30, 15]
	Activation functions	[ReLU, ReLU, ReLU, SoftMax]
	Pooling method	AveragePooling2D
	Optimizer	adam
	Loss function	categorical cross-entropy
	Random seed	100

From the analysis of the results obtained for each of the approaches, there are two main groups of conclusions that can be obtained. Regarding the use of mathematical morphology operators and image filters, it can be stated that these showed significantly worse results than those found in the baseline approach, with the exception of the pollen carrying attribute. The best results obtained for the species and health attribute using the defined approach had a decrease of at least 15% in the majority of metrics used (e.g. accuracy, precision, and recall). However, the pollen carrying attribute had a slight increase in obtained results, as the baseline approach's accuracy is of 95.74%, whilst the use of the opening mathematical operator yielded an accuracy of 99.07%. In regard to the efficiency (elapsed time), it seems the first approach is slightly less efficient than the baseline counterpart, which is an expected result, since, as stated previously, the opening operator is being applied as a pre-processing step to the baseline's classifier.

Regarding the optimization of the hyper-parameters of the CNN architecture, it can be stated that these showed a significant improvement of the overall results on all three attributes. On the specific cases of the health and species attributes, the best hyper-parameters found were able to increase the accuracy, precision, recall and f1-scores in 10% and 5%, respectively. Pollen carrying corresponds to the attribute with the smallest improvement, as the results of the optimized classifier were up to 2% better than those obtained by the baseline's classifier. In regard to the efficiency of the approach, it can be stated that it seems to be similar to the baseline's, as far as average elapsed time is concerned.

Table 17 – Hypothesis and results obtained for the Shapiro-Wilk test on each classifier comparison (baseline vs optimized approach).

Classifier Comparison	**Hypothesis and Results**
Health (baseline vs optimized)	**H_0**: Baseline's health elapsed times follow a normal distribution. **H_1**: Baseline's health elapsed times do not follow a normal distribution. **W**: 0.4014; **p-value**: 0.0
	H_0: Optimized approach's health elapsed times follow a normal distribution. **H_1**: Optimized approach's health elapsed times do not follow a normal distribution. **W**: 0.6966; **p-value**: $5.3e^{-39}$
Subspecies (baseline vs optimized)	**H_0**: Baseline's subspecies elapsed times follow a normal distribution. **H_1**: Baseline's subspecies elapsed times do not follow a normal distribution. **W**: 0.5089; **p-value**: 0.0
	H_0: Optimized approach's subspecies elapsed times follow a normal distribution. **H_1**: Optimized approach's subspecies elapsed times do not follow a normal distribution. **W**: 0.5910; **p-value**: $5.3e^{-41}$
Pollen Carrying (baseline vs filters)	**H_0**: Baseline's pollen carrying elapsed times follow a normal distribution. **H_1**: Baseline's pollen carrying elapsed times do not follow a normal distribution. **W**: 0.4753; **p-value**: 0.0
	H_0: Filter's pollen carrying elapsed times follow a normal distribution. **H_1**: Filter's pollen carrying elapsed times do not follow a normal distribution. **W**: 0.6466; **p-value**: $3.5e^{-41}$

However, in order to assess the difference between the baseline and the developed classifiers' efficiency, with statistical relevancy and confidence, t-test were conducted to the following pair of samples:

- Baseline and optimized health classifiers.
- Baseline and optimized subspecies classifiers.
- Baseline and image filter pollen carrying classifiers.

In order to assess whether to conduct a Mann-Whitney U or t-test, the distribution of the data must be checked first, as to ensure whether it follows a normal distribution or not. To assess the data's distribution, a Shapiro-Wilk test was used, with the hypothesis, and respective test results, defined in *Table 17*. Furthermore, the skewness and kurtosis values were computed, and the data samples' histogram and boxplots were rendered.

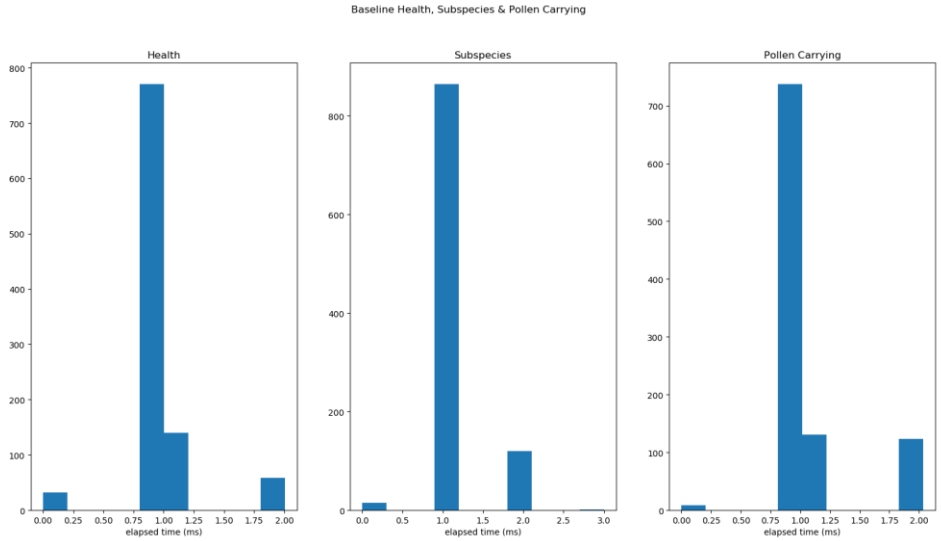

Figure 42 – Histograms of the baseline approach's elapsed time for the health, subspecies and pollen carrying attributes.

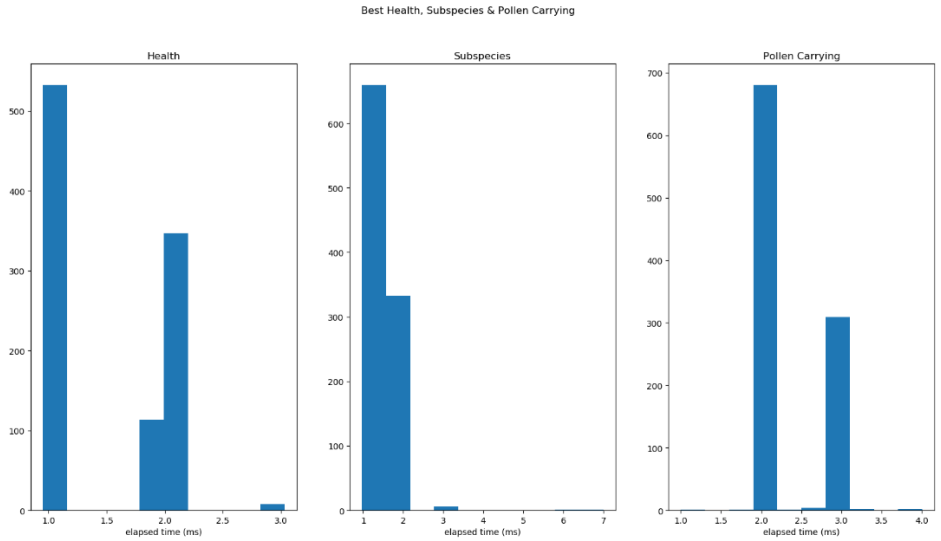

Figure 43 – Histograms of the best obtained models' elapsed time for the health, subspecies and pollen carrying attributes.

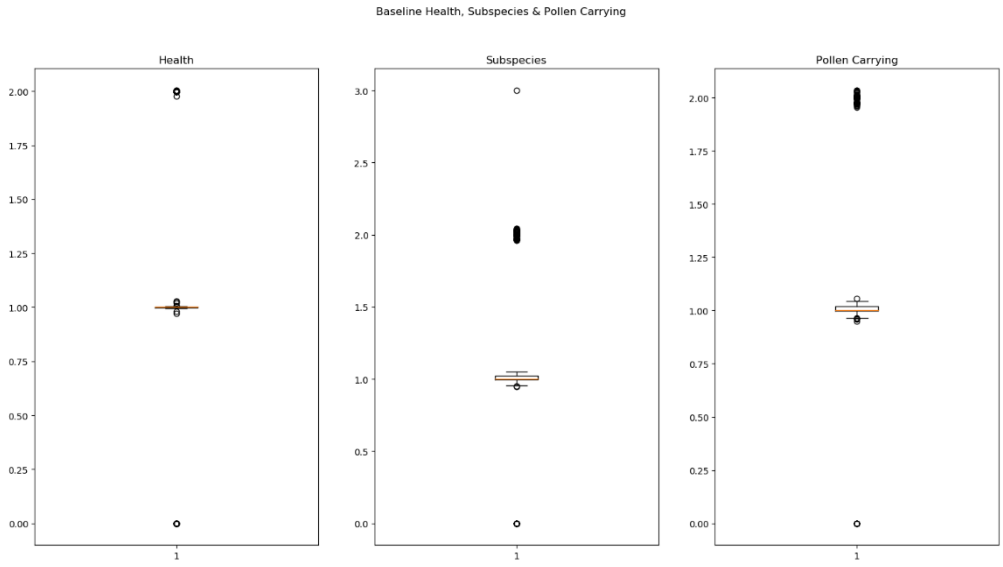

Figure 44 – Boxplots of the baseline approach's elapsed time for the health, subspecies and pollen carrying attributes.

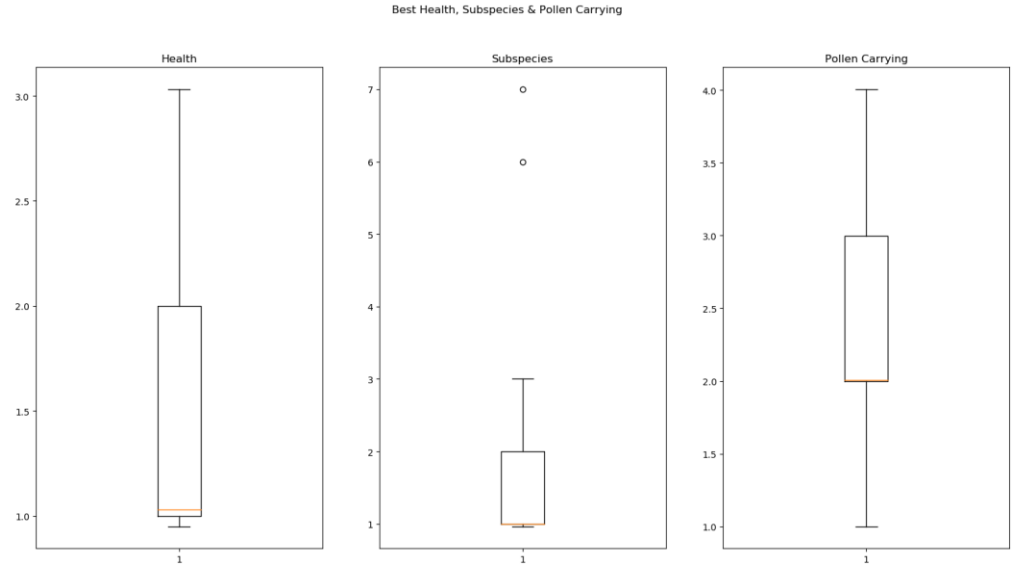

Figure 45 – Boxplots of the best obtained models' elapsed time for the health, subspecies and pollen carrying attributes.

Table 18 - Kurtosis and skewness values obtained for all elapsed time samples. As can be seen, all values suggest a non-normal distribution.

Elapsed Time Data Sample	Kurtosis	Skewness
Baseline Health	7.986	0.7111
Baseline Subspecies	3.835	1.555
Baseline Pollen Carrying	3.334	1.738
Best Approach Health	-1.502	0.2712
Best Approach Subspecies	15.93	2.327
Best Approach Pollen Carrying	-1.082	0.8124

In regard to the results of the Shapiro-Wilk test available in *Table 17*, it can be stated that all elapsed time measurements conducted do not follow a normal distribution, as all exhibit a p-value less than 0.05. Whenever such occurs, the null hypothesis (H_0) must be rejected, therefore meaning that the data does not follow a normal distribution. This conclusion is further supported by the kurtosis and skewness values shown in *Table 18*, as well as the histograms and boxplots in *Figure 42*, *Figure 43*, *Figure 44* and *Figure 45*. As can be seen, all elapsed time samples are right skewed (as the skewness is positive), with the elapsed time of the best approach's health model being the one with the least skewness. Furthermore, from the histograms, it can be stated that the different groups do not have the same shape.

Table 19 – Hypothesis and results obtained for the Mann-Whitney U test on the classifiers' elapsed time data, as to compare the baseline with the optimized approach to assess differences between them.

Classifier Comparison	Hypothesis and Results
Health (baseline vs optimized)	**H_0**: Baseline and optimized have the same average elapsed time. **($u_1=u_2$)** **H_1**: Baseline's average elapsed time is less than the optimized approach's average elapsed time. **($u_1<u_2$)** **p-value**: $1.01e^{-60}$;
Subspecies (baseline vs optimized)	**H_0**: Baseline and optimized have the same average elapsed time. **($u_1=u_2$)** **H_1**: Baseline's average elapsed time is less than the optimized approach's average elapsed time. **($u_1<u_2$)** **p-value**: $6.12e^{-25}$;
Pollen Carrying (baseline vs filters)	**H_0**: Baseline and filters have the same average elapsed time. **($u_1=u_2$)** **H_1**: Baseline's average elapsed time is less than the filter approach's average elapsed time. **($u_1<u_2$)** **p-value**: $4.26e^{-279}$;

As such, the usage of parametric tests such as t-test is not recommended, and the Mann-Whitney U test will be used to assess a significant statistical difference in elapsed times, in regard to the mean ranks. From the results shown in *Table 19*, it can be stated that all null hypothesis (H_0) are rejected, as the p-value of each test is significantly lower than the significance level ($\alpha = 0.05$, p-value $<< \alpha$). As such, it can be concluded that there is a significant statistical difference between the elapsed time of the baseline and optimized approaches. From the results in *Table 13*, *Table 14* and **Erro! A origem da referência não foi encontrada.**, it can be stated that, statistically, the optimized approach is less efficient than the baseline approach. However, it must be noted that the largest difference in average execution times between the baseline and proposed classifiers is of 1.21 milliseconds (which corresponds to the pollen_carrying attribute). On a real-life scenario, the difference is not significantly high to present itself as an issue. Furthermore, all execution times are significantly under the 1 second main objective, meaning that they all comply with the objective in an adequate fashion.

In sum, it can be stated that the results obtained for the bee image health classification task are satisfactory. Based on the work of Pukhov (2018), the developed approaches were able to improve the baseline's results significantly, with the particular case of health attribute classification having an increase of 10% in accuracy, to a total of 95%. Furthermore, the proposed approaches were able to increase the quality of the classification results whilst minimizing the impact on the efficiency of the classifier, with all cases realistically having zero impact in performance.

Considering the objectives defined for this task, as well as the respective hypothesis defined, it can be stated that these results suggest it is possible to develop a solution for the classification of a bee's health status, based on image of a bee, with satisfactory results and performance. Additionally, as stated previously, it is possible to improve the results of the baseline approach. Using the configurations provided by the defined methodology, stakeholders should be able to reproduce results and apply the classifiers in a real-life scenario.

6.3 Bee & Asian Wasp Detection

As stated previously, the goal of the bee and wasp detection task is to correctly detect bees and Asian wasps in images. The developed model should be able identify bees in several different conditions, so that these can be automatically cropped and passed onto the bee health classification models for further analysis. Furthermore, in the case of Asian wasps, proper detection of the Vespidae is important, so that the appropriate mechanisms can be triggered as soon as possible, like alerting local authorities and proceeding to the proper identification of the Vespidae nests and effective removal.

6.3.1 Data Description

Before the adequate description of the data set used for this task, a description of the context behind the data collection process is important. At the time of this thesis' writing, the amount of available data (i.e. images) about *V. velutina* is scarce, with no existing publicly available data sets for object detection (or other purposes), whether in an academic or business context. From conversations with experts in the field, the reasoning behind this fact is that as these Vespidae are quite dangerous, the fast removal of their presence from any given area is a priority and the standard. Furthermore, due to their dangerous nature, footage recordings are rare, as it is common not to be near them for extensive amounts of time. As the standard procedure is to alert local authorities for proper removal, footage of these Vespidae is scarce. Even on the context of businesses, a fast removal of *V. velutina* is desirable, as to reduce the impact of these invaders in the expected yield and profits. Additionally, even though the acquisition of images of bees is a relatively easy task, as any research institutions and/or business can place a camera to gather the data, the difficulty associated to image collection of *V. velutina* images is high, mainly due to the reasons stated, and the fact that institutions do not have "readily available" *V. velutina* hives.

Due to the reasons stated above, the data set collected for this task was obtained from footage of videos (live streams or recordings) in real-life scenarios with non-lab conditions. During data collection, special caution was taken to assess the quality of the footage, as to assess its potential use for the task. Even though the data pertains to live scenarios, it is important to note that different videos can have different recording qualities (i.e. resolution, blurriness, etc.). Due to these conditions, videos whose quality is too high may not be suitable for the task, as realistically, cameras used to monitor bee hives are not high quality (as the volume of data would be too high to store and handle appropriately, much like the case of security cameras). Likewise, videos whose quality is too low may not be suitable, as the respective images may degrade the performance and training of the object detection models (since the images may have too low quality for any significant feature extraction). In regard to the standards defined for this task's particular data collection process, special care was taken so that the collected footage, once resized to the dimensions specified in subsection 5.2.2 (around 520x520), would still comply with the standards also defined in subsection 5.2.2. In regard to the software used to collect the data from online footage, Open Broadcast Software (Jim, 2020) was used.

Finally, regarding the classes of the data collected, during literature review, several papers, articles, and news about *V. velutina* were found. From these, it was possible to conclude that *V. velutina*, commonly referred to as "Asian wasp", is a significant threat to honey bees, and that the species' invasion is currently rising. However, during the data collection process of *V. velutina* (using the terms "Asian wasp" and "*V. velutina"*), footage was found relating to another wasp species called *Vespa mandarina* (*V. mandarina*). From further research on this type of Vespidae, it can be stated that *V. mandarina* is another species of honey predator wasp, original from eastern Asia (McClenaghan et al., 2018; Monceau, Bonnard, et al., 2014). On an invasion context, the difference between *V. velutina* and *V. mandarina* is that the

former is the first invasive Vespidae predator to be accidentally introduced in Europe. Furthermore, whilst *V. velutina* are commonly called “Asian wasps”, *V. mandarina* are called “Giant Asian wasps”, which explains the reason behind the mix of videos of the two species. However, both species are equally dangerous to both humans and honey bees, and recent findings suggest that *V. mandarina* are expected to make a strong appearance on certain regions in Europe & US in upcoming years, including Portugal (Diário de Notícias, 2020; Siqueira, 2020). For these reasons, and in order to prepare the task’s solution so it can aid on the fast and effective removal of this invasive Vespidae in the future, images of *V. mandarina* are also included in the data set.

This task’s data set was extracted from Explore’s honey bee live streams (Explore, 2019b, 2019a), as well as from videos found on YouTube (ITV News, 2016; Ryouta, 2014; VOA News, 2019). Regarding the first live stream, it is RGB footage from the perspective of the landing pad of two different apiaries. The perspective and angle of the camera is different in each apiary and provides clear sight of the bees that are landing, as well as those immediately outside of the apiary, as can be seen in *Figure 46*. The second live stream contains Grayscale images bees inside an apiary, from a top-down perspective, as can be seen in *Figure 47*.

Figure 46 - Example of footage from the first live stream of honey bees. As can be seen, there are different two different perspectives of honey bees' landing.

Figure 47- Example of footage from the second live stream of honey bees.

Figure 48 - Example of annotations for object detection. In the image, bees whose quality is too low (e.g., blurry) are not annotated.

Regarding footage extract from the previously specified videos, only clips whose conditions were similar to those available in Explore's live streams were used, as is the example case shown in *Figure 49*. Furthermore, in clips which contained non-relevant information (such as logos, subtitles, and others), the extracted images only pertain to the portion of the video that is clear of such data. In case it is not possible, the image is excluded from the resulting data set.

Figure 49 - Example of the images extracted from YouTube videos. The figure pertains a bee hive invasion by V. mandarina in a perspective similar to the one found in Explore live streams.

For all footages, images were extracted every 5 frames, totalling 4003 images of honey bee and wasp footage for object detection, of which 1533 images are from the first live stream, 1217 from the second live stream, and the remaining from the specified videos. The images were manually annotated, in accordance to the following criteria (as specified in 5.2.2): only bees and wasps that are at least 50% visible and whose quality is high (i.e. there is no blur associated to the object) are annotated for object detection. *Figure 48* showcases an example of the annotations made following the previously mentioned criteria, using a custom-made tool for annotation.

In order to maximize the performance of the object detection algorithms, the images were manually annotated, resulting in a total of 614 images for the object detection phase. Of these, 32 correspond to empty beehive images, 24 contain images of flowers with no bees, and the

remaining contain images from bees and Asian wasps (both *V. velutina* and *V. mandarina*). The empty images have the goal to further refine the model with examples of scenes where bees normally are present and will be used in algorithms that support training with images that have zero annotations. The resulting total annotated data set has 558 images, of which 439, 17 and 177 are in respect to bees, *V. velutina* and *V. mandarina,* respectively. In regard to the class labels used for bees, *V. velutina* and *V. mandarina*, they are referred in the data set as "bee", "asian_wasp" and "asian_hornet", respectively. However, as far as the amount of object labels found (since there are images that simultaneously contain bees and Asian wasps), the data set contains 1640, 27 and 472 annotated objects of "bee", "asian_wasp" and "asian_hornet", respectively. In order to preserve the aspect ratio of the original images, as well as to comply with the standards defined in subsection 5.2.2, all images and respective annotations were scaled down to 576x290 pixels. Additionally, in order to perform a further validation of the models, a manual qualitative review and analysis was conducted, which centred on using each model to output detection results in videos, for posterior analysis. The used videos consist in clips of the Explore's live streams that are not present in the data set, as well as a BBC Earth clip about ants (BBC Earth, 2017), used to assess the FP rate of a classifier. The ant video was chosen due to their structural similarities with bees and wasps.

Finally, due to the significantly low volume of images for the problem task's DL requirements, the pre-trained weights used for MASK-RCNN, SSD and YOLO are the ones recommended by the implementation's authors (which were previously trained on data sets like COCO and PASCAL VOC), which are the COCO, VG166 and YOLOv3 weights, respectively (Anh, 2018/2020; Ferrari, 2017/2020; Waleed Abdulla, 2017/2020).

6.3.2 Approach & Configurations

From the collected data and literature review analysis, an approach can be defined which consists in the optimization of relevant hyper-parameters of each object detection algorithm, with pre-trained weights, in order to achieve the intended results. Furthermore, considering the class distribution of the data set, two further approaches can be defined:

- Simplification of the object detection task by simplifying the class distribution through the merge of "asian_wasp" and "asian_hornet" classes into one (corresponding only to "asian_wasp").
- Use of the data set as is, with the three-class distribution.

The reason behind the simplification approach resides in the fact that, depending on the context problem, adding more classes can reduce the accuracy of a given model (Ben Reinstate Monica, 2018). However, this kind of behaviour is context dependent, as in some cases, simplification of the number of classes can increase the quality of a model (e.g. distinguishing between 3 different objects may be easier than 10), and in other cases the problem might be too simplified to be solved. Due to these reasons, both approaches were considered during trials, as to assess if the simplification of the wasp classes (by condensing them into one class) improves or deteriorates the model's performance.

On the context of cross validation, a split that guarantees an equal distribution is important, as to avoid overrepresenting/underrepresenting a given class in training/testing, which can have a negative impact on the classifier. However, in the case of object detection, performing a split that complies with the standards is not a trivial task. As the input object of the classifier is an image, which can have multiple outputs, it is not possible to split an image in "parts" so that, for example, the training set has the same amount of examples of a given class that the test set has. Specifically, one image may have 3 examples of bees and 1 example of *V. velutina* (as these are usually seen attacking bees), which is not possible to be split. Essentially, although the input value is atomic, the output is non-atomic. Nonetheless, an algorithm that can perform this kind of split is still important, as it allows for a consistent data set split, even when changing the holdout percentage. For this problem task, a custom algorithm was developed to perform a holdout split into training and testing sets, with approximately equal class distribution. As briefly described in *Code Snippet 3*, the algorithm attempts to perform the distribution according to the following logic:

1. The entire data set is sorted by class, in descending fashion, following the defined class order. This means that when comparing two objects, the number of labels of the first classes is first compared. If they are the same, the number of labels of the second classes is compared, and so on. As the class order changes the final result, the algorithm sorts the classes alphabetically, in descending order.
2. The algorithm computes the approximately expected label count for both training and testing set, using a percentage defined (like the holdout method).
3. From these values, the algorithm first tries to add examples into the set that has the highest portion of the split (i.e. either train or test set), always comparing the current class count is already greater than or equal to the expected class count. On cases the where the class count meets this condition, the example is ignored.
4. With the above, the highest portion set has all objects that it could have without risking adding objects that could belong to the lowest portion set.
5. Steps 2-4 repeat for the lowest portion set, using the remaining objects. In cases where the data set can be perfectly split, the algorithm ends on this step.
6. In cases where there are still leftover items, it means that the data set cannot be split and comply with the split standards (e.g. 70% training, 30% testing) without excluding objects. As such, in these special cases, the algorithm computes the cost (i.e. the 'loss' in compliance) of adding the remaining examples in the following fashion:
    - Adding all examples to the highest portion set.
    - Adding all examples to the lowest portion set.
    - Adding odd-index and even-index examples to the highest portion and lowest portion sets, respectively.
    - Adding odd-index and even-index examples to the lowest portion and highest portion sets, respectively.
7. After computing the costs, the algorithm selects the scenario which minimizes the cost. The function used to compute the cost is the absolute value of the difference between the expected and the resulting number of labels in a set.

```
def train_test_split_object_detection(dataset,split):
    total_classes = build_total_classes_count(dataset)
    per_set_totals = build_per_set_totals(total_classes,split) # splits total_classes into the respective train and testing set total_classes
    high_set = max_class_count(per_set_totals['train'],per_set_totals['test']) # get the set with highest count
    low_set= min_class_count(per_set_totals['train'],per_set_totals['test']) # get set with lowest count

    # defines an inner function to provide the logic of how the examples are added
    def add_examples_to_set(examples,set):
        for example in examples:
            can_add_element = set_has_space_for_labels_of(set,example)
            if can_add_element:
                set.add(element)

    # add examples to respective sets, obtaining the remaining items
    add_examples_to_set(dataset, high_set)
    remaining_items = [elem for elem in dataset if elem not in high_set]
    add_examples_to_set(remaining_items, low_set)
    remaining_items = [elem for elem in dataset if elem not in remaining_items]
    # if there are leftover items, compute costs and choose best strategy accordingly to minimized cost
    # cost = abs(expected_class_count-resulting_class_count)
    if len(remaining_items) > 0:
        all_high = compute_cost_dataset_all(remaining_items, high_set)
        all_low = compute_cost_dataset_all(remaining_items, low_set)
        high_low = compute_cost_dataset_odd_even(remaining_items, high_set, low_set)
        low_high = compute_cost_dataset_odd_even(remaining_items, low_set, high_set)

        compare_and_add_minimize_cost(high_set, low_set, all_high, all_low, high_low, low_high)
    train_set, test_set = map_high_low_sets(high_set,low_set,per_set_totals)
    return train_set,test_set
```

Code Snippet 3 - Pseudo-algorithm to perform a split, with approximately equal distribution, according to the holdout method on data sets for object detection tasks.

For the case of splitting a data set, an optimization algorithm could have been implemented. However, as a proper implementation of an optimization algorithm (such as Genetic Algorithm or Particle Swarm Optimization, among others) would exceed this task's scope and complexity requirements (as it would require a comparison of different optimization approaches just to determine which would be best to use for splitting a data set), this option was discarded.

In regard to the base configuration used for each algorithm, *Table 20* describes the changes performed to the default hyper-parameters recommended for each architecture, corresponding to the base configuration of each algorithm. Some architectures (like SSD) allowed for less variation in the base configuration and respective hyper-parameters. Furthermore, in the case of SSD, the image size used was of 300x300, as is a requirement of the SSD300 algorithm. Additionally, all trials used rescale and translation image transforms to

augment the data set. Finally, *Table 21* describes all hyper-parameters that were optimized in a grid-search fashion, including the different values tested. The decision behind each hyper-parameter chosen was centred on its potential effect in increasing the algorithm's performance (from literature review), as well as the flexibility of the architecture in allowing changes to a given hyper-parameter. For example, SSD has several hyper-parameters whose values must not be changed in order to comply with a given model's weights. Additionally, as with the bee image health classification, combinations of each of the values were performed in different phases. For example, in the case of Mask-RCNN, resnet50 proved to output significantly worse predictions, albeit slightly more efficient. As such, it was discarded when using *4+* and *all* layers (these correspond to only training the first four layers or all layers, for example). In some cases, like Mask-RCNN and YOLO, the batch size had to be lowered, as the algorithms use large amounts of RAM (and an increased batch size would lead to an out-of-memory error). Furthermore, all parameters were tested using two and three classes, corresponding to the simplified and standard approaches, respectively.

Table 20 - Base configuration used for each architecture. The changes are in respect to the default configuration recommended for each algorithm and not the grid-search optimization values.

Architecture	**Hyper-parameter Values**
Mask-RCNN	IMAGE_MAX_DIM: 576
	IMAGE_MIN_DIM: 320
	GPU_COUNT: 1
	IMAGES_PER_GPU: 3
	STEPS_PER_EPOCH: 200
SSD	BATCH_SIZE: 12
YOLO	BATCH_SIZE:4
	WARMUP_EPOCHS: 3
	MAX_INPUT_SIZE: 576
	MIN_INPUT_SIZE: 320

Table 21 – Grid-search values used for the hyper-parameter optimization of each architecture. Highlighted in bold are the default values of each hyper-parameter.

Architecture	**Hyper-parameter Values**
Mask-RCNN	Layers: [**heads**, 3+, 4+, all]
	TRAIN_ROIS_PER_IMAGE: [32,**64**,96,128,256]
	BACKBONE: [resnet50, **resnet101**]
SSD	steps: [**8**, 12; **16**, 20; **32**, 36; **64**, 80; **100**, 120; **300**, 350]
	offsets: [0.4, 0.4, 0.4, 0.4, 0.4, 0.4], **[0.5, 0.5, 0.5, 0.5, 0.5, 0.5]**, [0.6, 0.6, 0.6, 0.6, 0.6, 0.6]
YOLO	grid scales: [0.5, 0.5, 0.5], **[1, 1, 1]**, [1.5, 1.5, 1.5]
	obj_scale: [1,2,**5**,7,10]

Regarding the number of epochs, each classifier was trained until stability was reached (i.e. low variance in loss/accuracy), as can be seen in *Figure 50*, which was usually around the 40-60 epoch mark. Additionally, as with the previous task, loss plots were also used to assess if the models developed in this task were overfitting. Finally, regarding cross-validation, as stated previously, a train test split using a custom algorithm was used, with a split value of 70% for training and 30% for testing. For reproducibility purposes, the random seed value used for all trials was 500.

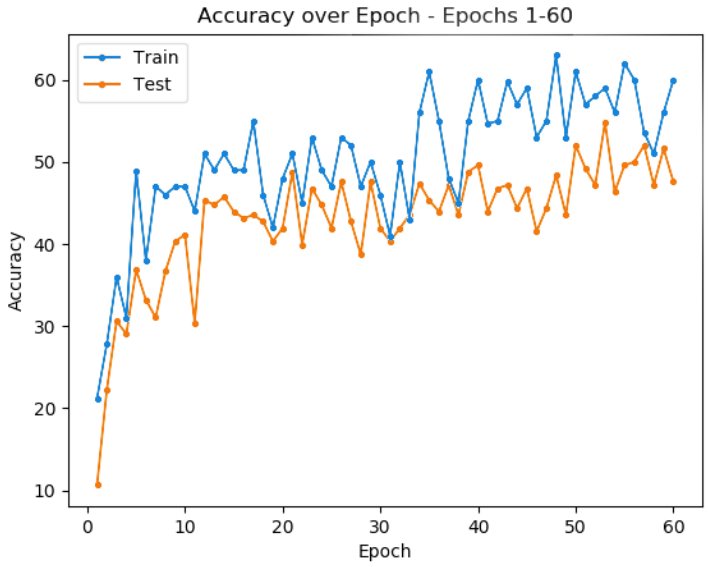

Figure 50 – Accuracy over epoch plot for a trial of Mask-RCNN. As can be seen, starting from epoch 40, there are no significant changes to the results, with both training and testing curves showing a relative stability in the accuracy.

6.3.3 Results & Discussion

Regarding the hyper-parameter optimization conducted for Mask-RCNN, SSD (300) and YOLOv3, *Table 22* and *Table 23* show the best results obtained on the validation set using 2 and 3 classes, respectively.

Table 22 – Accuracy and mAP results obtained for the Mask-RCNN, SSD, and YOLO object detection models using the validation set with 2 classes and the best configuration of hyper-parameters.

	Mask-RCNN	**SSD (300)**	**YOLOv3**
Accuracy (total)	0.7228	0.5304	0.3104
Accuracy (bee; asian_wasp)	(0.6408; 0.8047)	(0.4468; 0.6158)	(0.0411; 0.5798)
mAP	0.6753	0.5106	0.7143
Elapsed Time (ms)	237.6	39.22	53.22

Table 23 - Accuracy and mAP results obtained for the Mask-RCNN, SSD, and YOLO object detection models using the validation set with 3 classes and the best configuration of hyper-parameters.

	Mask-RCNN	**SSD (300)**	**YOLOv3**
Accuracy (total)	0.7720	0.5138	0.1735
Accuracy (bee; asian_wasp; asian_hornet)	(0.4371; 1.0; 0.8789)	(0.3299; 0.3333; 0.8782)	(0.0453; 0.0145; 0.4608)
mAP	0.7162	0.4824	0.6362
Elapsed Time (ms)	237.4	39.24	53.93

In regard to the results obtained for the 2 and the 3 classes approaches, it can be observed that there seems not to exist a consensus in the changes to the obtained results when using different class counts. In the case of Mask-RCNN, there seems to exist an improvement to the model's accuracy when using 3 classes, albeit small. However, in the case of SSD, there seems to exist a detriment to the model's accuracy when using 3 classes, albeit also small. Furthermore, when comparing the results of the YOLO approach, using 3 classes significantly reduces the accuracy of the model (around 14%). From these results, it is not possible to conclude about the over-simplification scenario defined previously, as there is not a global difference between all models (i.e. some models increased in accuracy, others decreased).

The best hyper-parameter values obtained for each model in *Table 24* are in respect to the approach where the model had the best results (i.e. 3 classes for Mask-RCNN, 2 classes for SSD and YOLO). It is important to state that during optimization of the SSD and YOLO classifiers, the best parameters found were the default values suggested by the respective implementations and papers. The reason behind this seems to be that both architectures are restrictive to the parameters that can be changed (as they need to match their original architecture so that the pre-trained weights can be properly loaded). Furthermore, the parameters that can be changed are also sensitive to changes and nontrivial to tune, which can lead to a significant decrease in their performance (as parameters that affect the anchors of the object detection models will impact its classification performance of the divided subspaces) (Christiansen, 2019; T. Yang et al., 2018).

Table 24 - Best hyper-parameter values obtained for each object detection model.

Architecture	**Hyper-parameters**
Mask-RCNN (3 classes)	Layers: all
	TRAIN_ROIS_PER_IMAGE: 256
	BACKBONE: resnet101]
SSD (2 classes)	steps: [8; 16; 32; 64; 100; 300]
	offsets: [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
YOLO (2 classes)	grid scales: [1, 1, 1]
	obj_scale: 5

In regard to the efficiency of each model, from the results shown in *Table 22* and *Table 23*, it seems that the Mask-RCNN is the least efficient model, averaging an elapsed time of 237ms. This is expected, as the architecture is known to be quite slow. In fact, as stated in the literature review (subsection 2.3.3), Mask-RCNN trades-off detection efficiency for the robust training process and higher quality detection. In the case of SSD and YOLO, there seems to exist a difference between the efficiency of both object detection models, albeit not as significant as with Mask-RCNN. In order to assess, with statistical confidence, if there is a difference between the elapsed time of each object detection model, statistical testing was conducted, using each model's best approach (i.e. the one that obtained the best results).

Before assessing the differences in the efficiency of the obtained models, the distribution of the data samples was checked, as to ensure whether they followed a normal distribution or not, using a Shapiro-Wilk test with the hypothesis and results shown in *Table 25*. Additionally, to further validate the results obtained by the Shapiro-Wilk test, the skewness and kurtosis values were computed, and the data samples' histogram and boxplots were rendered.

Table 25 - Hypothesis and results obtained for the Shapiro-Wilk test on Mask-RCNN, SSD and YOLO elapsed time data sets, including the skewness and kurtosis values.

Object Detection Model	**Hypothesis and Results**
Mask-RCNN (3 classes)	**H_0**: Mask-RCNN (3 classes) elapsed times' data follows a normal distribution. **H_1**: Mask-RCNN (3 classes) elapsed times' data does not follow a normal distribution. **W**: 0.9598; **p-value**: $6.13e^{-16}$ **Kurtosis:** 2.605; **Skewness:** 0.7587
SSD (2 classes)	**H_0**: SSD (2 classes) elapsed times' data follows a normal distribution. **H_1**: SSD (2 classes) elapsed times' data does not follow a normal distribution. **W**: 0.5758; **p-value**: $7.14e^{-44}$ **Kurtosis:** 3.664; **Skewness:** 1.595
YOLO (2 classes)	**H_0**: YOLO (2 classes) elapsed times' data follows a normal distribution. **H_1**: YOLO (2 classes) elapsed times' data does not follow a normal distribution. **W**: 0.6916; **p-value**: $1.68e^{-44}$ **Kurtosis:** 44.68; **Skewness:** 4.582

In regard to the results of the Shapiro-Wilk test, it can be stated that Mask-RCNN, SSD and YOLO's elapsed time data do not follow a normal distribution, as all data samples exhibit a p-value less than 0.05, which states that the null hypothesis can be rejected. This is further evidenced by the boxplots shown in *Figure 51*, the histograms shown in *Figure 52* and the kurtosis and skewness values in *Table 25*.

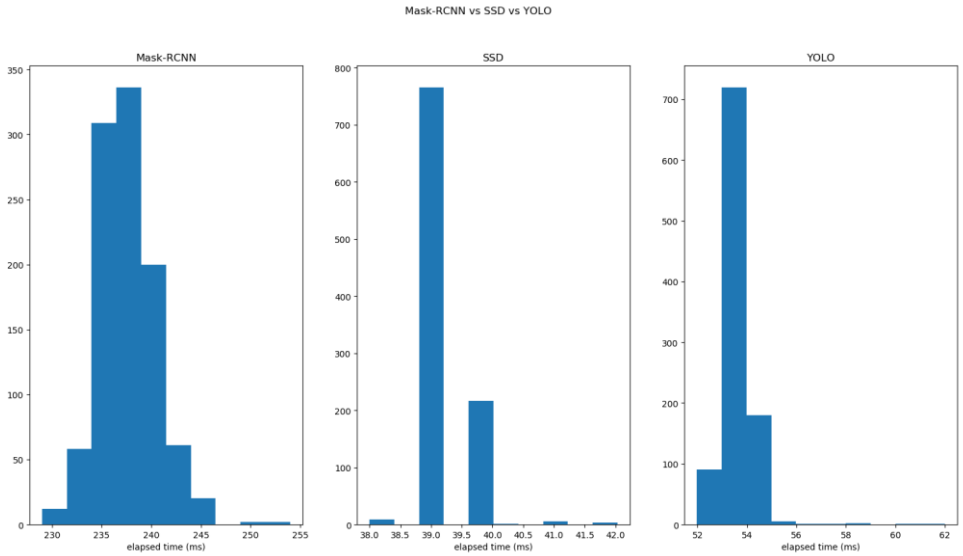

Figure 51 - Histogram of the Mask-RCNN, SSD & YOLO time elapsed values

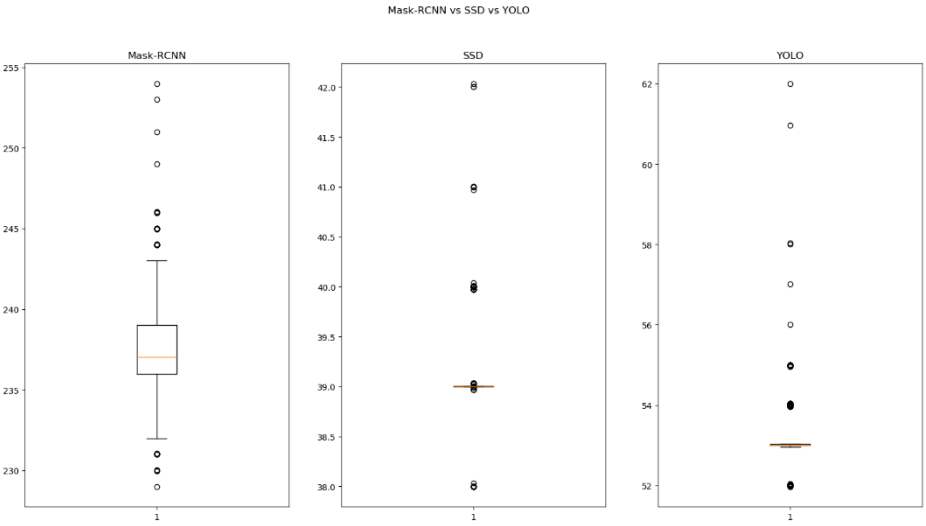

Figure 52 - Box plots of the Mask-RCNN, SSD & YOLO elapsed times. Several outliers can be visualized in the box plot.

Based on these results, it is possible to conclude that there is evidence that the data is right skewed, as the value is quite positive, with a narrow distribution, as kurtosis is higher than 1 (S. Brown, 2016; McNeese, 2016; SmartPLS, 2020). Based on these results, the following Kruskal test was performed:

- **H_0**: All object detection models have the same average elapsed time. **($u_1 = u_2 = u_3$)**
- **H_1**: At least two object detection models have different average elapsed times. **($u_{1<>}u_2$ V $u_{1<>}u_3$ V $u_{2<>}u_3$)**
    - **p-value:** 0.0

As the obtained p-value of the Kruskal test is significantly lower than the significance level α (0.05), the null hypothesis is rejected. As such, it cannot be stated that all object detection models have the same average elapsed time (i.e. there is significant difference in average elapsed times on at least two groups). In order to determine which elapsed times are different, as well as their respective relation, the Mann-Whitney U tests detailed in *Table 26* were conducted, in regard to the mean ranks, as the shapes of the data are not the same.

Table 26 – Hypothesis and results of the Mann-Whitney U tests conducted for the Mask-RCN, SSD and YOLO elapsed times.

Hypothesis	**Results**
H_0: Mask-RCNN and YOLO have the same average elapsed time ($u_1=u_2$) **H_1:** Mask-RCNN's elapsed time is greater than YOLO's ($u_1>u_2$)	**p-value**: 0.0;
H_0: YOLO and SSD have the same average elapsed time ($u_1=u_2$) **H_1:** YOLO's elapsed time is greater than SSD's ($u_1>u_2$)	**p-value**: 0.0;

As both obtained p-values are significantly lower than the significance level α (0.05), the following can be stated:

- Mask-RCNN's object detection model is less efficient than YOLO's.
- YOLO's object detection model is less efficient than SSD's.

From the above, it can be concluded that, by direct relationship of comparison, Mask-RCNN is the least efficient object detection model, whilst SSD is the most efficient detection model, out of all 3 tested models. Furthermore, comparing the difference in averages, it can be stated that although SSD is 35% more efficient than YOLO, SSD and YOLO are 605% and 440% more efficient than Mask-RCNN, respectively, which, comparatively, is a significant increase in efficiency. However, it must also be stated that all tested object detection models comply with the established maximum elapsed time (which is under 1 second).

In regard to the obtained results for both approaches, it can be stated that these are modest. In the case of Mask-RCNN, it was possible to obtain an object detection model close to the desirable accuracy for this thesis' task. However, in the case of SSD and YOLO, these exhibit low accuracy, and often inconsistent with the mAP values obtained. To further shed light onto the details of the reason behind the modest accuracy of the object detection models, the remainder of this section will describe a qualitative analysis conducted on videos with the prediction outputs of the models produced by Mask-RCNN, SSD and YOLO.

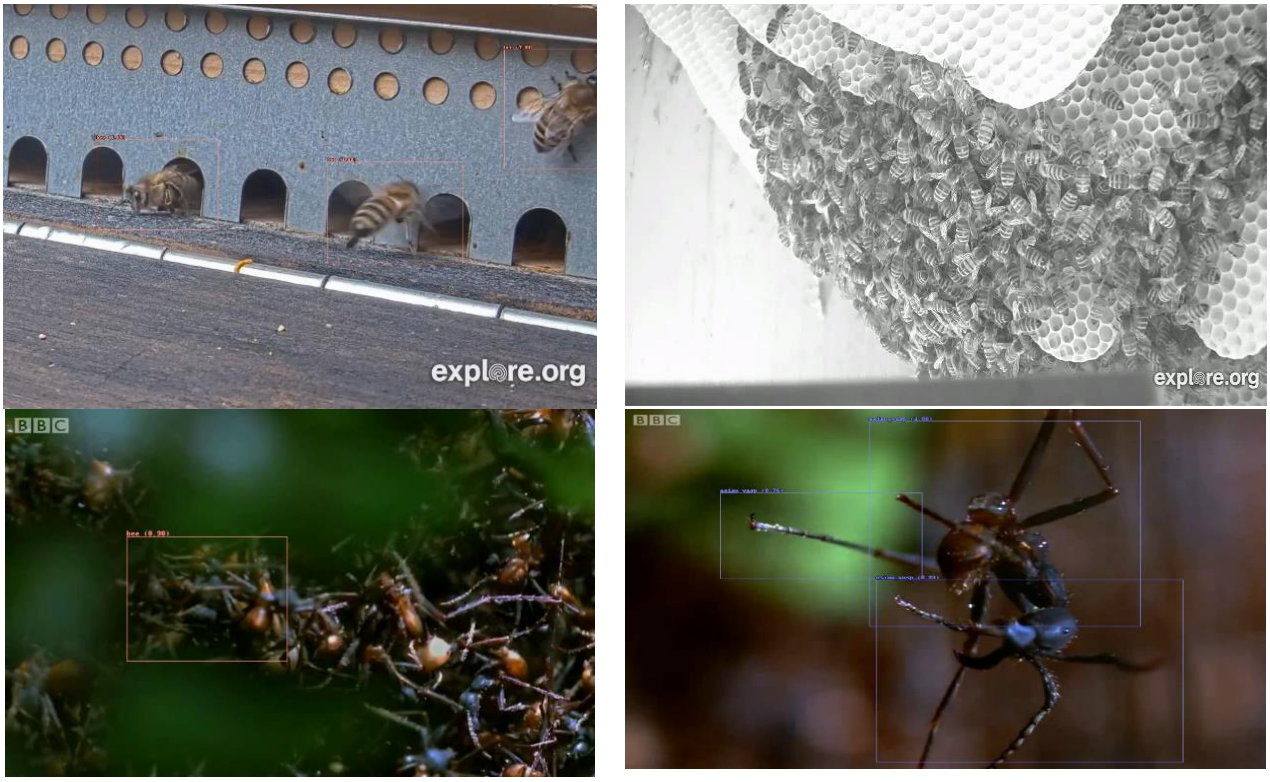

Figure 53 - Example footage of Mask-RCNN applied to video. As can be seen, although the algorithm can detect bees (in red) in environments like landing pads, it fails to detect them in a swarm and even confuses parts of ants with a V. velutina (in blue).

In the case of the Mask-RCNN model, *Figure 53* shows a few example images extracted from videos generated by the model. From the images, it is possible to describe that in controlled perspectives, such as the one shown in the top-left image, the model seems to have a high accuracy and satisfactory detection results. However, when changing the perspective of the input image to a less controlled one (as is the case of the top-right image, which has several bees overlaid), the Mask-RCNN model exhibits poor results and fails to detect bees. This type of behaviour seems consistent throughout the videos generated by the network – on "controlled" perspectives (landing pad), which are relatively easy, the model satisfies detection criteria. When changing from the landing pad perspectives into other scenes (even in "free-form" scenes where the camera is moving), it is possible to assess that the model has more difficulties detecting objects correctly. Furthermore, the bottom images show examples of ants from a BBC documentary that were incorrectly detected as bees and Asian wasps, showing a clear example of the FP generated by the model. As can be seen in the bottom-right image, the developed Mask-RCNN model assumes that a single ant corresponds to 3 Asian wasps, providing insights as to the features that the model may be using to assess whether a given object is an Asian wasp.

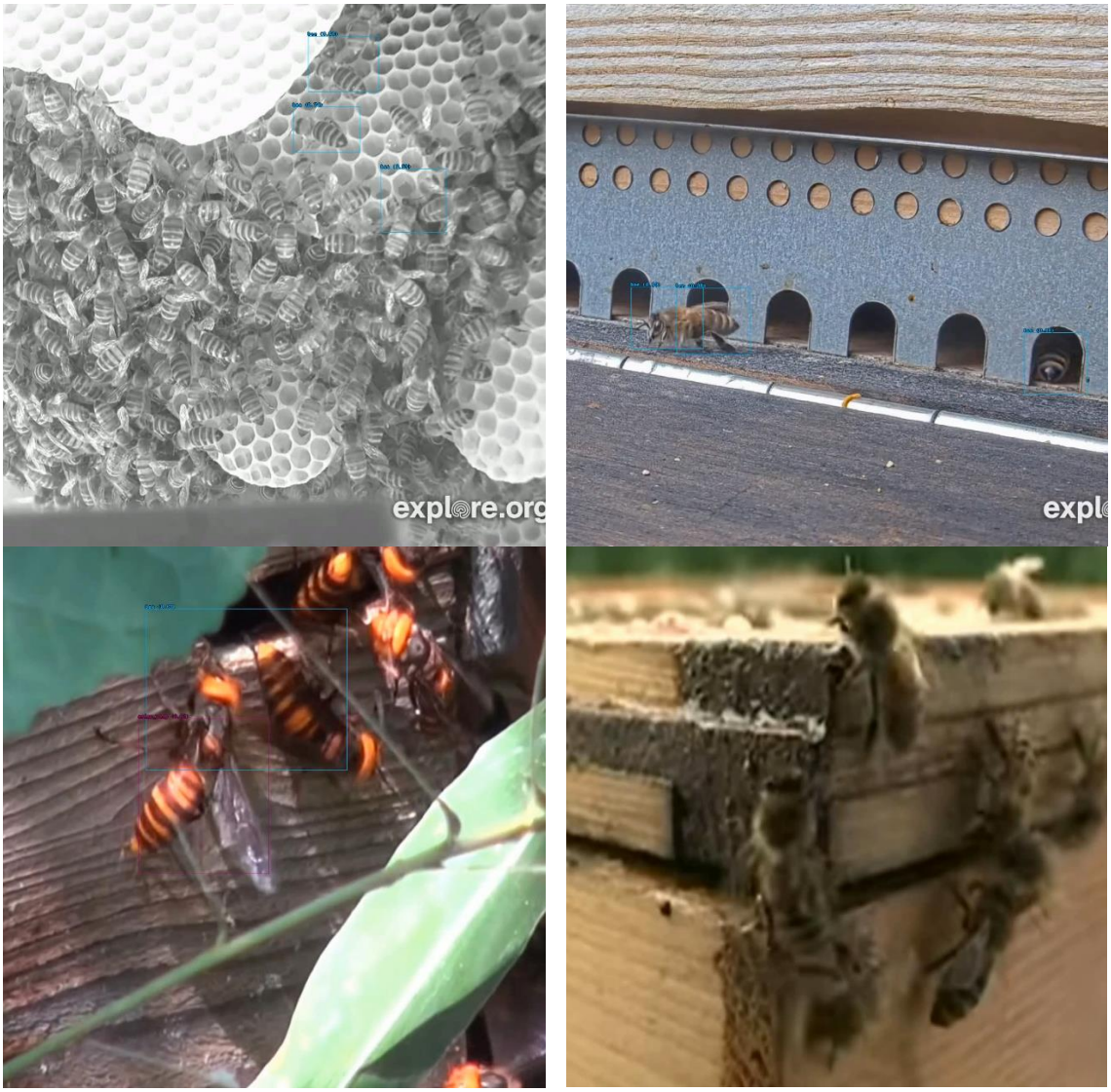

Figure 54 - Example footage of SSD applied to video. As can be seen, although the algorithm can detect some bees (light blue) in the grey version (alongside multiple bees), it cannot detect Asian wasps (in purple) and bees in a vast amount of environments like landing pads and hives, exhibiting extremely low TP ratio.

In the case of the SSD model, *Figure 54* shows a few example images extracted from videos generated by the model. From the images, it is possible to conclude that, overall, the model seems to have a low TP account. Contrary to the Mask-RCNN counterpart, the SSD model can detect some of the bees available in the top-left image, albeit some labels are being applied to two or more bees. However, the model frequently exhibits a behaviour where parts of the bee (like the abdomen) are also considered a bee, as is evidenced by the top-right image, where the same bee is considered as two different bees. Furthermore, the algorithm significantly fails at detecting Asian wasps, as seen in the bottom-left image, where one is mistaken as a bee and there are others who are not even identified by the model. This miss detection case also happens in lower quality settings with bees, as seen in the bottom-right image. From these images, it can be observed that the model seems to only provide modest

results in the case of controlled scenarios (like the top-right image), albeit it exhibits a rather low TP count.

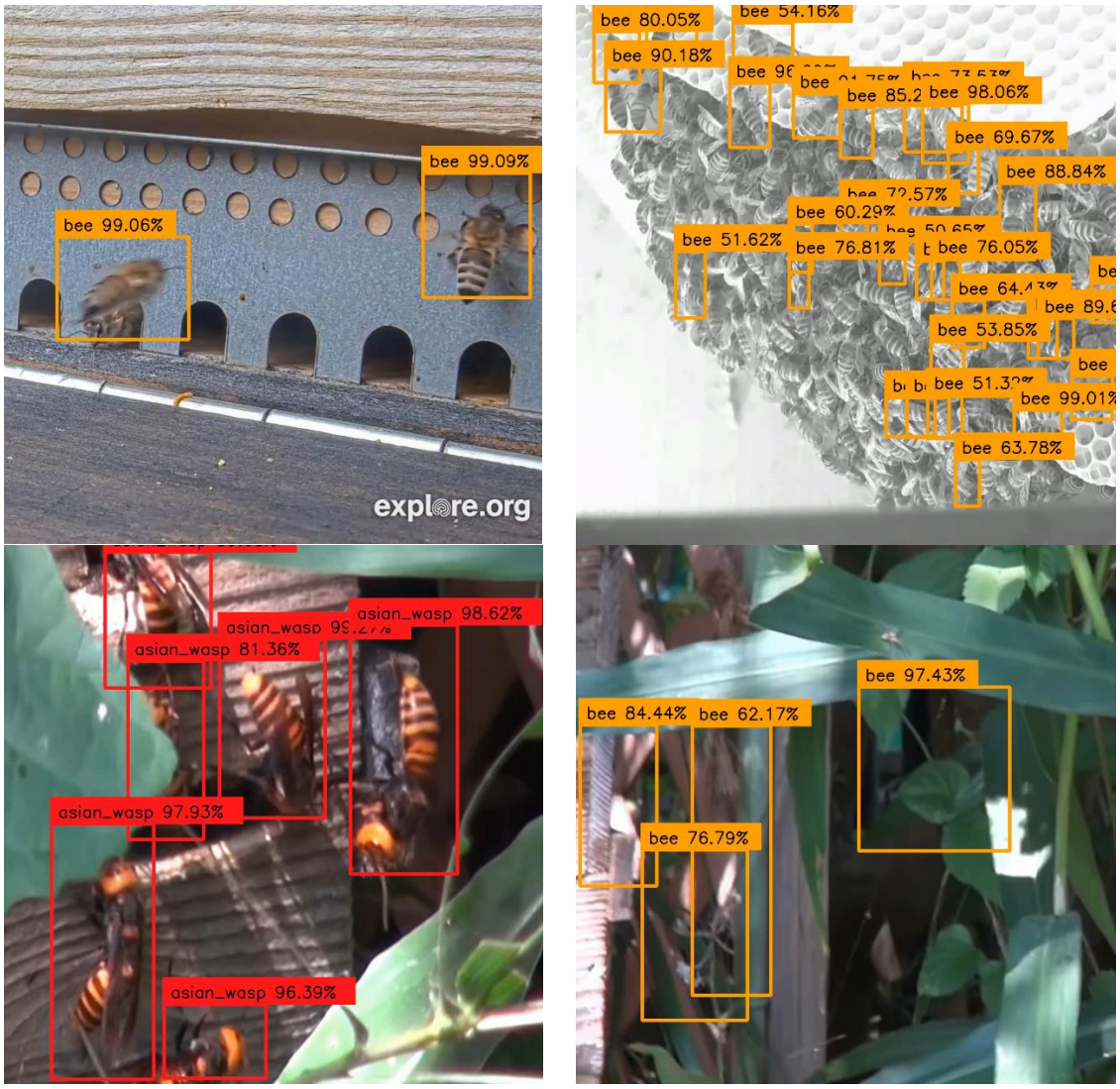

Figure 55 - Example footage of YOLO applied to video. As can be seen, although the algorithm can detect bees (orange) and Asian wasps (in red) in a vast amount of environments like landing pads and hives, it also showcases a lot of FP (as seen from the flora near the bee hive).

Finally, in the case of the YOLO model, *Figure 55* shows a few example images extracted from videos generated by the model. From the images, it is possible to observe that the SSD model has a surprisingly high accuracy, despite the results shown in *Table 22*. The top-left image shows accuracy quality in the detection results comparable to those of the Mask-RCNN model. Furthermore, of all models tested, it was the only model that was able to identify a large amount of bees in the scenario described by the top-right image, albeit with a small amount of FP (e.g. one detection mask for two bees and honey combs detected as bees). This behaviour is consistent in the footage observed from the generated videos, including the scenario described in the bottom-left image (where all Asian wasps are properly detected,

even with foreground objects hiding them). However, the high FP count observed in clips of the generated videos is also consistent, as can be seen on the bottom-right image where the detection model incorrectly states that several parts of the flora are bees.

From the qualitative analysis conducted, it is possible to assess evidence of images present in the data set whose detection difficulty is significantly high, which can contribute to the low accuracy observed in some of the developed models. Furthermore, from the analysis of the videos generated by all 3 best models, it seems that the inclusion of a wide variety of image qualities, with different perspectives, angles and pixel density may have been too ambitious for a first approach at detecting bees and Asian wasps. This is evidenced in the videos by the models' ease in detection of objects in images whose perspectives correspond to the landing pad of bee hives, as well as their respective difficulty in detecting objects in different background perspectives. As such, from the analysis of the obtained results in this task, it is possible to state that these are modest, with particular interest in the quality of the Mask-RCNN and YOLO models, with a special remark on the latter's accuracy in free-form video footage. Furthermore, it was concluded that the SSD model is the most efficient object detection model, with Mask-RCNN being the least efficient model. Annex A showcases more examples of the images generated by each of the object detection models.

On the implications of the obtained results, considering the hypothesis defined for the task, it can be stated that the results allow for the proper distinction of efficiency between each model, and in the case of the quality, albeit modest, the results also allow for comparison. Considering the objectives defined for this task, it can be stated that only the efficiency aspect was achieved, as no model was able to obtain an accuracy starting around 80% with the used data set. However, from the qualitative analysis, a further set of implications can be extracted. All models shared a common scenario – when detecting bees and Asian wasps in a controlled scenario (i.e. in a perspective and angle of a landing pad, usually directed towards the entrance of the hive), they were able to have satisfactory detection results. Furthermore, YOLO had a more robust detection rate between different angle and perspective, albeit it also showcased more FP. Mask-RCNN had a less robust detection rate in significantly different angles and perspectives, but exhibited less FP. SSD was the most efficient model out of the 3 tested models, albeit with slightly less detection rate present in the generated videos, when compared do Mask-RCNN and YOLO. Considering the methodology defined in this task, including the algorithms used to rigorously evaluate the accuracy of the models and ensure a balanced data set in accordance to the class distribution, it can be concluded that stakeholders who intend to develop an object detection model for the detection of bees and Asian wasps can obtain satisfactory results using controlled perspectives and algorithms such as YOLO, Mask-RCNN and SSD. In the case of generalizing for the detection in a variety of angles, perspectives and scenarios, it can be stated that the work done in this task provides insights to the related challenges, including the different configurations that were tested and the obtained issues in generalizing models to images with different conditions.

6.4 Bee Audio Health Classification

The goal of the bee audio health classification task is to develop a classifier that can identify health problems in audio clips of bees, similar to the bee image health classification task. The developed model should be able to detect different health states, so that the information can be used by a monitoring system to inform beekeepers of any issues that may be occurring. When paired with the bee image health classifier, it should add further validation to any obtained results.

6.4.1 Data Description

The data set used for this task has been obtained from the work of Inês Nolasco & Benetos (2018). The goal of the authors' work was to develop an annotated data set based on a selected set of recordings acquired in the context of two different projects: the Open Source Beehive and the NU-Hive projects (Cecchi et al., 2018; OSBeeHives, 2020), which both had the goal of developing a beehive monitoring system similar to the one proposed in this thesis' audio task. The recordings of the data set are a mix between citizen science initiative recordings (where general public recorded the sounds of their beehives and registered the hive state at the moment), and recordings that were collected in a more homogenous and controlled environment, using only two bee hives and the health states "queen bee is present" and "queen bee is not present". The entire data set contains recordings that are in respect to hive and external noises, with the example of the second type of recordings containing external cars of traffic, car honks and birds. Each recording of the data set has a respective ".lab" file which contains time segments labelled as "Bee" or "NoBee", depending on whether the perceived sound being of bees or external to the hive. In regard to the 'health' status of each recording, the file's name contains the output label (e.g. "MissingQueen"). The data set contains a total of 78 recordings with varying lengths, totalling 12 hours of audio, of which 25% corresponds to "NoBee" audio. Regarding the health state, 49.26%, 25.15% and 25.57% of recorded audio corresponds to the "MissingQueen", "QueenBee" (audio relative to a queen bee) and "Active" labels, respectively. All audio recordings have 16 kHz sample rate.

From the gathered data set, it can be stated that the specific goal of the problem is to classify audio clips of bees according to their health state, such as "Active", "QueenBee" and "MissingQueen". Additionally, as the data set allows for a distinction between audio clips that pertain to a bee or to external noise, this task will also have an additional study in regard to the development of a classifier that can distinguish between bee sounds and random noise. However, this is an additional task that can be conducted due to the data set and not a necessity. Even though health classification through audio must occur only on bee related audio clips, this pre-condition can be assured by using a microphone inside (or near the entrance) of the hive, mitigating any errors of having audio that does not corresponds to bee sounds.

Finally, regarding the quality of the audio recordings, it can be stated that it seems some difficulties can arise in the classification process, as some audio recordings have significantly different quality than others (as is expected from the data collection process described above). This difference in quality is present in the low contrast of some audio features, as well as the presence of constant noise (e.g. a 'beep' sound that lasts for the entire length of a given audio sample). Regardless, the data set will still be used for this task, due to the scarce availability of audio data for health detection, as well as time constraints associated to a novel data collection process for this thesis' purposes.

6.4.2 Approach & Configurations

Considering the available data set, as well as the literature review conducted in subsection 2.4, the approach defined for this task consists in the use of Long Short-Term Memory (LSTM) and LSTM+CNN to perform bee health classification through audio, and comparing their respective performance, optimizing hyper-parameters whenever possible. Furthermore, in the case of pre-processing steps that are applied to the original data set, the following set can be identified:

- **Down-sampling of all audio files to 8 kHz** – as the data set is very large, using 16 kHz would quickly exhaust all available memory on the computational environment. It was estimated that, in order to have the entire data set in memory, it would require more than 64GB of RAM. Additionally, when comparing loading efficiency between using the pre-processed data in disk and in RAM, it was concluded that training sessions that lasted around 1-2 hours would increase in time to 6-8 hours (around a 5-6x increase in time). This amount of time per training session would be impractical for this thesis' scope.
- **Conversion of audio data to time series data** – the audio data had to be converted into an adequate time series format for the LSTM and the LSTM+CNN, as both have different expected input formats. For these purposes, both algorithms analyse input objects corresponding to an audio clip with 1 second length, meaning that the sliding window used in the data set equalled the sample rate (as to compose an entire second of audio).
- **Usage of MFCC** – for all different configurations of the training sessions, two different types of data were used. The first corresponded to the raw information of the audio file (raw bytes). The second uses MFCC instead of the raw data of the file. The reason behind using MFCC consists in determining if these can increase the performance of the algorithm. For this second type of data, adequate pre-processing was conducted, as required for the data type conversion. For MFFC generation, the LibROSA package (Librosa, 2018) was used with default parameters. Librosa is a python package for audio analysis.

Table 27 describes the different configurations of the LSTM and LSTM+CNN used throughout the different trials. As with other tasks, the different values shown for a given hyper-parameter were combined with the remaining values. Furthermore, as the data set is

sufficiently large, even after transforming it so that all classes have the same distribution (over 1.1M examples for the “bee/nobee” task and over 28.6k examples for the health task), a holdout method with 50% split was used for the training and testing sets. *Figure 56* and *Figure 57* provide an example of the architectures used for the LSTM and LSTM+CNN approaches, respectively.

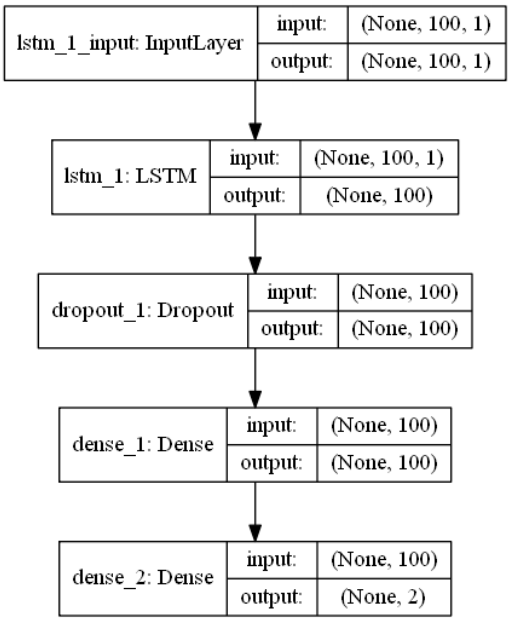

Figure 56 - Example of the architecture used for the LSTM approach.

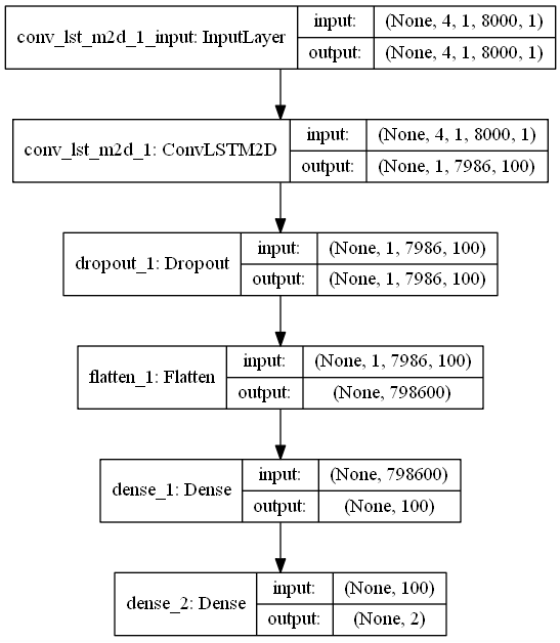

Figure 57 - Example of the architecture used for the LSTM+CNN approach

Table 27 – Hyper-parameters configurations tested for the LSTM and LSTM+CNN in both "bee/nobee" and health classification tasks.

Algorithm	Hyper-parameter Values
LSTM	LSTM Nodes: [10,20,30,40,50,60,70,80,90,100]
	Dense Nodes: [10,20,30,40,50,60,70,80,90,100]
LSTM+CNN	LSTMCTNN_filters: [10,32,48,64,80,100]
	Kernel_size: [(1,1),(1,3),(1,5),(1,10),(1,15)]
	Dense Nodes: [1,40,50,60,100]

Finally, it must be stated that for the case of detecting the health state of the data, only the segments of audio that pertained bee sounds were used, as per the .lab files, as to minimize any errors from inadequate classification of the "bee/nobee" classification models.

6.4.3 Results & Discussion

Regarding the raw data approach, *Table 28* shows the best results obtained for the LSTM and LSTM+CNN classifiers on both tasks. All results shown are in respect to the validation set.

Table 28 – Best results obtained for the LSTM approach in the "bee/nobee" and health tasks using raw data.

Metric & Task	LSTM Result	LSTM+CNN Result
Accuracy (bee/nobee)	0.6672	0.5894
Accuracy (health)	0.5205	0.5012
Elapsed Time (bee/nobee) (ms)	1196	35.93
Elapsed Time (health) (ms)	1201	34.56

From analysis of the obtained results, it can be stated that these are quite poor in quality. As this task's goal is to obtain models with an accuracy starting at around 80%, the current results for the raw data approach are quite unsatisfying and are comparable to those expected by a random number generator (which is around the 50-60% accuracy). Furthermore, from literature review, these obtained results for each of the architectures are odd, as they are expected to usually have at least over 70-75% accuracy, without any optimization. Considering the results obtained with the raw data approach, the next tested approach was the one which consisted in using MFCC values of the raw data instead. *Table 29* shows the best obtained results for this second approach.

Table 29 – Best results obtained for the LSTM approach in the "bee/nobee" and health tasks using the MFCC data.

Metric & Task	LSTM Result	LSTM+CNN Result
Accuracy (bee/nobee)	0.7243	0.6911
Accuracy (health)	0.5804	0.5379
Elapsed Time (bee/nobee) (ms)	1355	184.21
Elapsed Time (health) (ms)	1351	187.93

Comparing the best results obtained from the raw data and MFCC approach, it can be stated that the MFCC approach allowed for a slight increase in accuracy. However, like the results obtained previously, the MFCC approach's results are still under the objectives defined for this task, which state that the accuracy of the developed models should be around 80% or more. Furthermore, literature review conducted also suggests that these results for the model, using MFCC values, are oddly low for the algorithms used. Considering the unsatisfactory results, as well as the previously mentioned low quality of the data set used, an analysis of the potential cause of the poor results is important. Specifically, several validations were performed to the developed code regarding pre-processing, data loading, and model validation, with the goal of determining if any of these steps contained erroneous code. From these validations, no errors were found in the underlying architecture's implementation, or any other steps of the process. As such, since these validations were conducted, an assessment must be conducted on whether the causes of the poor results have a potential origin in the data set or in the methodology used.

In order to conduct the above assessment, the following strategy was defined: keep the same methodology (i.e. data loading, pre-processing, architecture, default hyper-parameters, and validation method) and use a different data set. With this strategy, if the problem is from the methodology itself, then results should show a similar behaviour. Otherwise, it can be stated that there is evidence that the data set's quality is too low to allow for a proper classification model. As such, the data set used for this assessment was the BUZZ1 data set, which corresponds to the same used by Kulyukin et al. (2018). As stated in subsection 2.5, the authors were able to obtain an accuracy of 95.21% on the BUZZ1 data set with their methodology (which consisted of a simple CNN). The data set of BUZZ1 contains annotated examples for the classification of audio into "bee", "cricket" or "noise" labels. From analysis of the authors' open-sourced code, these used raw data as input objects for the classifier. As such, the raw data of the audio files was used for this assessment, which, using the same cross validation method as the one defined for the above approaches (i.e. holdout 50% with equal class distribution), corresponds to over 656k examples for training and testing. *Table 30* shows the results obtained from the assessment, using the validation set and an LSTM, as well as the results reported by the Kulyukin et al. (2018).

Table 30 - Results obtained using an LSTM (100 LSTM nodes, 100 Dense Nodes) and the BUZZ1 data set.

Metric & Task	Defined Approach's Result	State-of-the-Art's Result
Accuracy	0.9782	0.9521
Elapsed Time (ms)	17.58	N/A

As can be seen from the obtained results in *Table 30*, it can be stated that these correspond to the ones expected by the LSTM algorithm (with 100 LSTM and Dense nodes). Furthermore, the results obtained are even higher than those reported by Kulyukin et al. (2018), using the same data as the authors did. From this assessment, it can be stated that there is evidence to suggest that the data set used for this task does not have enough quality to allow for the development of proper classification models.

In regard to the elapsed time of all models, it can be concluded that the LSTM+CNN combo seems to be significantly more efficient than the LSTM approach. Since the CNN has the ability to extract relevant features in a fast manner, it allows the LSTM to have less data to process, increasing the overall efficiency around 10-15 times. Furthermore, when comparing the raw data approach with the MFCC data, it can be stated that the latter is slightly less efficient, which is expected, as the respective approach corresponds to using the same model but with an additional pre-processing step.

In sum, there is evidence that the architecture used for this task and the assessment has potential in providing high quality classification results, as long as the data set also has enough quality (which is not the case of the data set used for this task, as per the results of the assessment using the BUZZ1 data). Finally, regarding efficiency, as it was not possible to obtain a classifier with the minimum desirable accuracy, a rigorous assessment of which model is more efficient was not conducted under this task's scope (as it is not relevant which is more efficient, since none of the approaches showed good results for the target data set used).

Considering the objectives defined for this task and the respective hypothesis, it can be stated that it was not possible to obtain a model which complied with the accuracy goal, albeit the efficiency was still obtained. The reason behind this fact lies in the quality of the data set used for the task. This conclusion was obtained by using the same methodology defined in this task for a different data set, which corresponds to the same used by Kulyukin et al. (2018). This methodology was able to obtain an accuracy higher than the one reported by Kulyukin et al. (2018), suggesting that the first data set used in this task (which was collected by the respective authors with the goal of detecting health states of a bee hive) does not have the required audio quality to obtain satisfactory results. Additionally, the validation of the data also suggests that the defined methodology, which consists of using an LSTM for the bee health classification task, shows potential in obtaining results comparable to ones of state-of-the-art. On the implications of these conclusions, it can be stated that stakeholders who intend to develop a solution for the health classification through audio must collect data

in a rigorous fashion, making sure that the audio's quality is not degraded with noise (such as sustained "beeps"). Furthermore, in regard to the data set of the Open Source Beehive, as stated previously, it corresponds to a compilation of audio recordings from different individuals worldwide, meaning that the recording conditions were uncontrolled. Due to this reason, and the analysis conducted in this task, it can also be concluded that the data set of the Open Source Beehive does not meet the required standards to perform an adequate classification of a hive's health, further highlighting the need of a rigorous data collection process. Finally, despite the impact of the used data's quality on the tasks' results, due to temporal constraints, it was not possible to perform a manual recording and collection of the data to be used for this task.

6.5 Summary

In this chapter, the development of the solution to this thesis' problem was described, including the computation study conducted on each of the sub solutions' developed models. Furthermore, the general considerations were described, and each tasks' data set, pre-processing steps and evaluation algorithms were presented, with the goal of providing further insights into the development process.

In the case of the bee health classification through image, it was possible to obtain a solution that complies with all objectives defined for the task, meaning that stakeholders who are interested in deploying a similar solution can do so with the defined methodology and expect satisfactory results.

In the case of the detection of bees and Asian wasps, it was possible to obtain a solution that complied with the efficiency requirements of the objectives. In the case of the quality, it was concluded that from the different architectures and respective models evaluated, the generalization of the detection capabilities, whilst maintaining satisfactory results, is a non-trivial task. However, when using the detection models on a controlled scenario, with controlled angles and perspectives such as the images provided by a static camera pointed at a landing pad, it was possible to conclude, from a qualitative analysis, that the models obtained satisfactory results. Stakeholders interested in developing such a solution to deploy in a commercial fashion are advised to first define the angle and perspective of the images that will be used for the detection process, to ensure a satisfactory quality.

Finally, in the case of bee health classification through audio, the defined methodology potential in providing state-of-the-art results was described and showcased. However, for the particular task, the used data set yielded low audio quality, which did not make it possible to obtain satisfactory results, in regard to the quality of the models, for the bee health classification. A description of the validation of the data set's audio quality was provided, including a test using a data set described in sub section 2.5, where the methodology defined in this thesis for the given task was able to obtain accuracy than those reported by Kulyukin et al. (2018).

7 Conclusions

In this thesis dissertation, a thorough description and development of AI solutions for the automatic detection of signs of health in bee hives, with the goal of aiding the development of BHM systems was proposed. It can be stated that this thesis describes the groundwork and several frameworks for the use of AI in bee health monitoring.

The motivation, objectives and approach were first described, as to provide a context of the thesis' problem, as well as the need for the solution that is being developed. A literature review was conducted to assess all relevant advancements in several areas of interest, such as CV and SP, but also in several technical aspects like related work, important technologies, and others. The literature review also described several definitions and historic contexts, as to assure that a common ground of contextualization and definition is established for all subsequent developments of this thesis work, further enabling the understanding of the work developed.

From the value analysis conducted, it was possible to assess the potential usage and value of the thesis' solutions for the intended stakeholders, as it could be stated that the developed solution should be pursued due to the demands and needs of beekeeping and AI solutions and the lack of systems that can effectively meet these demands. From the QFD, it was possible to establish that the target clients prioritize quality of the information over the speed at which it is obtained, which guided all design choices made for the solution, including the selection of the best architectures between a set of alternative designs. Furthermore, a recommended architecture for the implementation and development of the BHMS was proposed. Regarding the validation and testing, the evaluation plan was defined in accordance to the needs and questions extracted from the master thesis' objectives and requirements. The testing plan also described the possible statistical tests to be used in the computational study of the different sub solutions developed, including definitions of the tests, as to provide further

context of their use during model comparison. Additionally, the expected quality standards of the data to be collected for the development of the classifiers were also described.

Finally, regarding the solution developed for this master thesis' problem, it can be concluded that it consists of 3 sub solutions, which correspond to the respective solutions of each tasks' problem (i.e. bee health classification through image, through audio and object detection of bees and Asian wasps).

In the case of the bee health classification through image task, the defined objectives were achieved with satisfactory results, which were better than those provided by the baseline work used for the task. From these results, it can be concluded that the methodology can be used in a real-life scenario with satisfactory results to stakeholders.

In the case of the detection of bees and Asian wasps, only the efficiency objective was achieved, with modest accuracy results. From the qualitative analysis, the tested models performed better on images regarding controlled scenarios, which suggests that the generalization of the detection of bees and Asian wasps in a wide spread of scenarios is a non-trivial and ambitious task. Furthermore, it can be concluded from the obtained results that the methodology can be used by stakeholders to detect bees and Asian wasps, as long as these models are applied to images that describe a controlled scenario, with a pre-defined static angle and perspective (e.g. one where the camera is place at the entrance of a bee hive, pointed at the landing pad).

In the case of the bee health classification through audio task, the efficiency objective was achieved, but with the data set used, it was not possible to achieve the accuracy objective. From performed tests, it was possible to conclude that the used data set exhibited low audio quality, and that the methodology defined for the task has great potential in providing state-of-the-art results. The latter was confirmed when using a different data set (BUZZ1), which was also used by Kulyukin et al. (2018) to distinguish between bee, cricket, and random noise audio clips. Using this thesis' defined methodology, for the same problem task defined by the authors, it was possible to obtain better accuracy results than those reported by Kulyukin et al. (2018). From the conclusions, it can be stated that if stakeholders are cautious of the data collection process, using the defined methodology it may be possible to achieve state-of-the-art results for the bee health classification through audio.

7.1 Contributions

This section serves the purpose of highlighting the contributions of this thesis for the context problem and to its stakeholders. Specifically, two different groups of stakeholders will be addressed – business and academia stakeholders.

In the case of business, this thesis provides a solution for the bee health classification through image, with satisfactory results in a real-life scenario. Furthermore, this thesis also provides the groundwork conducted on the development of object detection and bee health

classification through audio models, providing insights to business stakeholders of the challenges that can arise during the development of these models. From this, stakeholders can better estimate the costs of developing such models, as well as the potential best approaches, architectures, and models, including related pitfalls, which can be used in the development process.

In the case of the scientific community, researchers and academia alike, a significant contribution on the possible applications of AI & ML in aiding the preservation of bees and respective dependent ecosystems was provided, including a broad groundwork of the challenges, approaches and conditions related to the bee health classification through image and audio, and object detection of bees and Asian wasps. From the work conducted in this thesis, it is possible to determine the different pre-processing and evaluation algorithms that may be needed to meet goals, as well as the standards of quality that are required to meet objectives and the impacts of bad data in obtained results. Furthermore, the methodologies provided also serve as a baseline work that can be used to further build upon and improve the obtained results, in order to achieve better state-of-the-art results. Considering the literature review conducted, it was possible to state that the number of contributions of AI applied to bee health monitoring is quite scarce, with a low variety in contribution topics. As such, this thesis serves as a further contribution on related work, with an ambitious exploration of the different problems associated to bee health monitoring and respective potential solutions.

7.2 Limitations and Future Work

On the context of the limitations present in the solution provided by this thesis, it can be said that the models developed for the object detection task cannot achieve a satisfactory accuracy when considering a large variety of images with different perspectives and angles, suggesting that the generalization of the object detection model is a non-trivial task. Additionally, with the data set used (Inês Nolasco & Benetos, 2018) in the bee health classification through audio task, the models developed could not achieve the target accuracy due to the quality of the data. Despite the LSTM obtaining better results than the ones reported in the state-of-the-art, for the BUZZ 1 data set (a different data set from the one used for the health classification), it was not possible to provide a solid conclusion that the defined methodology will yield state-of-the-art results in bee health classification through audio. Instead it can only be concluded that, depending on the quality of the data set, there is potential in providing such results for the bee health classification task (through audio) with the defined methodology.

Future work of this thesis will consist in further improving the accuracy of the object detection and bee health classification through audio models, which will consist majorly in obtaining more data with better quality, alongside hyper-parameter tuning. Additionally, extra concepts and questions obtained during the development and computational study of this thesis should also be further explored in future work. Finally, papers concerning these improvements will

also be published, as to obtain validation and review from peers in the scientific community. During this thesis' development, the following papers were produced:

- Towards a Decision Support System for the Automatic Detection of Asian Hornets and Removal Planning, published on the IJCISIM 2020, Volume 12 (Braga & Madureira, 2020).
- Health Status Monitoring of a Bee Hive, submitted to the Agriculture, Ecosystems & Environment Journal (Veldkamp & Li, 2020).

References

A. Ramezan, C., A. Warner, T., & E. Maxwell, A. (2019). Evaluation of Sampling and Cross-Validation Tuning Strategies for Regional-Scale Machine Learning Classification. *Remote Sensing*, *11*(2), 185. https://doi.org/10.3390/rs11020185

Adrion, R., Cherniavsky, J., & Branstad, M. (1982). *Validation, Verification, and Testing of Computer Software*. https://www.researchgate.net/publication/220566096_Validation_Verification_and_Testing_of_Computer_Software

Amlathe, P. (2018). *Standard Machine Learning Techniques in Audio Beehive Monitoring: Classification of Audio Samples with Logistic Regression, K-Nearest Neighbor, Random Forest and Support Vector Machine*. 57.

Anh, H. N. (2020). *Experiencor/keras-yolo3* [Python]. https://github.com/experiencor/keras-yolo3 (Original work published 2018)

Anjos, O., & IPCB, C. B. (2020). *OFÉLIA ANJOS | PhD | Polytechnic Institute of Castelo Branco, Castelo Branco | IPCB | Department of Biological Engineering and Food*. ResearchGate. https://www.researchgate.net/profile/Ofelia_Anjos

Antony, J., McGuinness, K., Connor, N. E. O., & Moran, K. (2016). Quantifying Radiographic Knee Osteoarthritis Severity using Deep Convolutional Neural Networks. *ArXiv:1609.02469 [Cs]*. http://arxiv.org/abs/1609.02469

Ayodele, T. O. (2010). *Types of machine learning algorithms*. INTECH Open Access Publisher. http://cdn.intechweb.org/pdfs/10694.pdf

Baehr, M. (2004). *Evaluation Methodology*. https://www.webpages.uidaho.edu/ele/Scholars/Practices/Evaluating_Projects/Resources/Evaluation%20Methodology.pdf

Baldi, P. (2012). *Autoencoders, Unsupervised Learning, and Deep Architectures*. 14.

Bansal, H. (2019, October 17). *Best Languages For Machine Learning in 2020! - Becoming Human: Artificial Intelligence Magazine*. https://becominghuman.ai/best-languages-for-machine-learning-in-2020-6034732dd242

BBC Earth. (2017, August 7). *Army Ants Rampage Through The Forest | The Hunt | BBC Earth*. https://www.youtube.com/watch?v=JsfiUR0ZzLw

Beesource Beekeeping Forum. (2017). *Smart hive monitoring*. Beesource Beekeeping Forums. https://www.beesource.com/forums/showthread.php?339485-smart-hive-monitoring

Ben Reinstate Monica. (2018, May 31). *machine learning—Do more object classes increase or decrease the accuracy of object detection*. Cross Validated. https://stats.stackexchange.com/questions/348584/do-more-object-classes-increase-or-decrease-the-accuracy-of-object-detection

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2017). Recurrent Neural Network Architectures. In F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, & R. Jenssen, *Recurrent Neural Networks for Short-Term Load Forecasting* (pp. 23–29). Springer International Publishing. https://doi.org/10.1007/978-3-319-70338-1_3

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bouvrie, J. (2006). *Notes on Convolutional Neural Networks*. 8.

Braga, D., & Madureira, A. (2020). Towards a Decision Support System for the Automatic Detection of Asian Hornets and Removal Planning. *International Journal of Computer Information Systems and Industrial Management Applications*, *12*, 8.

Brown, A., & Nvidia. (2017, November 2). *Introduction to Object Detection & Image Segmentation*.

Brown, S. (2016, May 22). *Measures of Shape: Skewness and Kurtosis*. https://brownmath.com/stat/shape.htm

Brownlee, J. (2019a). *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Brownlee, J. (2019b, April 18). A Gentle Introduction to Padding and Stride for Convolutional Neural Networks. *Machine Learning Mastery*. https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/

Brownlee, J. (2019c, August 5). *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

Bruintjes, R.-J. (2019, August 26). *On Object Detection Metrics*. Medium. https://medium.com/swlh/on-object-detection-metrics-ae1e2090bd65

Budhiraja, A. (2018, March 6). *Learning Less to Learn Better—Dropout in (Deep) Machine learning*. Medium. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

C A Glasbey, & G W Horgan. (2010). Mathematical Morphology. In *IMAGE ANALYSIS FOR THE BIOLOGICAL SCIENCES*. https://www.bioss.ac.uk/people/chris/ch5.pdf

Canadian Honey Council. (2020). *Bee Facts*. https://honeycouncil.ca/bee-facts/

Cardia, L., & Andrade, J. (2020). *EINOV3—Estratégia de Inovação*.

Caruana, R. (2003). *Performance Measures*. https://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf

Carvia Tech. (2019, December). *Difference between Loss, Accuracy, Validation loss, Validation accuracy in Keras*. https://www.javacodemonk.com/difference-between-loss-accuracy-validation-loss-validation-accuracy-in-keras-ff358faa

Cass, S. (2016, July 26). *The 2016 Top Programming Languages*. IEEE Spectrum: Technology, Engineering, and Science News. http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages

Cecchi, S., Terenzi, A., Orcioni, S., Riolo, P., & Isidoro, N. (2018, May 23). *A Preliminary Study of Sounds Emitted by Honey Bees in a Beehive*.

Chang, O. (2018, May 28). *Why are LSTMs better than Elman RNNs?* https://crazyoscarchang.github.io/2018/05/28/why-are-lstms-better-than-rnns/

Chatterjee, C. C. (2019a, July 31). *Basics of the Classic CNN*. Medium. https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add

Chatterjee, C. C. (2019b, November 28). *An Approach Towards Convolutional Recurrent Neural Networks*. Medium. https://towardsdatascience.com/an-approach-towards-convolutional-recurrent-neural-networks-a2e6ce722b19

Chazette, L., Becker, M., & Szczerbicka, H. (2016). *Basic algorithms for bee hive monitoring and laser-based mite control*. 1–8. https://doi.org/10.1109/SSCI.2016.7850001

Chollet, François et al. (2015). *Home—Keras Documentation*. https://keras.io/

Christiansen, A. (2019, May 22). *Anchor Boxes—The key to quality object detection*. Medium. https://medium.com/@andersasac/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9

Cleeremans, A., Servan-Schreiber, D., & McClelland, J. L. (1989). Finite State Automata and Simple Recurrent Networks. *Neural Computation*, *1*(3), 372–381. https://doi.org/10.1162/neco.1989.1.3.372

Coloss. (2019). *Velutina – COLOSS*. https://coloss.org/task-forces/velutina/

Cooley, C., Coleman, S., B., G., & Bryan, S. (2017, September). Saliency Detection and Object Classification. *ResearchGate*. https://www.researchgate.net/publication/323548105_Saliency_Detection_and_Object_Classification

Diário de Notícias. (2020, May 7). *Alarme nos EUA com a invasão de vespas asiáticas gigantes—DN*. https://www.dn.pt/mundo/alarme-nos-eua-com-a-invasao-de-vespas-asiaticas-gigantes-12165312.html

Dobra, A. (2017). Data Mining. In *Data Mining* (pp. 2–27). https://www.cise.ufl.edu/~adobra/datamining/classif-intro.pdf

Doty, S. R. (2008). *Python Basics*. https://pdfs.semanticscholar.org/5ee9/919199306c26efb0573788e2e313908bf65a.pdf

Dyson, R. G. (2004). Strategic development and SWOT analysis at the University of Warwick. *European Journal of Operational Research*, *152*(3), 631–640. https://doi.org/10.1016/S0377-2217(03)00062-6

Eidheim, O. C. (2007). *Introduction to Mathematical Morphology*. 40.

Elman, J. L. (1990). *Finding Structure in Time*. https://crl.ucsd.edu/~elman/Papers/fsit.pdf

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118. https://doi.org/10.1038/nature21056

European Commission. (2015, January). *Final Report Summary—STEP (Status and Trends of European Pollinators) | Report Summary | STEP | FP7*. CORDIS | European Commission. https://cordis.europa.eu/result/rcn/176061_en.html

Explore. (2019a, May). *Bee Cam—Live camera inside of a honey bee hive*. https://explore.org/livecams/honey-bees/honey-bee-hive-cam

Explore. (2019b, May). *Honey Bees Landing Zone Camera—Live video of bees*. https://explore.org/livecams/honey-bees/honey-bee-landing-zone-cam

Fang, J.-T., Day, C.-T., & Chang, P.-C. (2016, November 13). *Deep feature learning for cover song identification*. https://link.springer.com/article/10.1007/s11042-016-4107-6

FarmMeetsTable. (2016). *Are Honey Bees Really Dying out? – Farm Meets Table*. https://www.farmmeetstable.com:443/en/protecting-our-environment/2018/are-honey-bees-really-dying-out

Ferrari, P. (2020). *Pierluigiferrari/ssd_keras* [Python]. https://github.com/pierluigiferrari/ssd_keras (Original work published 2017)

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Florea, M. (2013, September). *Automatic detection of honeybees in a hive*. http://uu.diva-portal.org/smash/get/diva2:645634/FULLTEXT01.pdf

FNAP. (2020). *FNAP – Federação Nacional dos Apicultores de Portugal*. http://fnap.pt/

Food and Agriculture Organization of the United Nations. (2018, May 20). *FAO - News Article: Bees must be protected for the future of our food*. http://www.fao.org/news/story/en/item/1132329/icode/

Forsvarets Forskninginstitutt. (2018). *Introduction to TensorFlow*. https://www.uio.no/studier/emner/matnat/its/TEK5040/h18/lecture-slides/introduction-to-tensorflow.pdf

Freeman, P. (1976). The Central Role of Design in Software Engineering. In A. I. Wasserman & P. Freeman (Eds.), *Software Engineering Education: Needs and Objectives. Proceedings of an Interface Workshop* (pp. 116–119). Springer. https://doi.org/10.1007/978-1-4612-9898-4_20

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202. https://doi.org/10.1007/BF00344251

Gama, J., Ponce de Leon Carvalho, A., Oliveira, M., Lorena, A. C., & Faceli, K. (2015). *Extração de Conhecimento de Dados*. http://www.silabo.pt/Conteudos/8117_PDF.pdf

Girshick, R. (2015). Fast R-CNN. *ArXiv:1504.08083 [Cs]*. http://arxiv.org/abs/1504.08083

Godoy, D. (2019, February 7). *Understanding binary cross-entropy / log loss: A visual explanation*. Medium. https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

Google. (2020). *TensorFlow*. https://www.tensorflow.org/

Google Developers. (2020a). *Classification: Precision and Recall | Machine Learning Crash Course*. Google Developers. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

Google Developers. (2020b). *Classification: True vs. False and Positive vs. Negative*. https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

Google Trends. (2020). *Google Trends*. https://trends.google.com/trends/?geo=US

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. C., Mega, J. L., & Webster, D. R. (2016). Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, *316*(22), 2402. https://doi.org/10.1001/jama.2016.17216

Gurney, K. (2004). *Introduction to Neural Networks*. Taylor & Francis.

Hale, J. (2018). *Deep Learning Framework Power Scores 2018—Towards Data Science*. https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a

Hartford, J., Wright, J. R., & Leyton-Brown, K. (2016). *Deep Learning for Predicting Human Strategic Behavior*. 9.

Hassan, A. (2012). *The Value Proposition Concept in Marketing: How Customers Perceive the Value Delivered by Firms– A Study of Customer Perspectives on Supermarkets in Southampton in the United Kingdom*. https://doi.org/10.5539/ijms.v4n3p68

Hauser, J., Griffin, A., Katz, G., & Klein, R. L. (2010). Quality Function Deployment (QFD). In *Wiley International Encyclopedia of Marketing*. https://www.researchgate.net/publication/228014887_Quality_Function_Deployment_QFD

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *ArXiv:1703.06870 [Cs]*. http://arxiv.org/abs/1703.06870

HealthyWithHoney. (2018, July). *What is artificial honey or fake honey? What we have in our food!Discover the benefits of honey*. https://healthywithhoney.com/what-is-artificial-honey-or-fake-honey-what-we-have-in-our-food/

Hinton, G. E. (2007a, March 25). *Boltzmann Machines*. https://www.cs.toronto.edu/~hinton/csc321/readings/boltz321.pdf

Hinton, G. E. (2007b). Learning multiple layers of representation. *Trends in Cognitive Sciences*, *11*(10), 428–434. https://doi.org/10.1016/j.tics.2007.09.004

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, *18*(7), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

Hlaváč, V. (2018, December 12). *Mathematical morphology*. http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/71-3MatMorpholBinEn.pdf

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. https://www.bioinf.jku.at/publications/older/2604.pdf

Hosseini, H., Xiao, B., Jaiswal, M., & Poovendran, R. (2017). On the Limitation of Convolutional Neural Networks in Recognizing Negative Images. *ArXiv:1703.06857 [Cs, Stat]*. http://arxiv.org/abs/1703.06857

Hui, J. (2019, April 3). *MAP (mean Average Precision) for Object Detection*. Medium. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

Hung, K.-L. J., Kingston, J. M., Albrecht, M., Holway, D. A., & Kohn, J. R. (2018). The worldwide importance of honey bees as pollinators in natural habitats. *Proc. R. Soc. B*, *285*(1870), 20172140. https://doi.org/10.1098/rspb.2017.2140

IoBee. (2020). *IoBee | IoT application*. http://io-bee.eu/

ITV News. (2016, September 20). *Asian Hornets seen in Britain for first time*. https://www.youtube.com/watch?v=Jkl5L9gcu5Y

Jacques, A., Laurent, M., Consortium, E., Ribière-Chabert, M., Saussac, M., Bougeard, S., Budge, G. E., Hendrikx, P., & Chauzat, M.-P. (2017). A pan-European epidemiological study reveals honey bee colony survival depends on beekeeper education and disease control. *PLOS ONE*, *12*(3), e0172591. https://doi.org/10.1371/journal.pone.0172591

Jim. (2020). *Open Broadcaster Software | OBS*. https://obsproject.com/

Jordan, M. I. (1986). *Attractor dynamics and parallelism in a connectionist sequential machine*.

Kabir, S. M. S. (2016, July). *Formulating and Testing Hypothesis*. https://www.researchgate.net/publication/325846748_FORMULATING_AND_TESTING_HYPOTHESIS

Kalchbrenner, N., & Blunsom, P. (2013). *Recurrent Convolutional Neural Networks for Discourse Compositionality*. 8.

Keeling, M., Franklin, D., Datta, S., Brown, M., & Budge, G. (2017). *Predicting the spread of the Asian hornet (Vespa velutina) following its incursion into Great Britain*. https://www.nature.com/articles/s41598-017-06212-0

Kenneth, A. C. (2011). *Quality Function Deployment*. https://www.ieee.li/tmc/quality_function_deployment.pdf

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2019). *A Survey of the Recent Architectures of Deep Convolutional Neural Networks*. 62.

Kim, H.-E., & Hwang, S. (2016). Deconvolutional Feature Stacking for Weakly-Supervised Semantic Segmentation. *ArXiv:1602.04984 [Cs]*. http://arxiv.org/abs/1602.04984

Kim, L. (2015, June 1). *10 Most Popular Programming Languages Today*. Inc.Com. https://www.inc.com/larry-kim/10-most-popular-programming-languages-today.html

Kim, T. K. (2015). T test as a parametric statistic. *Korean Journal of Anesthesiology*. https://doi.org/10.4097/kjae.2015.68.6.540

Koen, P. A., Ajamian, G. M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., Johnson, A., Puri, P., & Seibert, R. (2002). Effective Methods, Tools, and Techniques. *The PDMA ToolBook for New Product Development*, 32.

Kotler, P., & Keller, K. L. (2012). *Marketing Management, 14th Edition* (14th ed.). Pearson.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–

1105). Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Kulyukin, V., Mukherjee, S., & Amlathe, P. (2018, September). *Toward Audio Beehive Monitoring: Deep Learning vs. Standard Machine Learning in Classifying Beehive Audio Samples*. ResearchGate. http://dx.doi.org/10.3390/app8091573

Laerd Statistics. (2018a). *Kruskal-Wallis H Test in SPSS Statistics*. https://statistics.laerd.com/spss-tutorials/kruskal-wallis-h-test-using-spss-statistics.php

Laerd Statistics. (2018b). *Mann-Whitney U Test in SPSS Statistics*. https://statistics.laerd.com/spss-tutorials/mann-whitney-u-test-using-spss-statistics.php

Laerd Statistics. (2018c). *Sign Test in SPSS Statistics—Procedure, output and interpretation of output using a relevant example | Laerd Statistics*. https://statistics.laerd.com/spss-tutorials/sign-test-using-spss-statistics.php

Laerd Statistics. (2018d). *Testing for Normality using SPSS Statistics when you have only one independent variable.* https://statistics.laerd.com/spss-tutorials/testing-for-normality-using-spss-statistics.php

Laerd Statistics. (2018e). *Understanding the different types of variable in statistics | Laerd Statistics*. https://statistics.laerd.com/statistical-guides/types-of-variable.php

Laerd Statistics. (2018f). *Wilcoxon Signed Rank Test in SPSS Statistics—Procedure, output and interpretation of output using a relevant example.* https://statistics.laerd.com/spss-tutorials/wilcoxon-signed-rank-test-using-spss-statistics.php

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lexico. (2020). *Hypothesis*. https://www.lexico.com/definition/hypothesis

Leza, M., Herrera, C., Marques, A., Roca, P., Sastre-Serra, J., & Pons, D. G. (2019). *The impact of the invasive species Vespa velutina on honeybees: A new approach based on oxidative stress—ScienceDirect*. *689*. https://www.sciencedirect.com/science/article/pii/S0048969719330839

Librosa. (2018). *LibROSA — librosa 0.6.2 documentation*. https://librosa.org/librosa/0.6.2/index.html

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *ArXiv:1405.0312 [Cs]*. http://arxiv.org/abs/1405.0312

Lindgreen, A., & Wynstra, F. (2005). Value in Business Markets: What do We Know? Where are We Going? *Industrial Marketing Management*, *34*(7). https://doi.org/10.1016/j.indmarman.2005.01.001

Lipton, J., & Priya, S. (2017, September). *What are some of the limitations or drawbacks of Convolutional Neural Networks? - Quora*. https://www.quora.com/What-are-some-of-the-limitations-or-drawbacks-of-Convolutional-Neural-Networks

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88. https://doi.org/10.1016/j.media.2017.07.005

Liu, J., Pan, Y., Li, M., Chen, Z., Tang, L., Lu, C., & Wang, J. (2018). Applications of deep learning to MRI images: A survey. *Big Data Mining and Analytics*, *1*(1), 1–18. https://doi.org/10.26599/BDMA.2018.9020001

Liu, S., Mocanu, D. C., & Pechenizkiy, M. (2019). Intrinsically Sparse Long Short-Term Memory Networks. *ArXiv:1901.09208 [Cs]*. http://arxiv.org/abs/1901.09208

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *ArXiv:1512.02325 [Cs]*, *9905*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Liu, X., Deng, Z., & Yang, Y. (2018). Recent progress in semantic image segmentation. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-018-9641-3

Lo, S. B., Lou, S. A., Lin, J. S., Freedman, M. T., Chien, M. V., & Mun, S. K. (1995). Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE Transactions on Medical Imaging*, *14*(4), 711–718. https://doi.org/10.1109/42.476112

Loucopoulos, V., & Karakostas, V. (2001). System requirements engineering. *Journal of Computer Science & Technology*, *1, no. 5*. http://sedici.unlp.edu.ar/handle/10915/9425

Louis. (2017). *Precision-Recall versus Accuracy and the Role of Large Data Sets*.

Machine Learning Notebook. (2017, April 7). *Convolutional Neural Networks—Basics · Machine Learning Notebook*. https://mlnotebook.github.io/post/CNN1/

Manal El Aidouni. (2019, October 5). *Evaluating Object Detection Models: Guide to Performance Metrics*. manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html

Mao, H., Yang, X., & Dally, W. J. (2019). A Delay Metric for Video Object Detection: What Average Precision Fails to Tell. *ArXiv:1908.06368 [Cs]*. http://arxiv.org/abs/1908.06368

Mathworks. (2020). *Types of Morphological Operations—MATLAB & Simulink*. https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html

McClenaghan, B., Schlaf, M., Geddes, M., Mazza, J., Pitman, G., McCallum, K., Rawluk, S., Hand, K., & Otis, G. (2018). Behavioral responses of honey bees, Apis cerana and Apis

mellifera , to Vespa mandarinia marking and alarm pheromones. *Journal of Apicultural Research*, 1–8. https://doi.org/10.1080/00218839.2018.1494917

Mcfarlane, D. (2013). *The Strategic Importance of Customer Value*.

McNeese, B. (2016, February). *Are the Skewness and Kurtosis Useful Statistics?* BPI Consulting. https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics

Meikle, W. G., & Holst, N. (2015). Application of continuous monitoring of honeybee colonies. *Apidologie*, *46*(1), 10–22. https://doi.org/10.1007/s13592-014-0298-x

Messner, W. (2013). *Making the Compelling Business Case: Decision-Making Techniques for Successful Business Growth*. Springer.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. 298.

Milius, S. (2018, February 13). *The mystery of vanishing honeybees is still not definitively solved*. Science News. https://www.sciencenews.org/article/mystery-vanishing-honeybees-still-not-definitively-solved

Monceau, K., Bonnard, O., & Thiery, D. (2014). Vespa velutina: A new invasive predator of honeybees in Europe. *Journal of Pest Science*, *87*. https://doi.org/10.1007/s10340-013-0537-3

Monceau, K., Thiery, D., & Bonnard, O. (2014). *Vespa velutina: A new invasive predator of honeybees in Europe*. https://www.researchgate.net/publication/258769112_Vespa_velutina_A_new_invasive_predator_of_honeybees_in_Europe

Nascimento Jr, C. L. (1994). Artificial neural networks in control and optimization. *Doctor Thesis. University of Manchester. Manchester*.

ftp://161.24.19.221/ele/cairo/My_Publications/1994/CairoNascimento_PhD_Thesis_1994.pdf

Nath, A. (2016, August). *(PDF) Colony Collapse Disorder*. ResearchGate. https://www.researchgate.net/publication/307989119_Colony_Collapse_Disorder

Nebeker, F. (2000). *Fifty Years of Signal Processing*. https://signalprocessingsociety.org/uploads/history/history.pdf

NeuralDesigner. (2020). *Neural Networks Tutorial | Neural Designer*. https://www.neuraldesigner.com/learning/neural-networks-tutorial

Ng, A. Y., Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., & Prochnow, B. (2013). Building high-level features using large scale unsupervised learning. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8595–8598. https://doi.org/10.1109/ICASSP.2013.6639343

Nicholson, C. (2019). *Evaluation Metrics for Machine Learning—Accuracy, Precision, Recall, and F1 Defined*. Pathmind. http://pathmind.com/wiki/accuracy-precision-recall-f1

Nicola, S. (2020). *Análise de Valor*.

Nolasco, Inês, & Benetos, E. (2018). *To bee or not to bee: An annotated dataset for beehive sound recognition* [Data set]. Zenodo. https://doi.org/10.5281/zenodo.1321278

Nolasco, Ines, & Benetos, E. (2018). *To bee or not to bee: Investigating machine learning approaches for beehive sound recognition*.

NPTEL. (2017). *Digital Signal Processing*. http://nptel.ac.in/courses/Webcourse-contents/IIT-KANPUR/Digi_Sign_Pro/pdf/ch1.pdf

OECD. (2005). *OSLO Manual—Third Edition*. https://rio.jrc.ec.europa.eu/en/library/oslo-manual-third-edition

OSBeeHives. (2020). *OSBeehives | BuzzBox Hive Health Monitor & Beekeeping App*. https://www.osbeehives.com

Oxford Learners Dictionaries. (2020a). *innovation noun—Definition, pictures, pronunciation and usage notes*. https://www.oxfordlearnersdictionaries.com/definition/english/innovation

Oxford Learners Dictionaries. (2020b). *Value—Definition, pictures, pronunciation and usage notes*. https://www.oxfordlearnersdictionaries.com/definition/english/value_1

Palik, E. D. (1998). *R Language Definition* (Vol. 3). Academic press. http://books.google.com/books?hl=en&lr=&id=nxoqxyoHfbIC&oi=fnd&pg=PR13&dq=%22Constants+.%22+&ots=EE4ZHDv_ei&sig=aIaa-xjQoMjb8lh9CZbuRFH9Z08

Pearlmutter. (1989). Learning state space trajectories in recurrent neural networks. *International Joint Conference on Neural Networks*, 365–372 vol.2. https://doi.org/10.1109/IJCNN.1989.118724

Pei, K., Cao, Y., Yang, J., & Jana, S. (2017). DeepXplore: Automated Whitebox Testing of Deep Learning Systems. *Proceedings of the 26th Symposium on Operating Systems Principles - SOSP '17*, 1–18. https://doi.org/10.1145/3132747.3132785

Pinheiro, P. O., & Collobert, R. (2014). *Recurrent Convolutional Neural Networks for Scene Labeling*. 9.

Pollenity. (2020, January 4). *Modern Beekeeping Products & Equipment | Pollenity*. https://pollenity.com/shop/

Powers, D. M. W. (2020). *What the F-measure doesn't measure…*. 19.

Princeton University, Stanford Vision Lab, & Stanford University. (2019). *ImageNet*. http://www.image-net.org/

Pukhov, D. (2018, October). *Honey Bee health detection with CNN*. https://kaggle.com/dmitrypukhov/honey-bee-health-detection-with-cnn

Puolivali, T. (2013, February 20). *A Short Introduction to Signal Processing*. http://www.mit.jyu.fi/palvelut/sovellusprojektit/luennot/signaalinkasittely2013k.pdf

Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S., & Sainath, T. (2019). Deep Learning for Audio Signal Processing. *IEEE Journal of Selected Topics in Signal Processing*, *13*(2), 206–219. https://doi.org/10.1109/JSTSP.2019.2908700

Python Software Foundation. (2020). *Welcome to Python.org*. https://www.python.org/

PyTorch. (2020). *PyTorch*. https://pytorch.org/

Quality-One. (2020). *QFD | Quality Function Deployment*. https://quality-one.com/qfd/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv:1506.01497 [Cs]*. http://arxiv.org/abs/1506.01497

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Ryouta, S. (2014, December 20). *Giant hornets attacked beehive*. https://www.youtube.com/watch?v=ZdtfWh9aO20

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4580–4584. https://doi.org/10.1109/ICASSP.2015.7178838

Sak, H., Senior, A., & Beaufays, F. (2015). *Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling*. 5.

Salakhutdinov, R., & Hinton, G. E. (2009). *Deep Boltzmann Machines*. http://www.utstat.toronto.edu/~rsalakhu/papers/dbm.pdf

Santos, L. (2020). *Object Localization and Detection · Artificial Inteligence*. https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html

Sasaki, Y. (2007). The truth of the F-measure. *Teach Tutor Mater*.

Sawyer, S. (2009). Analysis of Variance: The Fundamental Concepts. *Journal of Manual & Manipulative Therapy*, *17*, 27E-38E. https://doi.org/10.1179/jmt.2009.17.2.27E

Schapire, R. (2008). *Theoretical Machine Learning*. https://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf

Schneider, J. (1997). *Cross Validation*. https://www.cs.cmu.edu/~schneide/tut5/node42.html

Schurischuster, S., Zambanini, S., & Kampel, M. (2016). *Sensor Study for Monitoring Varroa Mites on Honey Bees (Apis mellifera)*.

SciPy. (2020). *SciPy.org*. https://www.scipy.org/

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Shapiro, L., & Stockman, G. (2000). *CSE576: Computer Vision—Chapter 10*. https://courses.cs.washington.edu/courses/cse576/book/ch10.pdf

Shung, K. P. (2020, January 22). *Accuracy, Precision, Recall or F1?* Medium. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Siegle, D. (2015, May 22). *Confidence Intervals and Levels | Educational Research Basics by Del Siegle*. https://researchbasics.education.uconn.edu/confidence-intervals-and-levels/

Silhavy, R., Silhavy, P., & Prokopova, Z. (2011, January). *Requirements gathering methods in system engineering*. ResearchGate.

https://www.researchgate.net/publication/228400297_Requirements_gathering_methods_in_system_engineering

Silva, N. (2015). *Introdução a Arquitetura de Software*.

Siqueira, F. (2020, April 12). *Europa teme invasão de vespas gigantes e assassinas vindas da Ásia*. R7.com. https://noticias.r7.com/hora-7/fotos/europa-teme-invasao-de-vespas-gigantes-e-assassinas-vindas-da-asia-12042020

Sirali, R., & Cinbirtoğlu, Ş. (2018, August). *The Importance Of Honey Bees In The Pollination And Plant Production*. ResearchGate. https://www.researchgate.net/publication/327040142_THE_IMPORTANCE_OF_HONEY_BEES_IN_THE_POLLINATION_AND_PLANT_PRODUCTION

SmartPLS. (2020). *How to Interpret Excess Kurtosis and Skewness*. https://www.smartpls.com/documentation/functionalities/excess-kurtosis-and-skewness

Sood, A. (2017). *Long Short-Term Memory Networks*. http://pages.cs.wisc.edu/~shavlik/cs638/lectureNotes/Long%20Short-Term%20Memory%20Networks.pdf

Špacek, M., & Vacík, E. (2016). Company Value Creation through Effective Innovation Process Management. *Journal of Innovation Management*, *4*(3), 65–78. https://doi.org/10.24840/2183-0606_004.003_0006

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. 30.

Srivastava, T. (2019, August 6). 11 Important Model Evaluation Error Metrics Everyone should know. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

Stephanie. (2014, November 19). *Shapiro-Wilk Test: What it is and How to Run it*. Statistics How To. https://www.statisticshowto.com/shapiro-wilk-test/

Stephanie. (2015, September 19). *Wilcoxon Signed Rank Test: Definition, How to Run*. Statistics How To. https://www.statisticshowto.com/wilcoxon-signed-rank-test/

Stephanie. (2016, February 25). *Kruskal Wallis H Test: Definition, Examples & Assumptions*. Statistics How To. https://www.statisticshowto.com/kruskal-wallis/

Strategyzer. (2016). *Figure 2. The Business Model Canvas and the Value Proposition Canvas....* ResearchGate. https://www.researchgate.net/figure/The-Business-Model-Canvas-and-the-Value-Proposition-Canvas-Source-Osterwalder-and_fig3_291171892

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *NIPS*, 9.

The Guardian. (2018). *Robotic bees could pollinate plants in case of insect apocalypse*. https://www.theguardian.com/environment/2018/oct/09/robotic-bees-could-pollinate-plants-in-case-of-insect-apocalypse

The Pascal VOC Project. (2020). *The PASCAL Visual Object Classes Homepage*. http://host.robots.ox.ac.uk/pascal/VOC/

The Portugal News. (2019, December 20). *Low-cost smart hive development*. https://www.theportugalnews.com/news/low-cost-smart-hive-development/52420

The R Foundation. (2020). *R: The R Project for Statistical Computing*. https://www.r-project.org/

The Warren Johnson Society. (2020). *Intro to Value Proposition Canvas*. http://wsjsociety.com/wp-content/uploads/2016/01/Intro-to-Value-Proposition-Canvas.pdf

Tiwari, A. (2018). A Deep Learning Approach to Recognizing Bees in Video Analysis of Bee Traffic. *All Graduate Theses and Dissertations*. https://digitalcommons.usu.edu/etd/7076

TutorialsPoint. (2019). *PyTorch Tutorial*. https://www.tutorialspoint.com/pytorch/pytorch_tutorial.pdf

Underwood, R. M., & vanEngelsdorp, D. (2008). *Colony Collapse Disorder: Have We Seen This Before?* 8.

University of Kentucky. (2020). *Basic Beekeeping Operations*. https://www.uky.edu/Ag/Entomology/ythfacts/4h/beekeep/basbeop.htm

Utah State University. (2019). *BeePi_Audio_Classification*. https://usu.app.box.com/v/BeePiAudioData

Vafeiadis, T., Diamantaras, K., Chatzisavvas, K., & Sarigiannidis, G. (2015, June). *A Comparison of Machine Learning Techniques for Customer Churn Prediction*. ResearchGate. http://dx.doi.org/10.1016/j.simpat.2015.03.003

vanEngelsdorp, D., Traynor, K. S., Andree, M., Lichtenberg, E. M., Chen, Y., Saegerman, C., & Cox-Foster, D. L. (2017). Colony Collapse Disorder (CCD) and bee age impact honey bee pathophysiology. *PLOS ONE*, *12*(7), e0179535. https://doi.org/10.1371/journal.pone.0179535

Veldkamp, T. A., & Li, Y. (2020). *Agriculture, Ecosystems & Environment*. https://www.journals.elsevier.com/agriculture-ecosystems-and-environment

VOA News. (2019, March 1). *Flying Drones Deployed Against Nests of Asian Hornets*. https://www.youtube.com/watch?v=sRMKlR8-_VY

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, *2018*. https://doi.org/10.1155/2018/7068349

Waleed Abdulla. (2020). *Matterport/Mask_RCNN* [Python]. Matterport, Inc. https://github.com/matterport/Mask_RCNN (Original work published 2017)

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). CNN-RNN: A Unified Framework for Multi-label Image Classification. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2285–2294. https://doi.org/10.1109/CVPR.2016.251

Wood, T. (2019, May 17). *Softmax Function*. DeepAI. https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

Woodall, T. (2003). Conceptualising 'value for the customer': An attributional, structural and dispositional analysis. *Academy of Marketing Science Review*, *2003*(12). http://www.amsreview.org/articles/woodall12-2003.pdf

Wu, J. (2019). *Convolutional neural networks*. 35.

Yang, J. (2018a). *The BeeImage Dataset: Annotated Honey Bee Images*. https://kaggle.com/jenny18/honey-bee-annotated-images

Yang, J. (2018b). *The BeeImage Dataset: Annotated Honey Bee Images*. https://kaggle.com/jenny18/honey-bee-annotated-images

Yang, T., Zhang, X., Li, Z., Zhang, W., & Sun, J. (2018). MetaAnchor: Learning to Detect Objects with Customized Anchors. *ArXiv:1807.00980 [Cs]*. http://arxiv.org/abs/1807.00980

Zeng, N. (2018, December 16). *An Introduction to Evaluation Metrics for Object Detection*. NickZeng|曾广宇. https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/

Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2019). Machine Learning Testing: Survey, Landscapes and Horizons. *ArXiv:1906.10742 [Cs, Stat]*. http://arxiv.org/abs/1906.10742

Zhang, X., Chen, F., & Huang, R. (2018). A Combination of RNN and CNN for Attention-based Relation Classification—ScienceDirect. *Elsevier*, *131*, 911–917.

Zhao, Z.-Q., Zheng, P., Xu, S., & Wu, X. (2019). Object Detection with Deep Learning: A Review. *ArXiv:1807.05511 [Cs]*. http://arxiv.org/abs/1807.05511

Annex A – Qualitative Analysis Images

The qualitative analysis was based on video footage generated using the Mask-RCNN, SSD and YOLO models' prediction masks. As each video has a significant number of frames, this annex contains links to some of the videos analysed, per object detection model, used to perform the qualitative analysis.

The videos can be download from the following OneDrive link:

https://myisepipp-my.sharepoint.com/:f:/r/personal/1140499_isep_ipp_pt/Documents/Thesis%20Videos?csf=1&web=1&e=lMIR7c