
# Vacancy State Detector Oriented to Convolutional Neural Network, Background Subtraction and Embedded Systems

**Isabelle de Moura Corrêa - a39509**

Thesis presented to the School of Technology and Management in the scope of the Master in Information Systems.

Supervisors:

Prof. Dr. Pedro João Soares Rodrigues

Prof. Dr. Erikson Freitas de Morais

This document does not include the suggestions made by the board.

Braganca

2018-2019


# Vacancy State Detector Oriented to Convolutional Neural Network, Background Subtraction and Embedded Systems

**Isabelle de Moura Corrêa - a39509**

Thesis presented to the School of Technology and Management in the scope of the Master in Information Systems.

Supervisors:

Prof. Dr. Pedro João Soares Rodrigues

Prof. Dr. Erikson Freitas de Morais

This document does not include the suggestions made by the board.

Braganca

2018-2019

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Diretoria de Pesquisa e Pós-Graduação

# TERMO DE APROVAÇÃO

VACANCY STATE DETECTOR ORIENTED TO CONVOLUTIONAL NEURAL NETWORK, BACKGROUND SUBTRACTION AND EMBEDDED SYSTEMS

por

ISABELLE DE MOURA CORRÊA

Esta Dissertação foi apresentada em 11 de dezembro de 2019 como requisito parcial para a obtenção do título de Mestre em Sistemas de Informação. A candidata foi arguida pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

| | |
|---|---|
| _____<br>Pedro João Soares Rodrigues<br>Prof. Orientador IPB | _____<br>José Eduardo Moreira Fernandes<br>Membro titular |
| _____<br>Erikson Freitas de Morais<br>Prof. Orientador UTFPR | _____<br>Rui Pedro Sanches de Castro Lopes<br>Membro titular |
| _____<br>Geraldo Ranthum<br>Responsável pelos Trabalhos<br>de Conclusão de Curso | _____<br>Mauren Louise Sguario<br>Coordenador do Curso<br>UTFPR - Campus Ponta Grossa |

# Dedication

To my parents, Neiva Romana de Moura and José Carlos Rocha Corrêa.

# Acknowledgment

First of all, God allowed all of this to happen throughout my life, and not only in these years as a college student but at all times, he is the greatest teacher anyone can know.

To my parents, Neiva and José, for love, encouragement, and unconditional support. Thank you all my family, who, in the moments of my absence dedicated to higher education, have always understood that the future is made from this constant dedication to the present.

To the Polytechnic Institute of Braganca and the Federal Technological University of Paraná, its faculty, management, and administration provided the window today with a glimpse of the upper horizon, and its trust in the merit and ethics present here. In particular to Professors Pedro João Soares Rodrigues and Erikson Freitas de Morais, for all their support, dedication, patience, and knowledge transfer.

In particular, I thank all my friends for their support all these years and for being my family in Ponta Grossa and Braganca, always available to help me at all possible times.

# Abstract

Much has been discussed recently related to population ascension, the reasons for this event, and, in particular, the aspects of society affected. Over the years, the city governments realized a higher level of growth, mainly in terms of urban scale, technology, and individuals numbers. It comprises improvements and investments in their structure and policies, motivated by improving conditions in population live quality and reduce environmental, energy, fuel, time, and money resources, besides population living costs, including the increasing demand for parking structures accessible to the general or private-public, and a waste of substantial daily time and fuel, disturbing the population routinely. Therefore, one way to achieve that challenge is focused on reducing energy, money, and time costs to travel to work or travel to another substantial location. That work presents a robust, and low computational power Smart Parking system adaptive to several environments changes to detect and report vacancy states in a parking space oriented to Deep Learning, and Embedded Systems. This project consists of determining the parking vacancy status through statistical and image processing methods, creates a robust image data set, and the Convolutional Neural Network model focused on predict three final classes. In order to save computational power, this approach uses the Background Subtraction based on the Mixture of Gaussian method, only updating parking space status, in which large levels of motion are detected. The proposed model presents 94 percent of precision at the designed domain.

**Keywords:** Smart Parking, Deep Learning, Background Subtraction.

# Resumo

Muito se discutiu recentemente sobre a ascensão populacional, as razões deste evento e, em particular, os aspectos da sociedade afetados. Ao longo dos anos, os governos perceberam um grande nível de crescimento, principalmente em termos de escala urbana, tecnologia e número de indivíduos. Este fato deve-se a melhorias e investimentos na estrutura urbana e políticas motivados por melhorar as condições de qualidade de vida da população e reduzir a utilização de recursos ambientais, energéticos, combustíveis, temporais e monetários, além dos custos de vida da população, incluindo a crescente demanda por estruturas de estacionamento acessíveis ao público em geral ou público-privado. Portanto, uma maneira de alcançar esse desafio é manter a atenção na redução de custos de energia, dinheiro e tempo para viajar para o trabalho ou para outro local substancial. Esse trabalho apresenta um sistema robusto de Smart Parking, com baixo consumo computacional, adaptável a diversas mudanças no ambiente observado para detectar e relatar os estados das vagas de estacionamento, orientado por Deep Learning e Embedded Systems. Este projeto consiste em determinar o status da vaga de estacionamento por meio de métodos estatísticos e de processamento de imagem, criando um conjunto robusto de dados e um modelo de Rede Neuronal Convolucional com foco na previsão de três classes finais. A fim de reduzir consumo computacional, essa abordagem usa o método de Background Subtraction, somente atualizando o status do espaço de estacionamento em que grandes níveis de movimento são detectados. O modelo proposto apresenta 94 porcento da precisão no domínio projetado.

**Palavras-chave:** Smart Parking, Deep Learning, Background Subtraction.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Chapter 1 briefly presents the introductory information for this research, starting with a the problem description, its general and specific objectives, and the document's structure.

## 1.1 Problem Description

Over the years, the cities have continuous and decentralized population growth, due to improvements in some management areas, such as the economy, governance model, quality of life, city structure, mobility structure, and several other areas. On account of this growth, the traditional city model has to be renewed, creating another definition known as Smart City [1]. That concept represents a city that has urban development based on technological advancements to solve short and long term issues, mainly reducing environmental and living costs.

Therefore, population growth imposing extra time and resources costs in urban activities, such as mobility and transportation routines. What processes should provide accessibility and efficiency in several daily activities, reducing time costs to travel to work, school, entertainment places, or other substantial locations, it allows improvements in life quality [2]. In this way, Smart Cities treated mobility operations through Smart Mobility systems.

Mobility issues it is found in almost all the big cities, and according to researchers, in New York, in the year 2014, nearly 40 percent of the cost time in traffic situations are caused by the search of vacancy [3]. Comprising the mobility processes importance and thinking in developing technologies turned to reduce the city pollution levels, saving traffic time and resources, the Smart Parking concept comes to improve the parking spaces management, providing automatic, safety, and fast arrivals and departures of vehicles [4].

Therefore, Smart Parking should provide convenience and speed solutions in search of parking spaces, creating a suitable environment for the city needs [2]. That concept is directly evolving into the Internet of Things (IoT), where devices will communicate and interact with each other through ubiquitous computing [5].

Thus, Smart Parking solutions should be allowing the main parking spaces management processes, because those environments are dynamic and non-controlled, such as the vacancy allocation, prices, or infrastructure. It is common to find drivers that are not satisfied by the management of the current space, due to that kind of solution provide expensive or slow processes [6].

The research proposes several types of systems using the most exciting technologies, such as Multi-Agents Systems (MAS), Computer Vision Systems (CVS), or other variations [6]. Each one of these technologies must provide a different solution, as MAS deals with the sensors position, vacancy structure, and managing using automated agents, in that case, CVS treated the visual solutions, such as detecting and allocate vacancies through image sequences.

Thinking in providing visual solutions without the installation and maintenance of sensors, the Computer Vision systems are the focus in represents the real scenarios through the video analysis. It is due to the capacity of this kind of system to process dynamic and real-time information, such as environments with much-moving objects, like great avenues, high stores, or great parking spaces [7].

But systems capable of detecting the interesting object should be adaptive to the most several environmental conditions, dealing with illumination, shadow, and object status (insertion, removing, and overlapping) changes in real-time [8]. That process is not

trivial, and to consider all of these conditions, it is worth use Background Subtraction, a process that segment the moving objects to the stable objects.

From this process, it is possible applying Deep Learning methods to propose robust technologies to detect interesting objects with more accuracy [7]. Moreover, Computer Vision systems also integrate Artificial Intelligence to make decisions, such as choosing the closest free vacancy in a parking space, saving up-time, and generate environment reports about the scenario analysis [9].

The present work has as the main objective to develop a robust Smart Parking visual-based system adaptive to several environment changes, that detect vacancy status through the junction of Background Subtraction and Deep Learning methods, focused on ensure lower computational power costs, and suitable for Embedded Systems.

## 1.2 Objectives

The main objective of this work is to develop a robust Smart Parking system adaptive to several environments changes to detect and report vacancy status in a parking space through Computer Vision and Deep Learning methods. In this context, this work presents a solution to identify vacancy spots status, recognizing it's spaces, and classifying in "Occupied Spot" or "Available Spot" reporting the result to the final user. This classification intends to identify positive and negative objects; positive labels encompass cars and motorcycle vehicles in several positions; it classifies negative ratings as persons, animals, bicycles and any other label, making parking management service as automatic as possible.

As a study case, the Polytechnic Institute of Braganca made available record video information about parking located at ESTIG; this space provides an challenge in terms of viewing: the vacancies delimitation. Data provided by IPB is from the year 2019 in sunny weather. This work proposes solutions for these specific objectives:

- Get a trustworthy Images Data Set for Vehicle Recognition;

- Develop a robust Vehicle Detector Model using Convolutional Neural Networks;
- Improve this Vehicle Detector using the Mixture of Gaussian as Background Subtraction method to Motion Detection;
- Study a robust Vacancy State Detector using Selective Search and Harris Corner to define vacancies position automatically;
- Develop a solution for Parking Status Detection;
- Make this solution stable to work out in Embedded Systems; and
- Report these results to the user.

To successfully conclude this proposals, it is directives proved to be the most important activities and challenges: research Deep Learning, Computer Vision and Background subtraction methods; raise the main requirements to implement the vacancy detector system, such as implementation architecture, programming language, and environment structure; integration of the developed methods and implementation in the chosen architecture; applying the method in the test scenarios and verifying these results; and improve the method based on this extracted results.

## 1.3 Document's Structure

The present work it is organized as follows: Chapter 2 describes the Literature Review, explaining about Smart Parking systems, Background Subtraction and Deep Learning methods, also, in that chapter is described the Related Works; Chapter 3 presents the Development process to conclude that research; Chapter 4 clarifies the Analysis and Discussions of Results; and Chapter 5 describes the Conclusions and Future Works.

# Chapter 2

# Literature Review

Chapter presents introductory concepts of this research, as well common terms in the Smart Parking, Background Subtraction, Image Processing, and Deep Learning areas, also presenting its related works.

## 2.1 Smart Parking

Over the years, the population has been steadily concentrating in urban locations, occasion increase in the average size of the population, and urban areas. That growth due to improvements in several city areas, such as surveillance, technology, economy, and according to researchers, has the forecast of 85 percent of growth into the next 30 years [10]. With this event, the city model has to be renewed to deal with new daily and long term issues [1].

The Smart City concept segments these problems in six different categories: Smart Environment, Smart Economy, Smart Governance, Smart Mobility, Smart People, and Smart Living. Also, that concept represents a city that has urban development based on technological advancements, focused on reducing mainly environmental, management, and urban life costs, since saving environmental resources has become an increasingly debated topic.

Into this term it is possible to deal with almost the city problem types and classifications, as issues based on the city structure are treated by the Smart Environment area, even as economy issues can be treated by the Smart Economy area, governance issues can be treated by the Smart Governance area, mobility issues can be treated by the Smart Mobility area, there are also the Smart People and Smart Living areas, ensuring people's quality of life [10]. Table 2.1 shows Smart City function in these areas.

Table 2.1: Smart City areas description, reproduced from [10].

| **Smart City Areas** | **Most Common Functions** |
|---|---|
| Environment | Attractive to natural options;<br>Reduce pollution;<br>Increase environmental protection;<br>Sustainable resource management. |
| Economy | Encourage entrepreneurship;<br>Flexibility of the labor market;<br>International embeddedness;<br>Increase economic stages. |
| Governance | Improvements in public and social services;<br>Transparent governance;<br>Political strategies;<br>Better perspective conditions. |
| Mobility | Availability of ICT-Infrastructure;<br>Improvements in terms of sustainable Innovative, and safe transport systems;<br>Reduce environmental needed resources. |
| People | Encourage qualification level;<br>Social and ethnic plurality;<br>Better long term quality of life. |
| Living | Improve social, touristic, cultural, health, and individual safety and facilities. |

According to Table 2.1, the Smart Cities term becomes to model a new society, focused on applying technology and policies as the primary resources to reduce environmental costs, improve public transport systems, and providing long term quality of life [10]. Thus, the Smart City concept evolves systems integration through ubiquitous computing, allowing different systems to communicate, providing an integrated urban environment [5].

Based on this integrated urban environment, the Internet of Things (IoT) can describe

the ubiquitous network as sensors or devices capturing the environment behaviors and actuators computing, monitoring, controlling and actuating into the environment, making it possible to create new automatic solutions to the city organization strategies, advantage time, energy and other resources cost [2].

Also, that concept allowing the development of some automated systems, using remote computers and devices through several system types, as Multi-Agents Systems (MAS), Global Positioning System (GPS), Wireless Sensor Network Systems (WSNS), Computer Vision Systems (CVS) or other variations to work when the environments are dynamics and non-controlled, such as mobility analysis with too much objects motion, solving complex parking issues [6].

In that way, some crucial function into the Smart Cities it is the mobility services allowing accessibility and efficiency in several daily activities, such as travel to work, school, or entertainment places. The population growth affects the operation of these services, causing traffic congestion, extra time costs to drive and expensive time, fuel, energy, and money costs mainly in search of vacancies, a usual day-to-day event [11].

Thus, surveys indicate that 95 percent of usual vehicle time usage, it is at the park, and just 5 percent of this time it is in motion or driving since people usually use this patrimony to travel to work, leaving it at a park [12]. Therefore, Smart Cities has Smart Mobility area, and to improve the Smart Mobility World, the researches describe nine central regions to manage the urban environment and each one of these areas has a significant function in the city organization as described below:

- Driving Safety: it consists in to develop and update technologies to safe and secure mobility in iterations with other vehicles, preventing traffic-accidents;

- Smart Traffic Lights: it consists in to develop and update technologies to provide better lightning structure, reducing and improving the traffic flow;

- Sharing and Urban Mobility: it consists in to develop and update technologies to improve sustainability in urban transport through sharing vehicles;

- Electric Mobility: it consists in to develop and update technologies to save environmental resources;
- Green Mobility: it consists in to develop and update technologies to minimize the environmental impact;
- Smart Payment: it consists in to develop and update technologies to change the conventional payment methods reducing time;
- Intelligent Transportation: it consists in to develop and update technologies to reduce the need for more transportation infrastructure; and
- Smart Parking: it consists of to develop and update technologies to automatize the arrival and departure of vehicles in parking spaces, saving time, fuel, and reducing city pollution and congestion.

Mobility issues it is found in almost all the big cities, and the vacancy search in traffic situations can be the most expensive activity considering time, fuel, energy, and pollution as the primary resources [3]. Reduce it is resources, provide convenience and speed solutions in traffic congestion, improvements in people's stress and energy levels, and offer a significant reduction in search time of parking spaces. It is a solution that must be developed by the Smart Parking area, providing a suitable environment for the city needs [13].

The Smart Parking systems can be divided into four main categories, they are Parking Guidance Information System (PGSI), it provides surveillance and monitoring traffic systems in static or dynamic environments; Transit Based Information Systems (TBIS) focuses on providing park-and-ride facilities through real-time information; E-parking provides an alternative to reserve parking spaces, reporting to the user that information; and Automated Parking deal with the automatic allocation of parking spaces [4].

The Smart Parking system is not only used to help drivers in daily activities but also vehicle park management. Recently, Parking Companies has implemented systems to manage the environment, such as allocate cars in a better vacancy to the company

or report to the drivers if their vacancy is available [14]. Figure 2.1 shows how Smart Parking Systems usually works based on the Internet of Things and Embedded Systems.

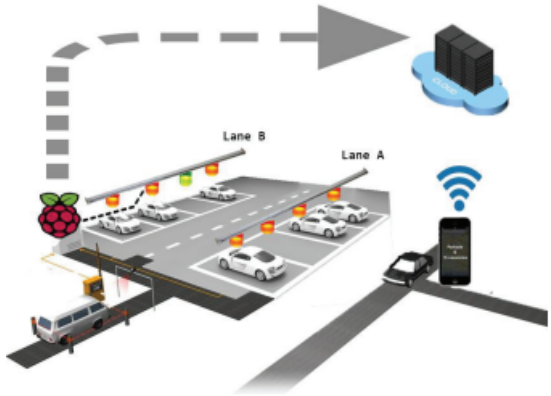


Figure 2.1: Common Smart Parking System, reproduced from [15].

As described by Figure 2.1, a usual Smart Parking system needs a purpose, as to take control of a parking space. To be able to produce this solution, it is needed a controller device and at least one agent device (see A and B devices respectively), the controller device it will be able to communicate to agent device to ask if it is an available vacancy and the role of agent device it answers the vacancy spot status. All of this communication usually, it is through an inter-network installed into the place [16]. Based on this proceeding, Figure 2.2 shows the several tendency technologies adopted in smart parking researches from 2001 to 2016.

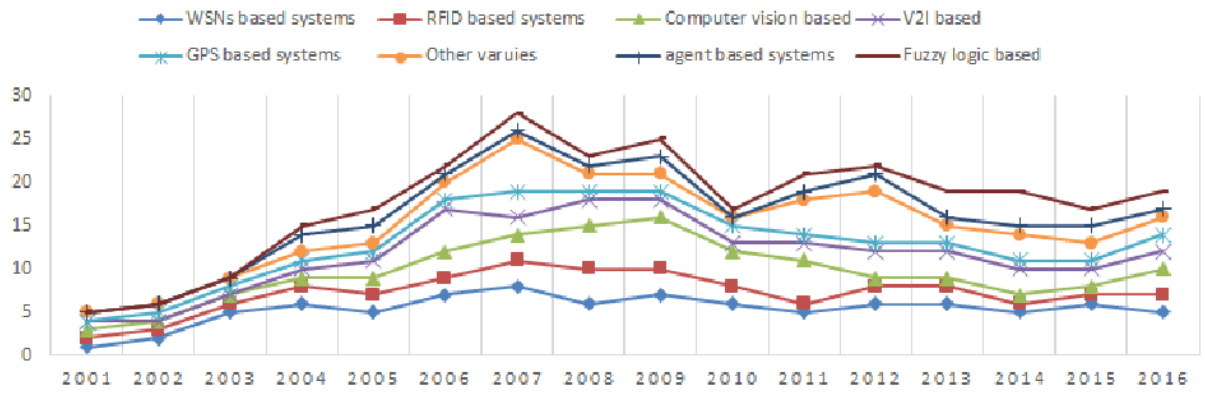


Figure 2.2: Tendency technologies in the Smart Parking domain, reproduced from [6].

According to Figure 2.2, the tendency of technologies applied in Smart Parking domains presents that systems based on Fuzzy Logic and Multi-Agents are the most common solutions in this area. But, that solution has limitations in terms of installation costs and infrastructure issues, such as difficulty and complexity of finding the professional manufacturer or in structure analysis to find the best device positions [6]. Thus, systems that do not need all of this infrastructure support should be a better solution considering these issues and costs.

As Figure 2.2 illustrates, Computer Vision-based systems also tend in this period. That kind of system can be accurate within use several devices, such as sensors or displays, and can report this information using IoT concept, through easy device communication [2]. Also, robust Computer Vision systems analyze video information in adaptive ways and in real-time, that is, in the most varied and non-controlled environments, such as in a day or night lights, sunny or rainy weather, or in complex environments as much object's motion [8].

Thinking in developing a system focused on Smart Parking solutions, owing to reduce time, fuel, environment and money resources, this research develops a vision-based real-time system effortless to install processes and mainly low cost solution, only using a vision-based device, able to verify and update vacancy spaces status through Embedded Systems and Internet of Things concepts.

## 2.2 Background Subtraction

In the past, tracking and detecting systems have limited computational power to create real-time applications. In this way, the most common solution was restricting themselves to work in controlled environments and scenes [8]. Recently, the computational power provides better conditions, through increasingly memory capacity and processing power, enable to the Visual Surveillance researches to rapidly increasing and automatic processing methods in real-time [17].

Currently, the real challenge is developing automated systems through Computer Vision, able to detect and track moving objects, succeeding translating their behaviors to the computer. Computer Vision methods are designed to solve potential observations problems and modeling the real world, interpreting visual information offering autonomy to the computer to detect, decide, and to recognize objects classes [18].

Robust tracking and detection systems it is used in several applications, like surveillance, monitoring, and support services, estimates of areas with more movement, object recognition, sound detection, and to solve various potential image problems, like vehicle collision detection [8]. Thus, that type of system can act in different areas, such as vigilance, medicinal, or entertainment, just adapting it is methods to each purpose.

To be able to produce this solution, the Computer Vision area needs one image or sets of images as video sequences. Images it is computationally represented by a frame tensor, or set of colors into coordinated coordinates, usually known as pixels. Thus, a frame is computationally represented as a tensor or matrix of height per width length, and each one of these coordinates or pixels it is model as Equation 2.1, making a set of color or a history of colors, in video situations [8].

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) | 1 < i < t\} \tag{2.1}$$

These frame structure also is visible in Figure 2.3, where it is defined by a bi-dimensional function of discrete and integer spacial coordinates $(x, y)$, being proportional to the colour intensities or bright. Figure 2.3.a) shows a graphic representation of image, the closer to the $x$ and $y$ plane, the darker the image becomes; Figure 2.3.b) shows a matrix of visual intensities, and Figure 2.3.c) represents a numeric 2-D, when 0, 0,5 and 1 correspond to black, gray and white, respectively [18].

According to these model, a pixel $X$ it is represented by a colors set $I(x0, y0)$ and a set of frames it is represented by $\{X_1, X_2, X_3, \ldots, X_t\}$, where the index assumes video frame rate value. This set of pixels, when analyzed with it correspondent pixel from next frame, makes a color history, and these values can be studied as a probabilistic Gaussian

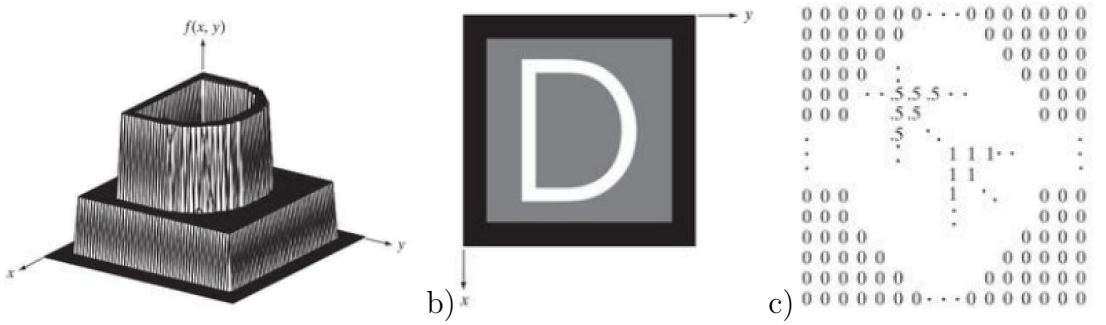


Figure 2.3: Digital image represented as a) colour intensities, b) matrix of visual intensities and c) numeric matrix, adapted from [18].

distribution [8].

Computer Vision systems must be sensitive to changes in the object status, as movement variation, insertion, removal, and object overlapping. This approach assumes the observations are independent at each pixel and can be applied to pixel regions to improve the results. Also, this system needs to be efficient in detecting shadows, illumination, and short or long term changes [17]. The main issues in the tracking process are described in the sequel.

- Movement detection: the background model should be adaptive to changes in the object status and positions, like insertion, removal, overlapping, and shadow; and
- Illumination changes: the background model should be adaptive to gradual light changes on the scene.

Thus, in several scenarios, when the state of the objects (intensity changes in the same pixel position) undergoes alterations in a set of ordered frames, there are known as foreground, and the static (multimodal) distributions of intensities over time are known as background. That classification process is commonly known as background subtraction or background extraction [8].

Through the analysis of a set of ordered frames and the application of the background subtraction method, it is possible to model the scenario background, considering the most several conditions of the object states in the scene and obtain an average image,

complementing the error between these two picture categories [8]. So, the scene model process should adaptive get the set of frames updating it is pixel values when an object is considered part of the foreground.

Background Extraction assumes that each pixel frame values are in history structure, based on this technique always be applied to frame sequences. The positions classified foreground suffer post-processing, where another classification is stored [19]. The research objective should work with the most relevant rating, like using the background to restore images or using the foreground to detect movement, as shown in Figure 2.4. This process could enables the system to reduce the computer power consumption if applied as an automatic movement detection criterion, only using processing power when it criterion is true.

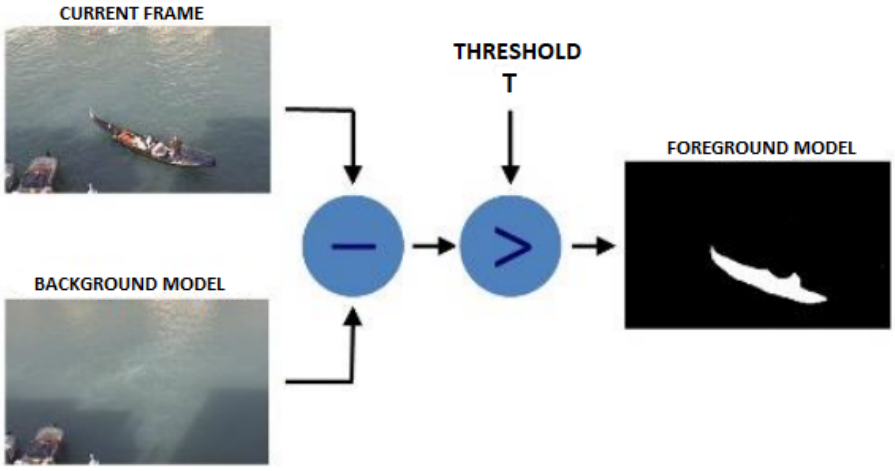


Figure 2.4: Background Subtraction process, adapted from [20].

The background subtraction method has an image segmentation process; the most common types are the Subtraction, the unimodal, and multimodal distributions. In the first approach, the model detects changes between two image frames; the current video frame $f(x, y, t_i)$ and the scene static average image $f(x, y, t_j)$. If the difference is smaller than a defined threshold, that pixel composes the background, otherwise, comprises the foreground indicating object motion [18].

This model cannot be useful in scenarios that have too much illumination changes

because it is susceptible to small alterations, resulting in a non-realistic background model. Also, it assumes that all frames are of the same size and with an ordered sequence of frames [19].

The second type can be modeled by a single distribution, sustaining the most relevant pixel values in the same position and assuming that the intensity of each pixel values was above the unimodal distribution. According to the intensity, value moves away to the distribution mean, that value does not compose the background, representing an object in motion. However, this model also cannot handle illumination changes, because it is susceptible to small alterations in the scene, also resulting in a non-realistic background model [19].

The Multimodal Distribution creates more than one unimodal distributions of pixel values to the same position, enable the system to create an average cost of the n better distributions, representing the most relevant values. In this approach, works with some distributions, it is enough to represent all the scenes variations, like leafs or waves, composing the better alternative to solving the small alterations problem [8].

When the scenario changes in a short period on a set of ordered frames, the alterations in the background are not permanent [14]. In this case, probably some objects will be overlapped in the same pixel value, such a movement in waves, sky, or leafs. To these cases, some models who treat a single distribution are not adequate. Observing this problem, a mixture of these values can be a potential solution [8].

Considering that property, Stauffer, and Grimson describe in their work a solution using the term Mixture of Gaussian Distributions, observing the same pixel values in a period and using three or five distributions per value [8]. To each frame update, it will be necessary to verify if the current pixel value matches with some preview ordered distribution updating all the positions.

That preview distributions represent the mean of the Gaussian wave, allowed that each pixel is classified in one of these mean distributions. If it does not match, just the last distribution will be replaced by that pixel value, but with a lower weight and high deviation. In this way, the method adaptation power could get all the changes in the

scenario. That mixture of distributions is also known as pixel history.

This pixel history is composing by the coloration means of each pixel in RGB format, the density probability function, the weight value, the matching value, and the distribution standard deviation [8]. Without the match value, its components are represented in a normalized format (from zero to one) according to the property of a Gaussian distributions. Also, all of these components belongs to just one Gaussian distribution, describing as mixture of Gaussian the set of distribution components.

## 2.3 Deep Learning

Has long been arguing about it machine intelligence, thinking, and automated systems subjects. This discussion become possible through technology development focused on helping humans to execute since daily activities to substantial hard activities. Thus, enable computer systems to act into several areas, as research support, medical diagnostics, investment solutions, or pattern recognition, creates a new environment, lighter and less physically costly for humans [21].

Based on it is discussions, Artificial Intelligence (AI) is a research field aiming at solving fundamental human tasks challenging to describe all processes, as to recognize faces, words, voice, or solve math problems. For example, there is no way to explain how you look for some people face and remember their name and information, unless through the Neuroscience field because there are no steps to describe and follow [21].

This process is challenging due to it is an automatic function of human brains, as Neuroscience describes, it could be possible because the human brain has hundreds of nerve connections searching information all the time, constituting a cognitive system based in neurons sharing information through the synapses, that enables a neuron pass electrical and chemical signals from one cell to another. Therefore, AI enables systems to emulate some of it is connections, creating a set of methods that allows machines to find out necessary patterns to conclude specific tasks [21].

Thus, most indicated challenges in AI are solved through pattern recognition, such

as image data patterns to face detection. Therefore, Deep Learning describes different patterns sets with multiple learning levels as biological neurons works, presenting a structure based on graphs with learning processing layers increasingly complex, as illustrated in Figure 2.5 [22].

These Deep Learning models work through artificial neurons, artificial connections, and input and output values. It is Artificial Neurons represent vertices focused on learning thought train data, Artificial Connections represent edges to share results into each neuron, and input or output values represent possible input data and solutions. These structure in layers present three functions: the first layer extract patterns of the input value, the second layer get results from the previous layer as input values, and the last layer shows its correspondent output values.

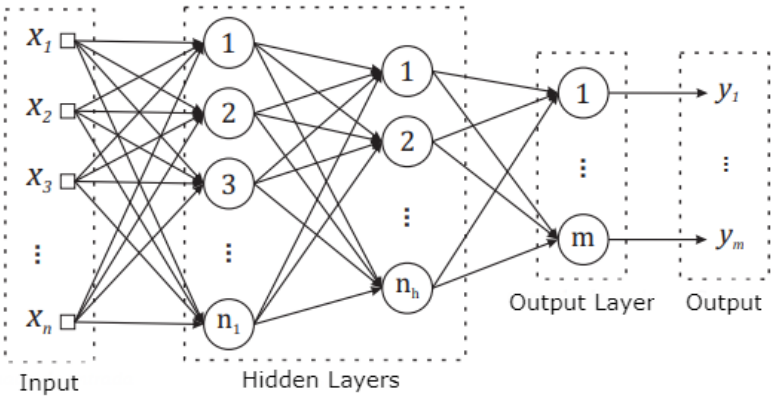


Figure 2.5: Neural Network structure, adapted from [23].

Moreover, an input value goes through the analysis by the first layer; the second layer analyzes previous layer output; the third layer does the same process (if it exists), until the output layer, which produces a final answer to the problem. Into this structure, the first layer represents the Input Layer term, second, third, and other possible layers before the last layer, are represented by the Hidden Layers term, and the final layer is represented by the Output Layer term [22].

Each neuron represented into a Deep Learning model works with simple math equations connecteds, creating an environment of parallel computing. These math equations are described as Activation Functions and the intelligence behind this structure it is in the

weights update process, described in the subsection Training Processes. That means that each neuron has five directives: input values, a sum component, an activation function, a weight, and output associated [22].

There are three most common paradigms that this models can assume: Supervisioned describes a model that presents a correct or defined answers to each input value; Non Supervisioned describes a model with structured in answer classes, aiming to find common parameters and aggroup it into labels; and Hybrid models presents a model using both paradigms [24]. The Complexity evaluation of these models commonly based on the time associated with estimating the solution though train data patterns, regardless of whether the answer is correct [25].

Moreover, it is possible to aggroup these models by Feedforward and Feedback terms. Feedforward consists of a static, only presenting one direction (forward) and no memory network, known as Multiple Layer Perceptron (MLP), designated just to produce one output, independent to the previous model state; thus, this model does not present loops [26]. On the other hand, Feedback or Recurrent models represent dynamic and dependent networks, that modify it is neurons inputs using previous model state and neurons results, being structured with loops. These structures are shown in Figure 2.6 [26].

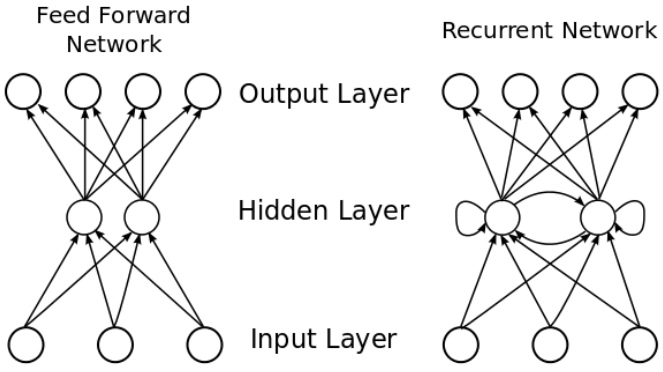


Figure 2.6: Feed Forward and Feed Back (Recurrent) Networks, reproduced from [26].

As Figure 2.6 describes, Feedforward Networks only share neuron results in one direction, making each neuron output being shared to the next layer as input, while Feedback Networks allows as many loops as needed to get solutions closer to expect output, being

able to feedback neurons and update weight as necessary [27].

### 2.3.1 Activation Functions

Deep Learning models present different approaches to produce outputs closer to the expected results, as process images or text data; present a binary or multi labeled outputs; and set up its neurons in using different technologies. Due to this, Activation Functions are present within each neuron to ensure the mathematical set up that allows the complete execution of the training processes, presenting nonlinear, monotonous, and differentiable properties. That makes consistent results and could be created by the weights addition into the input data.

#### Step Function

Therefore, the process of associate a weight to each neuron input data represented as $w_1 x_1$ it was developed to demonstrate data importance level. This feature becomes a threshold when the thought of as weighted sum equation, only accepting values characterized by the function $\sum w_1 x_1$, returns 0 or 1 according to the equation described in Equation 2.2 [26].

$$y = f(x) = \begin{cases} 0, \sum_j w_j x_j \leq 0 \\ 1, \sum_j w_j x_j > 0 \end{cases} \tag{2.2}$$

This function it is called Step or Threshold Function and belongs to the Non-Linear Activation Functions subject, due to the fact that independent of the input, the only possible solutions assumes 0 or 1, that is when $y$ assumes 1 and $x$ assumes values greater than 0 or $y$ assumes 0 and $x$ assumes negative values, as shown in Figure 2.7, the only possible results it is 0 or 1 [28].

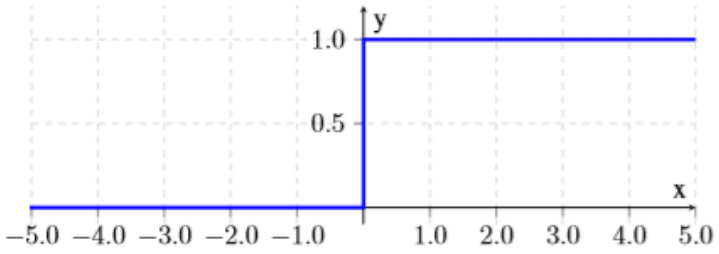


Figure 2.7: Step (Threshold) Activation Function, reproduced from [26].

### Sigmoid Function

Sigmoid Function becomes to standardize neural networks that present output directly affected by weight or bias small changes, generating an arbitrary result that could affect the entire network structure. Sigmoid Neurons allows changes into their weight or bias values, just affecting their node result, being able to produce any amount between 0 and 1, according to the expression $\alpha(wx + b)$, when $\alpha$ match to the function sigmoid [26] expressed by Equation 2.3.

$$\alpha(z) \approx \frac{1}{1 + e^{-z}} \tag{2.3}$$

Therefore, it is possible assumes that values of $\alpha$ directly imply small weight changes of Sigmoid Neurons presenting a linear output or a differential equation, increasing predictability of each network neuron [26]. Thus, Sigmoid values restrict themselves to an interval of 0 or 1 according to the function resenting by Figure 2.8:

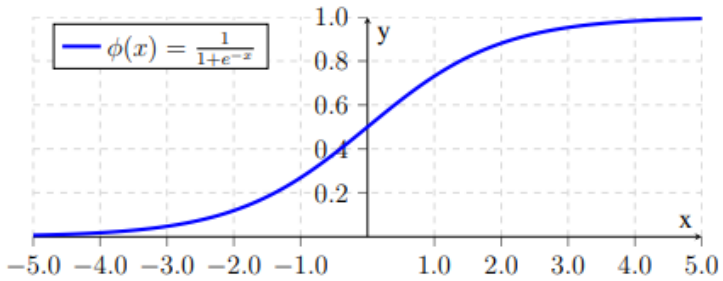


Figure 2.8: Sigmoid Activation Function, reproduced from [26].

### Hyperbolic Function

Hyperbolic function presents different comportment between Perceptrons Networks but performs a similar function of Sigmoid, restrict themselves to an interval of -1 and 1 output values [26]. Consistently improve Sigmoid function through a faster saturation or approximation of their values, comprising the Hiperbolic Tangent described in Figure 2.9:

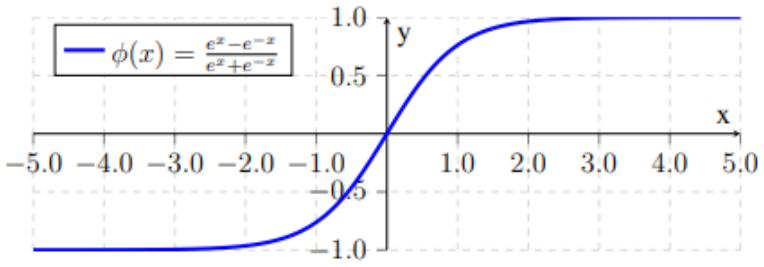


Figure 2.9: Hyperbolic Activation Function, reproduced from [26].

### Softmax Function

Softmax Function presents the probability of input values to adequate output to a previously established class, due to it operation, the Softmax function is commonly used in the last layer, producing the output value. Therefore, its neurons outputs it is restrict to an interval of 0 and 1, 0 for smaller matches, and 1 for more significant matches. The matches value it is based on the probability of the input value belongs to each one of the final classifications, and it is denoted by Equation 2.4:

$$\phi_i = \frac{e^{zi}}{\sum_{j\ \epsilon group} e^{zj}} \tag{2.4}$$

where z represents the vector of outputs, i represents the index of each output node, and j represents the index of all group nodes. Making with that output presents in math way the hierarchy classes of each input value, depending on the outputs of antecedent neurons; that is, it depends on the recognized patterns [29]. It function it is commonly indicated to image classification, based on its feature of demonstrate the possibility of

each input being considered closer to each expected final classification.

#### Rectified Linear Unit Function

In view of Sigmoid, Hyperbolic and Softmax functions that are commonly used in output layers, the Rectified Linear Unit (ReLU) constitutes values for hidden layers, presenting different processes, as an approximation to some value just it is present into the negative domain to 0, while in the positive domain their values are not restricted, as described in Figure 2.10 [26]. It indicates that after weights initialization process, almost 50 percent of output values assume zero.

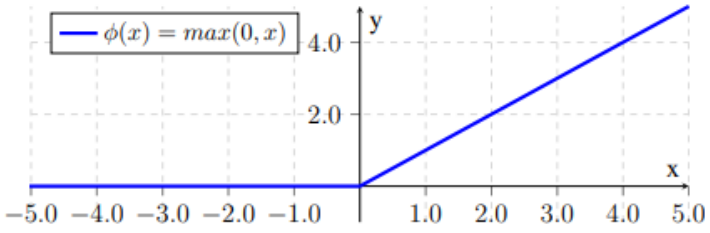


Figure 2.10: Rectified Linear Unit (ReLU) Activation Function, reproduced from [26]..

### 2.3.2 Training Processes

The weights update process it is organized according to train data patterns, it usually starting using randomly or small values, modifying it on each iteration to get solutions closer to the expected result, providing loss minimization metric. The process to improve weights update increase the weights interval, searching for typical values. Thus, the Learning Rate coefficient significantly increase the range may cause too large values, as represented in Figure 2.11, increasing the loss rate [26].

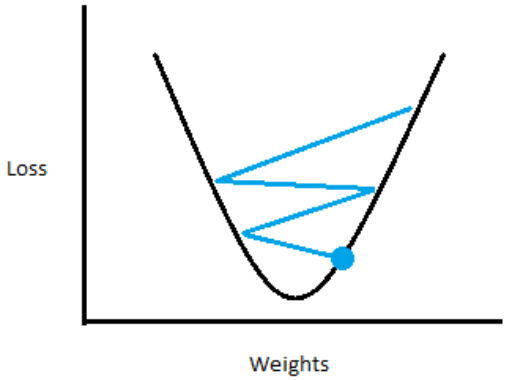


Figure 2.11: Learning Rate, adapted from [26].

Therefore, Neural Networks (NN) does not have a criterion that prevents the creation of many layers; it must be created according to the problem needs, allowing the NN to make sophisticated decisions about the subject [21]. But, too many iterations not ensure appropriate results for unknown patterns, characterizing the Overfitting term, as Underfitting represents a model that does not provide suitable results, drawing a straight line between their answers [27]. Both methods are shown in Figure 2.12, as well as the appropriate capacity of the models.

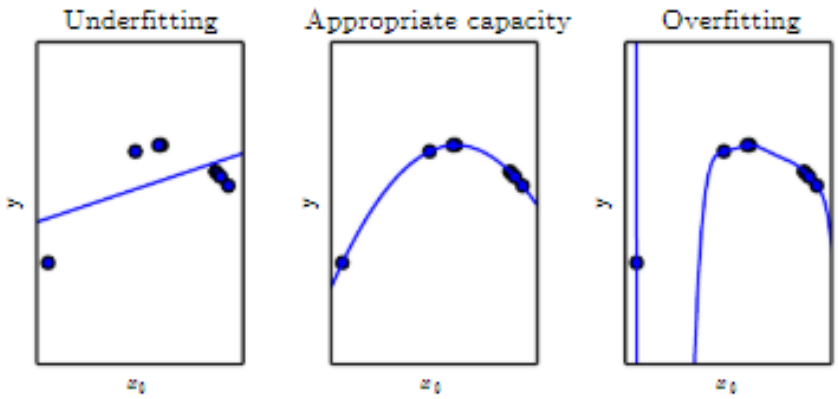


Figure 2.12: Underfitting, Appropriate Capacity and Overfitting behaviors, reproduced from [21].

To prevent it is overfitting issues, it is recommended to decrease the complexity of the model, reducing active layers, neurons, or epochs numbers. It could be possible mainly

four ways: manually removing it is neurons; using dropout function; using early stopping feature, and through kernel regularization [21]. The process to remove it is the chosen number of neurons could be done in the model creation process through not adding new layers, but does not ensure the most trustworthy solution or balanced model.

The dropout method presents the process of drop a chosen percentage of connections between their neurons randomly; it consists of selecting the percentage and each neuron that it is dropped without human iteration providing an increase in chances to have a balanced model. Figure 2.13 shows the dropout method applied during the training process, removing the sealed units with X [30].

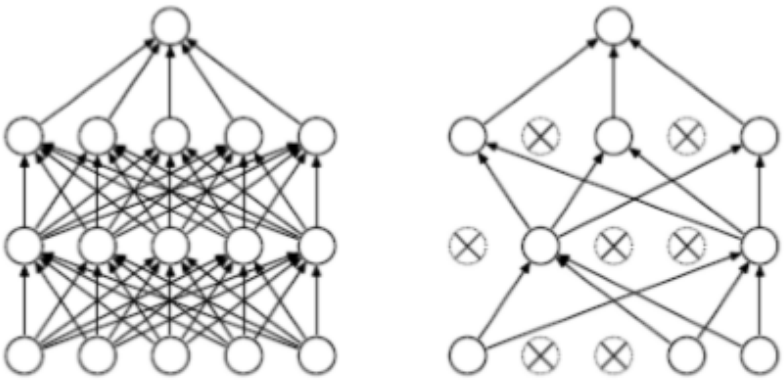

Figure 2.13: Fully-Connected model and Dropout method, reproduced from [30].

Early Stopping method provides the model's validation loss control through stop early the training process, instead of making a fixed number of iterations. This method stops when the validation data set loss coefficient assumes minimal and almost equal loss values, as shown in Figure 2.11. That makes the model take indefinite time to conclude the training process, but allows that the model solution approaches the maximum of the expected result and stop when this happens [26].

Reduce on Plateau represents the theory of Furthermore time essential measures lose effectiveness. This concept is adapted to Deep Learning to stop the learning rate when a specific metric has stopped improving. The metric could be chosen between the evaluation metrics, producing improvements in terms of model evaluation [31].

The method of penalizing the weight matrix on each artificial neuron is known as Kernel Regularization and assumes that a Neural Network with smaller weights best presents smaller domain solutions, adding the regularization term into the cost function. Kernel Regularization presents two categories: L2, which forces the weight tensors to decay towards zero value and L1, that penalize the absolute weight tensors, reducing to zero [32].

Batch Normalization comprises the technique that uses very high learning rate values, may have been less careful during weight initialization. Usually, it is used one batch by iteration, due to the amount of computational processing required. Batch Normalization provides the uses of several small pieces of samples by repetition, taking advantage of the parallelization capability, as Graphics Processing Unit allows, accelerating the learning process [26].

Another possible solution for Overfitting is the amount of sample data that a Deep Learning model has access; still, it directly affects it is chances of getting closer to the expected results or accuracy evaluation. Therefore, use techniques to increase the data set provides changes in terms of colors and geometry transformation, to manipulate train data, offering as many samples as the machine learning professional find enough, as disposed into Figure 2.14 [26].

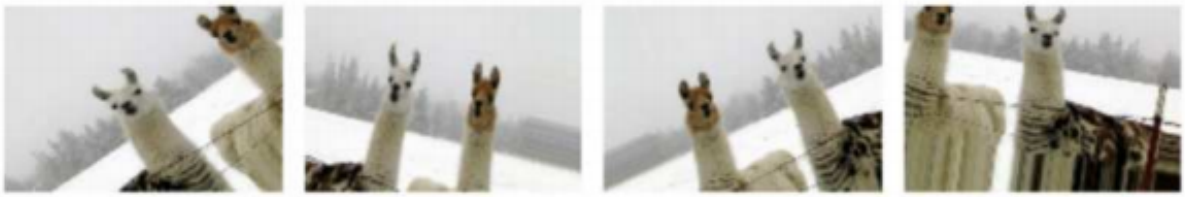

Figure 2.14: Samples of Data Augmentation, reproduced from [26].

Therefore, Neuron Networks models aiming to produce a set of weights that makes the model get closer solutions to the expected results. During the training process, the results it is rated by the difference of train outputs and the expected outputs, commonly denoted by Equation 2.5, where $T$ represents the train set; also, $j$ represents the model output iteration, characterizing Backpropagation term [26].

$$E = \frac{1}{2} \sum_{k \in T_r} \sum_{j=1}^{m} (y_j(x_k, w) - d_j k)^2 \tag{2.5}$$

Moreover, humans apply relevant knowledge from previous learning experiences. Aiming to use this method from computer systems, Transfer Learning is a technique that provides a new purpose for previously trained models. It consists of the improvement of learning through the transfer of knowledge from a related purpose that already been learned. As an example, it is possible to train a Deep Learning model on a base data set for vehicle detection and repurpose to a second network and data set from wheels detection [33].

Based on Transfer Learning, Fine-Tuning presents a solution to be able to increase or decrease the learning rate needed into each network layer, providing network optimization to minimize loss rates in different network domains. It consists of the implementation of Transfer Learning with the replacement of activation functions as needed to solve a previously classification problem to become the newest NN structure able to the new classification domain [26].

### 2.3.3 Optimization Methods

Optimization methods allow the research for minimum and maximum values for increase performance through choose variable learning rate values for different desired domains and solutions. It consists of the utilization of parallelism to improve the computer process of solving math equations in a lot [26].

Stochastic Gradient Descent (SGD) reduces minimum local occurrence through mini-batches of samples reducing the learning variance, becoming the model a highly parallelable method [do Nascimento, 16]. Adam, it is a method that comprises an efficient stochastic optimization based on low memory usage and calculating individual adaptive learning rates for each batch samples [26].

Furthermore Adam is an adaptive learning rate method, that uses the estimation of the first and second moments of the gradient to adapt the learning rate, being moment the

expected value for a variable in the N potence. In a probabilistic way, the moment zero demonstrates the total probability, the moment one represents the probability mean, the second moment describes the distribution central variation, the third and four moment represents the distribution asymmetry and kurtosis, respectively.

### 2.3.4 Evaluation Metrics

A robust Neural Network is represented by values to their accuracy and loss functions, as a percentage of hits and loss. In terms of a normal distribution, accuracy function presents measures that approach decimally the value 1, assuming 0 to pour measures and 1 to suitable measures. The same can be described as loss function, but it is valued opposite, considering 1 to pour measures and 0 to useful standards [21], as presents Figure 2.11.

These train metrics represents it is functions to the percentage of hits and misses to train patterns recognition, validation metrics represents the same percentage in sets destined to validate train patterns and test metrics represents the percentage of hits and misses to unknown patterns and NN should be able to future, by input set increment, learning and improve it is accuracy [21].

There is some significant terms in order to evaluate the system, $P$ (Positives) represents output set that it is classified as positives; $N$ (Negatives) represents output set that it is classified as negatives; $TP$ (True Positives) consists into the correctly outputs classified as positives; $TN$ (True Negatives) consists into the correctly outputs classified as negatives; and $FN$ (False Negatives) it is the number of incorrect output classified as negatives [34].

All through the evaluation process, two other metrics it is considered: precision and recall. Precision regards the percentage of outputs correctly classified as true on the classified data set, and it is denoted by Equation 2.5. In another way, Recall or Sensibility represents the percentage of output correctly classified as accurate into the classified data set, denoted by Equation 2.6 [34]. F1-Score (F-Score or S-Measure) represents the measures of the test accuracy, considering precision and recall as essential features, it is

equation it is described as Equation 2.7 [35].

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.8}$$

Confusion Matrix or Error Matrix comprises the visualization of an algorithm through the matching of predicted classes and actual classes. This method is denoted by reports of false positives, false negatives, true positives, and true negatives, also demonstrating accuracy, precision, recall, and f1-score. As demonstrated in Figure 2.15, it is created a correlation matrix, presenting the number of inputs classified as positive and negative [36].

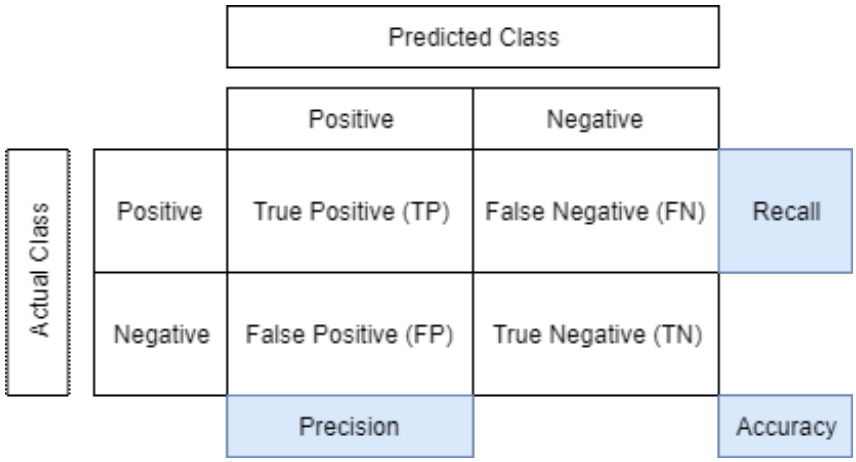


Figure 2.15: Confusion Matrix and its respective functions.

The model evaluation presents several metrics, and depending on the model domain, it is solutions could be binary or categorical, as True or False, and the percentage of each input value has into the solutions set. Thus, Categorical Cross-Entropy represents a loss function for problems where one result can be correct, and it is probably is compared by

the distribution of predictions, producing only one correct answer [37].

Moreover, the Cross-Validation method consists of a technique that, at the end of each training epoch, model it is subject to an unknown data set, aiming to maximize neural network capacity, indicating the better iterations number to adjust their weight values. Therefore, the test data set is set up into two groups: validation, which provides Cross-Validation application and test, that offers models evaluation. The weight adjust process consists of determinate the iteration that has fewer error values in the test set. According to this method, the samples need to be disposed of in just one data set [38].

### 2.3.5 Convolutional Neural Networks

Humans successfully classify and recognize objects, persons, or animals based on vision, low-level features as formats, corners, lines, curves, or colors [39]. Motivated by image object detection and classification through image analysis Convolutional Neural Networks emulate this human capacity, computationally processing images as tensors and denoted by $H$x$W$x$D$, where $H$ represents image height, $W$ to image width and $D$ to image channels or dimensions [26].

In this concept, tensors usually represent the image color intensity in all it is coordinates; it consists of three dimensions or channels for color images: red, green, and blue intensities. By default, a Convolutional Neural Network presents as first feature the Convolution Layer. It consists when using a convolution kernel, to go through image pixel to pixel applying the multiplication of kernel value and pixel intensity, as shown in Figure 2.16 [26].

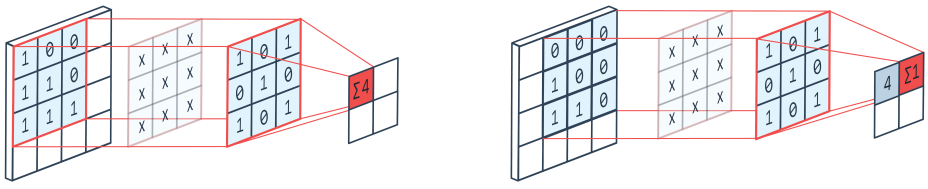


Figure 2.16: Convolution sample.

Moreover, apply its is convolution technique enables to obtain low-level characteristics,

as a demonstrated corner in Figure 2.16. This first process is represented as the second image part, where 0 represents no accounted intensity values, and the third image part represents the tensor kernel to be multiplied. The main feature extracted it is represented as an Activation Map, the probability of each pixel to demonstrate it is low-level image features [39].

Therefore, each Activation Map it is processed by one neural unit and presents the probability of images to contain a specific low-level feature. This process has, as a consequence, a very high-resolution feature map, making possible decreasing the network speed [40]. Thus, the Pooling method aiming to reduce the amount of generated data, through a downsampling process, providing an increase in network speed, a low-resolution feature map [26].

Once Convolutional Neural Networks extracts low level features from input images and combine features for speed improvements, next process it is combine several hidden layers using ReLU function and fully-connected layers providing discover most relevant features into the input image, that may assist in the final classification, as shown in Figure 2.17 [26].

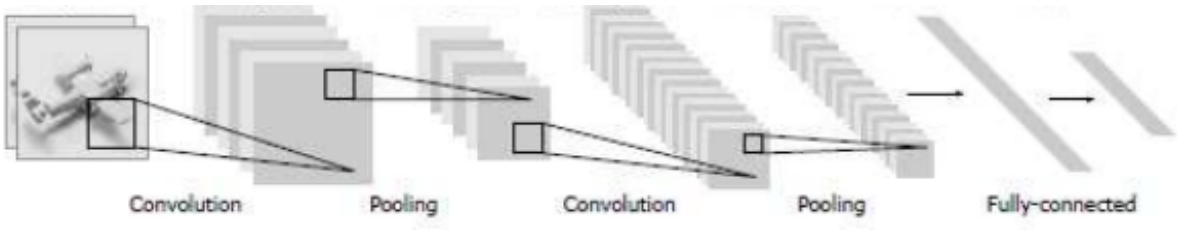


Figure 2.17: Convolutional Neural Network structure, reproduced from [26].

### 2.3.6 MobileNet

Differently from Convolutional Neural Networks, MobileNet presents a Google proposed architecture for suitable mobile and embedded based vision systems. It comprises a light Deep Learning model due to the significant reduction in layers parameters through Dephtwise Separable Convolution. This concept also deals with depth dimension, or the number of channels, requiring a three-channel input image; providing the newest process

for convolution. Its process consists of factorizing the kernel into three MxNx1 different kernels, as shown in Figure 2.18 [41].

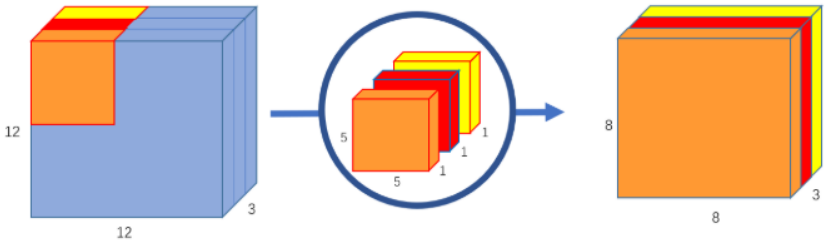


Figure 2.18: MobileNet Convolution's process, reproduced from [41].

Through this process, each kernel iterates one channel of the image, and generates one intermediate convolution image with the same number input of channels, being smaller than the normal convolution process. The second process consists of producing a final image through a pointwise kernel, to get an image of 1 channel. MobileNets comprises an efficient model focused on speed and accuracy improvements [41].

## 2.4 Related Works

Motivated by the failure in object distinction provided to the background extraction technique, Friedman and Russel have to propose a pixel classification in distinct classes. That project identifies and highlights shadows, improving the vehicle's identification in tracking problems. In this research, they concluded that the application of the mixture of Gaussian is not capable of producing excellent results in scenarios with extreme illumination variations [42].

In your research, Ridder, Munkelt, et al. proposed the utilization of an adaptive background estimation with the Kalman filter, making the model sensitive to illumination changes in the scene. That application does not consider several object movements; also, the authors concluded that detection does not depend on the pixel position and that the smaller image resolution, the better the result will be [43].

Aiming to create a robust system of adaptive tracking, sensitive to objects overlap,

illumination changes, objects iterations (insertion, removal, movement) and shadows in scene, current failure of the traditional methods of object detection, Stauffer and Grimson pattern a color values of a determined pixel according to the mixture of Gaussian and obtained a classification with heuristic evaluation in background, re-adjusting the classifiers [8].

That works concluded the better probabilistic method of tracking objects in this epoch, but this process could be slow learning and cannot distinct moving shadows and moving objects [8]. Thinking in improving the mixture of the Gaussian technique, KaewTraKulPong and Bowden present a study that reinvestigating all the equations and the equations positions on the code, updating the Stauffer and Grimson work and allows the system to learn faster with more accuracy on distinguishing movement or shadow [44]. This method could be founded in the OpenCV Library, known as BackgroundSubtractionMOG2.

Another essential background subtraction approach was self-organization through neural networks. That work can handle scenes with objects movement and illumination changes, including into the background object shadows. Also, it can deal with different types of video. Using this technique, the researchers concluded that improvement in the processing speed and accuracy in surveillance systems [17].

In order to produce a mobile system based on vacancy state indication, Valipour, et. al [45]. develop a system that reaches about 99 percent of accuracy based in VGGNet-F Neural Network. The proposed architecture uses the manual vacancy detection method, and provides a solution for mobile applications using the server as processor of input images get through the camera devices. The accuracy achieved by Valipour work it is based on its VGGNet-F structure and has as future works the vacancy automatic detection feature.

Motivated by uses the MobileNet structure through Transfer Learning and Almeida Parking Images Data Set [46] as an IoT based parking system, Jose and Veni [47] proposes reaches about 88 percent of accuracy and indicates as significant challenges the illumination changes and the vacancy occlusion due to the perspective distortion. The

accuracy achieved by this work it was considerably bellow that the Valipour proposes, and its based on the data set size and Transfer Learning architecture. Motivated by experiments in Parking Data Set, Amato et. al. [7] reaches about 2 percent of accuracy evaluation above Jose and Veni model, using a AllexNet based model and detecting its vacancy spots through manual input coordinates.

# Chapter 3

# Development Methodology

Chapter presents the development process obtained to conclude the proposed work, including the used tools, data analysis and the model development.

## 3.1 Programming Languages, Libraries and Devices

This section presents the most relevant used tools, starting by the chosen programming language, its libraries, devices and frameworks.

### 3.1.1 Python

Python is a high level, multi-platform, interpreted, and interactive programming language, poorly used as extension language to process application quickly and produce easy to use scripts. Moreover, Python is an open-source language, that comprises a community of open source libraries, especially in Machine Learning, Deep Learning areas due to clear syntax, easy understanding and efficient data structures, reducing maintenance costs [48]. That work presents Python in it is version 3.6, together with the below-described libraries.

### 3.1.2 OpenCV

Open Source Computer Vision Library or OpenCV is a multi-platform and open-source library developed to produce applications in the Computer Vision area, owing Linear Algebra, Data Structure, Digital Image Processing, and Video modules, Furthermore it presents image detection and recognition algorithms. OpenCV was developed in C/C++ but also enables support for Python, Java, and Visual Basic. In this work, OpenCV is used in image/video processing, allowing discovers in object detection and recognition area [20].

### 3.1.3 TensorFlow

TensorFlow is an open-source library developed to produce solutions in terms of quickly numeric computing, enable us to integrate applications using one or more Central Processing Units (CPUs), Graphical Processing Units (GPUs), or mobile dispositive. TensorFlow was developed by Google Brain Team, aiming to produce solutions in Machine Learning and Deep Learning areas through math operations and multi-dimension tensors [49]. In this work, it was used the implementation of the TensorFlow base and Tensorflow GPU aiming to produce an efficient Deep Learning model.

### 3.1.4 Keras

Keras is a high-level Neural Networks API, written in Python and supported by the TensorFlow library. Keras enables the creation of Deep Learning models through the quick and easy way, focused on test its methods in the as short time as possible, supported by CPU or GPU. Keras also support several model architectures, data generation, layers types, activation functions, learning rate methods, regularizers, optimizers, evaluation metrics, and including Convolutional Neural Networks and applications as support for MobileNet [50]. In this work, Keras was used to support Deep Learning methods studied in the literature.

### 3.1.5 Anaconda

Anaconda is an open-source multi-platform distribution for Python and R programming languages aiming to produce Data Science and Machine Learning applications, through simple environment management and deployment, including Anaconda Navigator GUI and command-line interface. Through Anaconda, it is possible to create several environments and install through conda or pip different libraries on each deployment environment. This distribution enables installation of Jupyter Notebook or Spyder GUI for Python and R development, making possible integration of languages and devices [51].

### 3.1.6 Sci-Kit Learn

Sci-Kit Learn is a Python open-source library focused on solve problems in terms of supervisioned and non-supervisioned algorithms. Moreover, Sci-Kit Learn presents functions and modules to evaluate these models, also allowing the comparison and report of these algorithms [52]. In this work, this library is used to evaluate and report the proposed model in terms of accuracy, precision, recall, f1-score, and confusion matrix.

### 3.1.7 Flask

Flask comprises a Python Web Micro-Framework, providing essential framework services, as sharing information through network and routes, does not need to allocate several computational resources [53]. This framework provides to this work a quick solution to be able to present exciting results in a short time.

### 3.1.8 Raspberry Pi

Raspberry Pi is a multi-platform single-board computer commonly used in the Internet of Things are to be able to acquire, process, and report data through a potential CPU and network services. This device provides a solution on low-cost and could be integrated into modules, as RaspCam V2 and Movidius specific devices [54]. Figure 3.1 presents a

Raspberry with integrated RaspCam.

Figure 3.1: Raspberry Pi's sctructure integrated with RaspCam, reproduced from [54].

Raspberry Pi 1 presents a system on chip BCM2835, ARM1176JZF-S processor of 700 MHz, GPU video-core IV, 512 MB in RAM, and input/output connections. As a complement, currently, Raspberry Pi 3B+ presents a system on a chip, a 1.2 GHz 64-bit quad-core processor, ARMv8 GPU, 1 GB DDR2 of RAM and Bluetooth 4.1 [54]. It reduces the computer provides needed processing for acquiring, process, and report information about the proposed system.

## 3.2 Data Pre-Processing

This work proposed a solution using Raspberry Pi 3B+, RaspCam V2 to acquire, process and report data, and a machine to implement and train Deep Learning models in a Python environment by remote access. The machine presents the follows features: Linux Ubuntu 18.04.1, Intel(R) Core(TM) i7-5930K CPU@3.50GHz, memory of 32GB, and two available GPUs GP102 (GeForce GTX 1080 Ti).

### 3.2.1 Data Acquisition

The data disposed of by Polytechnic Institute of Braganca (IPB) is divided into videos from the Higher School of Technology and Management (ESTIG) parking space. This

data is in format MP4 with 25 frame rate per second (fps), and represent images of movement in the park, recorded through Raspberry setup environment proposed for this work, being authorized by a confidentiality agreement between the master's student and IPB moderator.

The essential challenge provided by this park is that it does not presents divisions between each vacancy space, and due to that challenge, it is not possible to estimate the exact positioning of each parking lot, as disposed of in Figure 3.2. Also, according to the positioned camera, the vacancies spot have an angulation that does not favors its visualization, making the situation of as the vehicles its position, a clear view of the vehicle will not be possible, requesting extra attention during the analyzes.

Figure 3.2: Data extracted from IPB.

This step aiming to extract attributes that influence into the classification model, developing a use case to apply the proposed solution. Making these data understandable, there is some information about the data acquisition: it was acquired 2 videos, totaling 12 recorded hours in 25 fps. It is substantial to make it clear that this set of images does not belong to the data set created to train the Convolutional Neural Network described in the following subsection, considering a solid framework for testing.

### 3.2.2 Data Set Acquisition

The data set disposed to conclude this work presents 62740 images divided into three different classes: cars, motorcycles, and negative images. The vast majority of images come from the Open Images DataSet V3, that contains 15851536 boxes on 600 categories of images, and the resultant images come from researches on Google platform though specifics terminologies, namely 'car dataset', 'motorcycle dataset', 'vehicle dataset', 'parking dataset', 'bicycle dataset', 'people dataset', and 'animals dataset'. Through the 'parking dataset' term, it was found, in special, the PKLot Dataset [46], that provides images of occupied and available vacancies from parking spaces.

The established classes of images on this data set it was decided to consider categories of objects that may belong to the environment of a parking lot. The inclusion of people and animals into this data set belongs to the goal of reaching not only vehicle images, as found in these environments, making the system sustainable to several environments.

The disposition of images it was established according to the Cross-Validation technique together with the Softmax activation functions, that requires three classes of image sets plus as many as possible classes of objects to being recognized: train image set, validation image set, and test image set, Furthermore presents the same classes of images: cars, motorcycles, and negatives. The data set structure is disposed in Figure 3.3.

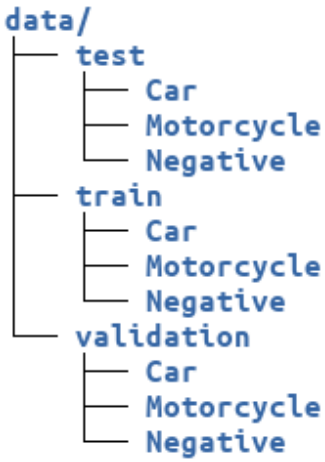


Figure 3.3: Proposed image data set structure.

The negatives image data set comprises Furthermore the images that do not belong to cars and motorcycle image classes, like people, animals, and bicycles. Also, the quantity of images disposed into each set is shown into Table 3.1, representing about 70 percent for a train set, 15 percent for the validation set, and 15 percent for test set as described in the literature as indicated percentage. It is substantial that the validation image set belongs to the train set, according to the Cross-Validation concept, and the test set must not own images from the train set.

Table 3.1: Proposed image data set structure and it number of images.

| **Data Set Structure** | **Image Classes** | **Number of Images** |
|---|---|---|
| Train Data | Car | 6156 |
| | Motorcycle | 8290 |
| | Negatives | 26989 |
| Validation Data | Car | 2596 |
| | Motorcycle | 1768 |
| | Negative | 6127 |
| Test Data | Car | 2596 |
| | Motorcycle | 1773 |
| | Negative | 6445 |
| 3 | 9 | 62740 |

Furthermore it was needed to get images from Open Images Dataset V3, that also enable the downloads it is labeled, which contains the positioning of each object of interest into each image. Due to this process, it was needed to clip each image and save it into the respective path, according to it labels, in the specific region of interest as demonstrated in Figure 3.4.

Figure 3.4: Data set labeled cut sample.

The proposed images data set is represented in Figure 3.5, demonstrating images of all classifications, that it is used to recognize positives and negatives objects of interest. As the first place, the positive objects consists into the cars and motorcycles classifications, besides the negative objects it is represented by bicycles, persons, dogs, and cats classifications.


Figure 3.5: Images set from proposed data set.

Furthermore, as previously described, this images data set was composed in the proposed model to successively train a Convolutional Neural Network to recognize objects of interest, supporting the model of vacancy state detection, through presenting the probabilities of such objects in the scene.

### 3.2.3 Data Augmentation

Starting from the data set defined in the previous subsection, it was applied Data Augmentation method to prevent Overfitting issues. The data Augmentation process consisted of the realization of mathematical morphologies in each data structure. Firstly,

it was applied image normalization en each structure; second, it was applied five parameters into the train data set:

- Random image rotations of 20 degrees;
- An image movement to 20 percent have been performed according to the longitudinal and transverse axes;
- A horizontal flip into these images; and
- Extension of the morphological bounds, in terms of color and shapes.

These applications it was enabled during the training process, creating this modified set into Convolutional Neural Network memory, which does not save these images, and Keras API provides the process through 'ImageDataGenerator' method, as described in Table 3.2.

Table 3.2: Proposed data augmentation process

| **Images Data Set** | **Data Augmentation Process** | **Parameter** |
|---|---|---|
| Train Data Set | Rescale | 1./255 |
| | Rotation Range | 20 |
| | Width Shift Range | 0.2 |
| | Height Shift Range | 0.2 |
| | Horizoontal Flip | 0.2 |
| Validation Data Set | Rescale | 1./255 |
| Test Data Set | Rescale | 1./255 |

As described, motivated by producing a model that is less likely to potentiate the overfitting problem, the data augmentation proposed it is based on mathematical transformations that comprise the distribution and differentiation of the images arranged in the data set proposed in the previous section.

## 3.3 Proposed Convolutional Neural Network Model

The Keras API supports several Deep Learning models through Application topology, as well as providing previously trained models on ImageNet data set, as Xceptron, VGG16,

ResNet50, InceptronV3, or MobileNet. This section presents the MobileNet model used to conclude the proposed work, as well it is necessary adaptations to get expected solutions and classifications.

As previously described, this work proposed a vision-based vacancy state detection through Convolutional Neural Networks, Background Subtraction, and Embedded Systems. Motivated in provides a solution for vacancy state detection, it was needed produce a robust Deep Learning model that has the features of detecting the state of a parking space, considering changes in lighting and movement in the scenes and also providing the application of this model in a mobile device, as permitted by Internet of Things area.

Firstly, it was necessary to produce a robust Deep Learning model, focused on classify parking vacancies in 'occupied' or 'available' states. Using Convolutional Neural Networks, it is possible to develop a model that differentiates these types of vacancies through Computer Vision. It consists of the acquisition of parking images and processing this data through a Neural Network, which may return the probability that objects of interest compose the input image.

The choose of use a MobileNet comes from their potential application in mobile devices, due to it be inexpensive in computational terms. The MobileNet structure described in literature is built on 28 depthwise separable convolution layers except for the first layer that it fully convolutional.

However, the Keras MobileNet presents 93 layers distributed into 1 Input Layer, 91 Layers divided between sets of Pooling, 2-D Convolution, Depthwise 2-D Convolution, Dropout, BatchNormalization, and ReLU, Furthermore 1 Softmax Layer and 1 Reshape Layer. The proposed model, including the MobileNet structure it is described in Appendix A.

Keras makes this model available bellow, making it possible to include or exclude its last layers through 'include top' param, also making it possible choose the training weights data set through 'weights' param, and making it possible choose the data input shape through 'input shape' param, disposed of in Table 3.3.

Table 3.3: Proposed MobileNet Required Parameters.

| **MobileNet Input Parameter** | **Parameter** |
|---|---|
| Include Top Layers | True |
| Import Weights From | ImageNet |
| Input Shape | (rows, cols, channels) |
| Image Rows | 96 |
| Image Cols | 96 |
| Image Channels | 3 |

This approach re-train the MobileNet through the Transfer Learning and the Fine Tunning methods; after download it models, it was necessary to deactivate the final layers, to include the proposed Convolutional Neural Network. This process proposes the exclusion of the last 5 layers of this model based on remove the final dropout, convolution, activation and reshape layer, keeping the last MobileNet reshape layer providing an entry parameter for the newest proposed layers. Thus, it was chosen two other layers to ensure the correct model finalization: a fully connected layer with the ReLU activation function and a fully connected layer with a Softmax activation function, disposed of in Table 3.4.

Table 3.4: Proposed Convolutional Neural Network.

| **Proposed Layer** | **Parameter** | **Value** |
|---|---|---|
| MobileNet | Last 5 Layers Trainable | False |
| Dense | Input Size | 512 |
| | Kernel Regularizer L2 | 0.0001 |
| | Activity Regularizer L1 | 0.0001 |
| | Activation Function | ReLU |
| Dense | Output Size | 3 |
| | Activation Function | Softmax |

To prevent overfitting issues, the added hidden layer receives 512 input values, a value described by the , furthermore it uses the technique of kernel regularization, apply L2

and L1 regularizer both assuming 0.0001 learning rate metric, based on the regularization of weights and outputs from proposed network. The proposed model have several other configure set, presenting the current model as the suitable to manage it parking domain.

Aiming to replicates the model on 2 available GPUs, it was used the 'multi gpu model' function, also disposed of by Keras API. This provides in running different parts of the same model on different devices, and for that, it is needed that the model has parallel architecture. It can be achieved by using a TensorFlow function named 'device', to include and merge the different parts of the parallel model.

Moreover, the 'multi gpu model' method consists into divide the model's inputs into multiple mini-batches, apply a model copy on each mini-batch being executed on a dedicated GPU, and concatenate the results on CPU into one batch. As described into literature, the recommended number of batches must to be set up according to the number of GPU used devices and its image size, in this case, the batch size its 64 and the number of mini-batches of 32 samples.

With the assurance that the model can be paralleled, the next step is to format the way the model will be compiled. It comprises the Keras 'compile' function, and requires an optimizer, a loss function, and an evaluation metric, as described bellow. As described in the previous section, Adam optimizes the model based on the first and second order gradient, and the 'categorical cross entropy' loss function provides a multi-class output and accuracy metric. The proposed work determines a learning rate for Adam of 0.0001.

After the model was developed, it is needed to choose the Keras 'fit generator' function, which enables choose the number of epochs, steps for each epoch, steps for validation data, data samples, and callback functions, as shown in bellow. Steps are denoted by the number of samples in data set divided by the defined data set batch, in this case, the chosen batch size comprises the value 64. The training process its disposed of in Table 3.5.

The callbacks proposed into this work are available on Keras by: 'ModelCheckpoint' (MC), 'ReduceLROnPlateau' (RLROP), 'EarlyStopping' (ES) and 'CsvLogger' (CSVL) methods, providing control on the training process stopping it when the metrics have to

stop to improving, in order to improve weights update, reducing and minimizing the loss rate. Moreover, The CSVLogger saves the model training progress.

Table 3.5: Proposed Model Training.

| **Model Training Parameter** | **Parameter Description** | **Value** |
|---|---|---|
| Training Data | Train Path | - |
| Steps Per Epoch | Training Samples / Training Bach Size | 647 |
| Validation data | Validation Path | - |
| Validation Steps | Validation Samples / Validation Bach Size | 163 |
| Epochs | Number of Epochs | 30 |
| Callbacks | MC, RLROP, and ES | - |

The proposed ModelCheckpoint saves the model after every epoch that contains the formatting options, as save the best model only in the epoch that the current monitor value presents minimization. The proposed ReduceLROnPlateau reduces the learning rate when the metric validation loss has stopped improving, with a factor of 0.2, and patience of 4 epochs. Furthermore the proposed EarlyStopping also stopping the train when the validation loss has stopped improving, under the same values to factor and patience. The proposed structure is shown in Table 3.6.

Table 3.6: Proposed Checkpoints for training.

| **Checkpoint** | **Parameter** | **Value** |
|---|---|---|
| ModelCheckpoint | Monitored Metric | Validation Loss |
| | Save Best Model Only | True |
| | Mode | Minimum |
| Reduce LR on Plateau | Monitored Metric | Validation Loss |
| | Factor | 0.2 |
| | Patience | 4 |
| | Minimum Learning Rate | 0.0001 |
| Early Stopping | Monitored Metric | Validation Loss |
| | Factor | 0.2 |
| | Patience | 4 |
| | Mode | Minimum |

In order to evaluate the proposed model, this work proposed two evaluation approaches: validation and test. Both of these approaches considering four scenarios: a) predictions for the input samples from data generation, b) model evaluation on data generator, c) classification report according to evaluation metrics, and d) plotting of confusion

matrix Furthermore the history demonstration presents how the model behaves during the training process.

The history demonstration belongs to Keras 'History' callback method and provides information about the model behavior on each training epoch, highlighting two validation and test evaluation metrics: accuracy and loss functions. This method memorizes this information, and it is possible to plot a graph to evidence these values posteriorly.

The model evaluation on the data generator belongs to Keras 'predict generator' method and returns a list of predictions for a single batch of samples. Through this prediction list, it is possible to obtain the highest value and rank it according to the expected results. The model evaluation on the data generator belongs to Keras 'evaluate generator' method, and test the model on a single batch of samples, showing accuracy and loss function results.

The classification report belongs to the Sci-Kit Learn API metrics and build a text report through the model evaluation, showing the results of the leading classification metrics. This work considers as main classification metrics the precision, recall, and f1-score. Just like the previous method, the confusion matrix belongs to Sci-Kit Learn API metrics, and compute the evaluation of the classification accuracy. These four presented methods it is disposed of in Appendix B for testing evaluation.

Through the proposed model, it is possible to recognize positive and negative images with significant accuracy and loss rates, as demonstrated in the next chapter. Also, Figure 3.6 shows how is the expected model operation. When an image input is provided, the model process the input features, classify it according to the learned features and produce a probability result, that allows classifying its input data as an existing classification.

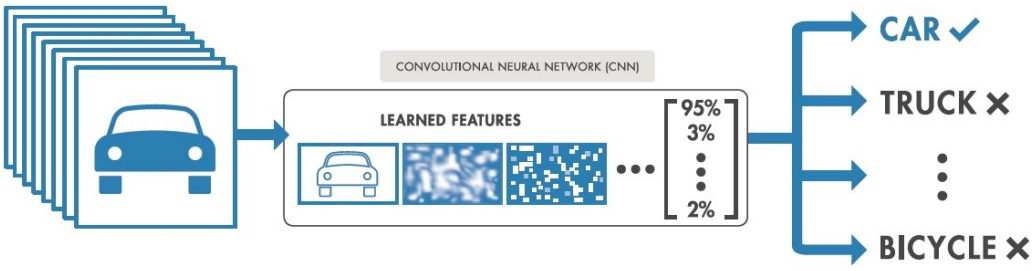


Figure 3.6: Convolutional Neural Network behaviors, reproduced from [27]

In this case, Figure 3.6 provides a positive example that ensures, according to the model, 95 percent of chances of the input values belong to the classification 'Car'. Therefore, this model proposes a default input shape named as 'channels last' data format (height, width, channels) of size 128x128x3 as recommended in the Keras MobileNet structure.

## 3.4 Proposed Vacancy State Model

To solve the parking spot status detection, this works propose a model that integrates Convolutional Neural Networks models, Background Subtraction models, and Embedded Systems solutions. With the development of the data set, and the model considering it completed training, it is needed to develop a solution that integrates these subjects, allowing the application of the detection model in complex scenes and ensuring its operation through a mobile device.

These work propose, as a third step, the image data acquisition process. This process consists of to develop a vision-based system focused on extract, process, and report parking vacancy information. The first challenge consists in a no marked park inside of the ESTIG's space, and it situation enables each park user to choose the parking vacancy position, according to though common sense to keep as many vacancies as possible available.

The second presented challenge about this park is that the acquisition device is set up to detect as many visibility suitable vacancies as possible, witch evidence the inclination of

each vacancy clearer, forced by the camera position. It is considered a challenge based on the detection of each spot cause visibility between each vacancy to be reduced, implying into researches to provide the most suitable vacancy positions for space.

As a first and most suitable option, it is the manual vacancy position detection. It enables the park's manager to include the number of vacancies manually as space allows, ensuring the when it is founded a park's user that does not park properly, the spaces occupied by this user will be established as occupied spot state. These process, allows the parking management to contact the user, and also enable the parking for other vehicles to disable into this occupied vacancies, through the criterion of lack of space.

In the literature, it is common to detect parking vacancies the uses of a region-based Convolutional Neural Network, known as the RCNN model, to extract region proposals to identify the position of the interest object into the input image, in this case, a vacancy spot. In that way, each one of its proposals regions it is computing though CNN features to classify its regions [55].

This proposal has three most significant requirements that can become possible issues: the first one, it is that this process takes a considerable amount of time to classify all these founded regions, once it process is based on Corner [18] or Selective Search [56] detection methods; second problem it is that this solution can not be implemented in real-time, based on the first requirement; and third the process to detect corner positions does not be able to learn, enables not suitable region proposals candidates [55].

To provide improvements over the RCNN model, it was developed the Fast-RCNN and the Faster-RCNN structures, that instead of the region proposals, it enables the creation of a convolutional feature map, predicting the region proposals, proposing a less computationally costly solution [57] [58].

With the motivation to apply some of these procedures within this work, it is proposed to develop a vacancy position detector method that allows the search of vacancies without losing the characteristic of real-time operation, as being an activity apart from this work proposes. Thus, the second method proposed to detect vacancy position comes from the use of Corner and Selective Search methods as the first process after the data acquisition,

introducing a semantic segmentation process.

In order to detect the parking vacancy position, the first approach was after the first real-time data acquisition, and a Corner detection method it is applied, fitting to propose several regions of interest (ROI). Through this ROI's, it was possible to apply the CNN developed model, which classifies each one of its regions, considering as positive the car and motorcycle detection. In this way, this work provides two essential features: the detection of corners and the Convolutional Neural Network model proposed.

In parallel, it was applied to the Selective Search detection method, fitting to propose different regions of interest, to produce a different result from the Corner detection method. Figure 3.7 and 3.8 shown both processes, and through these methods, it was studied the most assertive process that does not take high processing time, and its results it is disposed into the next chapter.

Figure 3.7: Corner Detection and bounding box drawing process.

In this way, through the parking vacancy detection process, the system can continue the primary processing, there is vacancy status detection. Therefore, in order to presents a robust method, also was proposed, as the third structure, the corner fitting method, that makes the detection of corners and produce a region-based where vehicles are most likely to be parked. Also, it was chose a vacancy interval of sizes to produce Figure 3.7 and Figure 3.8.

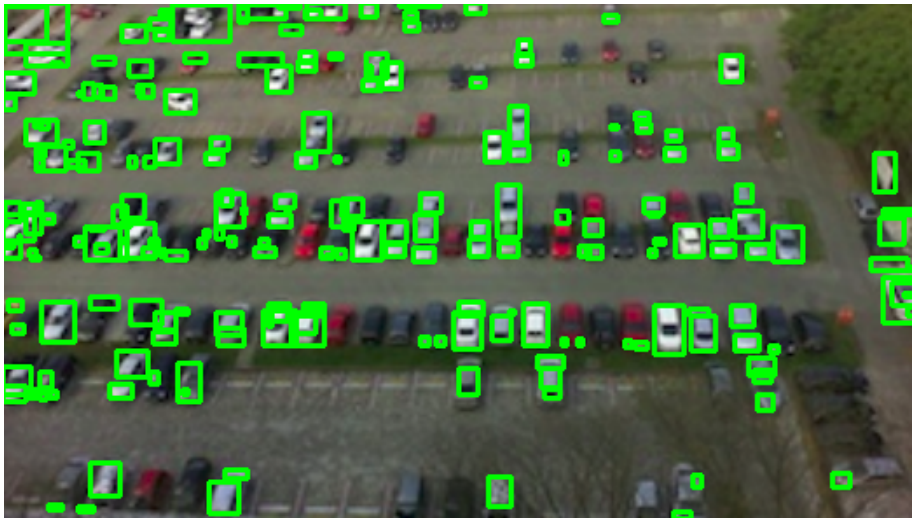

Figure 3.8: Selective Search and bounding box drawing process.

For this approach it was used the Linear Regression between these corners, providing a possible line of parking into the input image. The only requirement of its process it is that some parking vacancies have to be occupied in the time of the processing, this result it is disposed into the next chapter.

In order to crowd these processes, this works proposes two described approaches: a) the manual vacancy state detection, and b) the semi-manual vacancy state detection. And as previously described, after the vacancy position detection, the main processes it was approached: the vacancy state detection. The biggest challenge into this problem is the computational power consumption, which usually reaches high levels, hindering its integration into mobile devices. The model choose was though according to the reduction of this computational consumption and the vacancy state model proposes complementing this goal.

It is possible to reduce computational consumption through computational practices, and as the most relevant chosen practice, it is the Background Subtraction method. This method provides autonomy to apply into several non-controlled scenarios considering as essential features: illumination and movement changes. It allows the system to automatic classification the most indicated moments to use the Convolutional Neural Network model proposed, aiming to detect vacancies status.

As previously described, there are two proposed vacancy status classifications: 'occupied spot' and 'available spot'. Once the model always updating the vacancy status every second, the computational power has reached high values, making that the model classification takes longer to complete and presents the solution. The proposed model describes the use of Background Subtraction only to make the CNN model classify the region of interest when there is movement, as shown in Figure 3.9.

Figure 3.9: Movement detection through Background Subtraction method, adapted from [59]

In that way, the proposed system receives as input a parking image, and for each vacancy spot, it is detected, as the first place, what is the vacancy classification. This process of vacancy classification it is provided by the Convolutional Neural Network proposed, classifying into 'occupied spots', the spots that have ranked as positive objects (car or motorcycle classes), and the 'available spots' are there it is classified as negative objects.

Once the vacancy classification is completed, this result is reported to the service, and the next classification process will only start when a movement interval is detected into the vacancy spot. This movement interval comprises five consecutive seconds or five

successive frames (for frames per second rate equals to 1), named by this work as vacancy memory, based on a mean for each vehicle park process.

This vacancy memory comprises how much movement it is detected for each vacancy spot, being considered true when the value access five consecutive changes detected. Therefore, the number of memories matches the amount of vacancies spots detected. Thus, as described in Figure 3.10, the vacancy state solution comprises the vacancy detection, the classification of a vacancy spot through CNN model classification, and the interval of checks that the system implements. Through this approach, the system does not need sensors and can be applied to a mobile device.

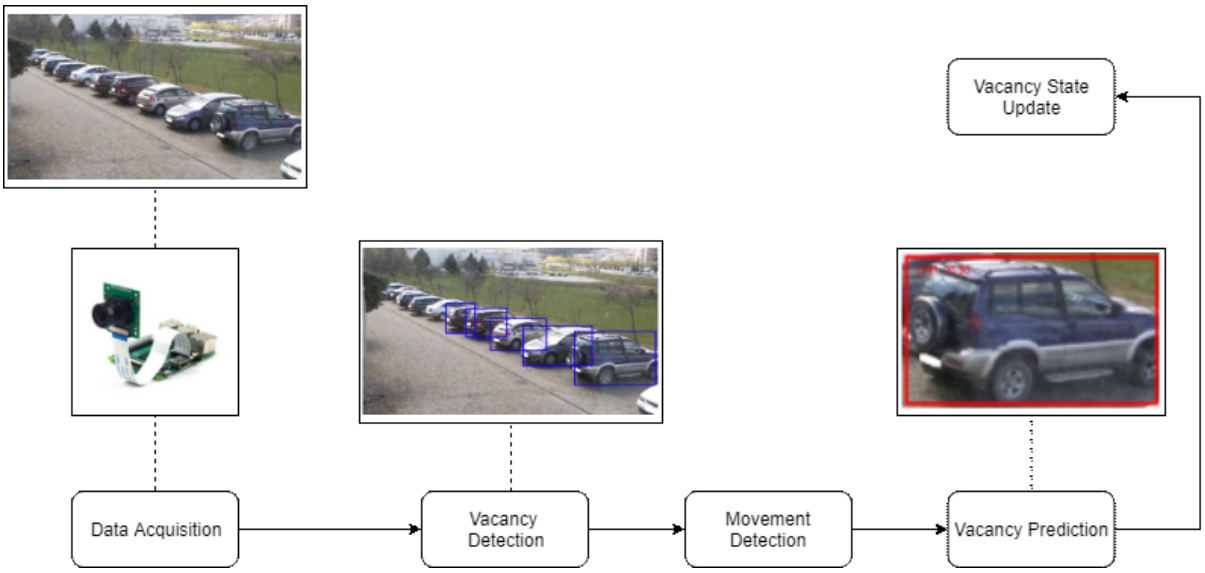


Figure 3.10: Proposed vacancy state detection architecture.

It process of memory movement detection integrated with the vacancy detection enables that the system does not always be looking for an update each vacancy status, providing as the main resource the computational cost reduction. Thus, it provides the integration with the Internet of Things and Embedded Systems concepts, allowing the process to be applied into a mobile device.

# Chapter 4

# Analysis and Result's Discussion

Chapter presents the proposed model feature analysis, presenting comparisons between works found in the literature review.

## 4.1 Proposed Vacancy State Model Analysis

Designed to construct a real-time vision-based application to the proposed Convolutional Neural Network, the vacancy state model was implemented in a real scenario, and comprises five determinate sections: the vacancy detection, the movement detection, the memory vacancy feature, the CNN model application, and the complete vacancy detection application, including its physical structure.

As first impression, the central propose of this work it is determine the park vacancies status, considering occupied spots or available spots through CNN, Background Subtraction, and presents support to Embedded Systems. Moreover, the proposed vacancy detection methods complements this work to improve the chance of proper system operation. Therefore, considering the first section, it was preliminary study two essential types of vacancy detection: the manual detection and the automatic detection.

### 4.1.1 Manual Detection of Vacancy Limits

In terms of dispose manual detection results, it was provided as park manager first competency, the demarcation of parking spaces available in the observed scenario. Its process comprises the uses of the OpenCV Library to allow the system to be able to capture the amount of vacancies included by the user in real time as the user requests. It provides the needed autonomy to the system be prepared for the following processes.

Figure 4.1 presents the vacancy bounding box selection and the system real application of its selected vacancy, as complement the manual vacancy detection method it is shown the following described processes:

- In order to select a bounding box it is needed to the user position the input device over the desired space and select the area;
- In order to confirm the desired area, the user must press the enter key;
- In order to correct the desired area, the user must to return to the step one; and
- In order to terminate the vacancy selection action, the user must to press the enter key.

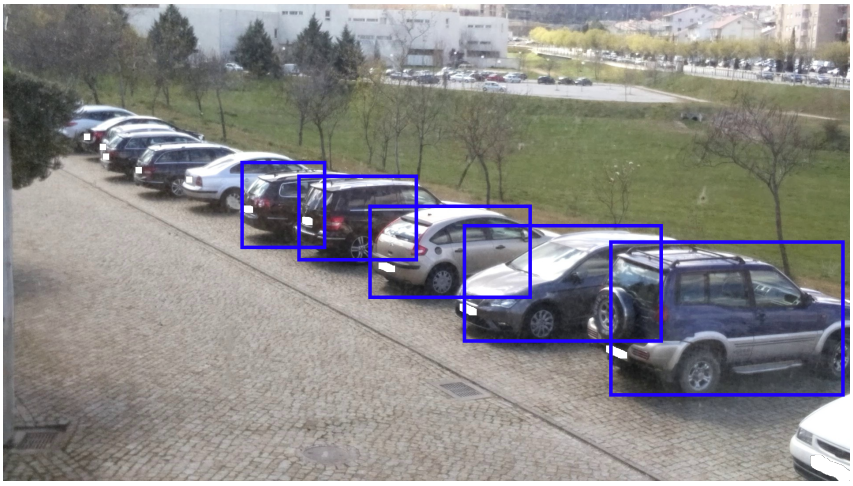

Figure 4.1: Manual vacancy detection.

### 4.1.2 Automatic Detection of Vacancy Limits

In terms of dispose of results of the automatic detection case study, as previously described, it was proposed the Corner Detection, the Linear Regression [60], and the Selective Search methods. Both methods aims to study efficient ways to implement automatic detection of the vacancy position as complement of the proposed status detection domain, and were based on Region Convolutional Neural Networks modeling structure.

As the first method studied, the Corner Detection presents important features, it comprises the method of extract interest points on way to modeling the observed scenario, may indicate motion detection or object recognition. The corner detection application in this work proposes a solution to recognize possible vacancy position through scenario interest points, as described in Figure 4.2, and can be an input parameter for the desired procedure.

Thus, Figure 4.2 shows the original scenarios and its obtained vacancy detection results, considering the bounding box inclusion, in order to represent positive classified regions. In other words, each designed region describe a position where the proposed CNN model predict as positive classification, not showing in this image the negative classifications.

Figure 4.2: Automatic vacancy detection through Corner Detection method.

Looking at the Figure 4.2 it is possible to realize that the large margin of vacancies

detected is according to the specification belonging to one of the positive classifications, described as car or motorcycle objects. Although, this figure also describes its possible issues and challenges: false negative objects, as the box marked as blue in the top right image corner.

It occurs due to the large colour change between image background and foreground, indicating the position of an object, along with the failure to recognize the object through the model due to the distance the camera is located. In other words, it was detected a corner, a region is drawn over its object, and this region it is erroneously predicted as positive object. Therefore, its possible realize that the most lighter regions presents more efficient results, precisely because of its easy identification compared to the image background.

In order to complement the first method, the proposed objective for Linear Regression it is considering the detected corners as input parameters, allowing the system to plot an average between the arranged points, according to the Figure 4.3, formalizing an area of positive predicted objects. Due this process, it was possible realize that the region of interest can be considered as solution when analyzed in order to comprises all available vacancy positions.

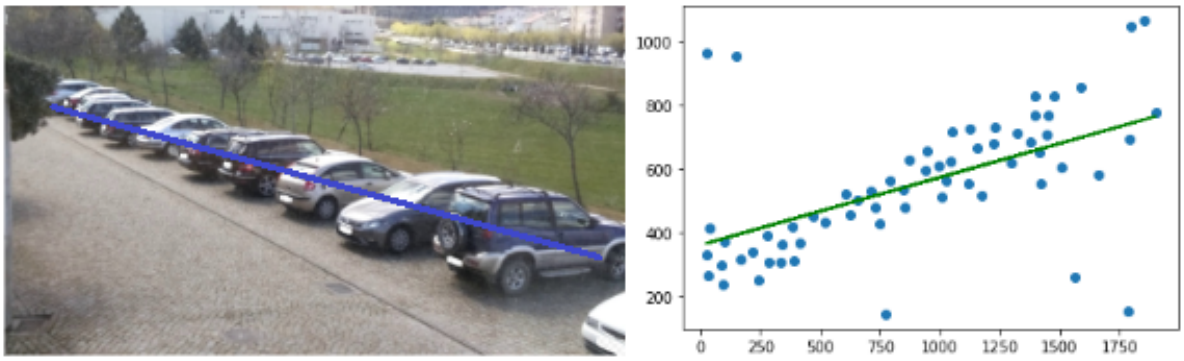


Figure 4.3: Vacancy's mean region through Corner Detection and Linear Regression methods.

The demarcation of parking spaces available in the observed scenario comprises the uses of the OpenCV Library to allow the system to be able to capture the amount of

vacancies included by the user in real time as the user requests. It provides the needed autonomy to the system be prepared for the following processes. In that case, the study of automatic detection of vacancy positions proposes a solution based on no human-iteration, based on reaches lower loss evaluation levels.

Furthermore, the Selective Search method becomes as an alternative to Corner Detection based on computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility. This method could not present assertive objects locations, due to each specific object usually contains at least 2 segmented parts, thus, the regions predicted based on its method provide an analysis of objects that have very high overlap with actual objects.

The third solution proposed to region detection by this work, the Selective Search method and its results are disposed in Figure 4.4, describing the most relevant regions predicted as positive objects in the observed domain. As it is possible visualize, some purposed regions are determined over other regions, allowing the process of regions intersection, creating a bigger one region based on each positive classification.

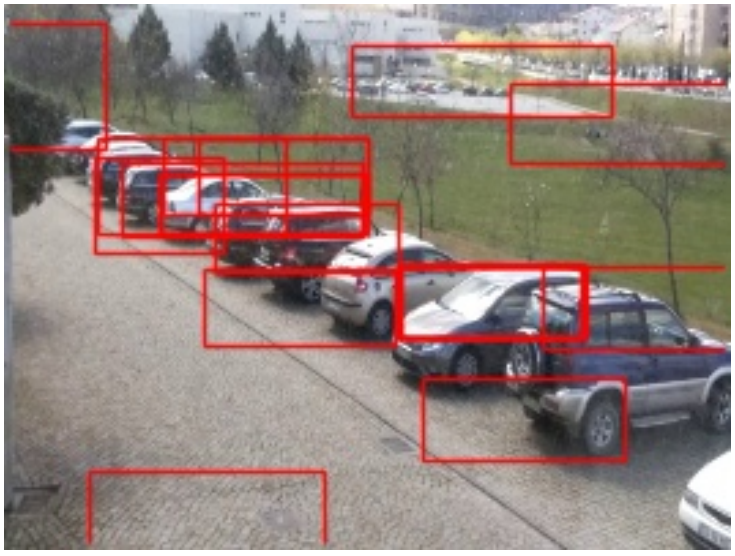

Figure 4.4: Automatic Vacancy Detection through Selective Search method.

Therefore, also it is possible realize that these intersected regions take positions in front of other vacancies, making it impossible to separate and delimited each vacancy. This

challenge occurs due to the Selective Search, used in Region CNN models it is indicate in domains which the object of interest does not is presented in right angles. In terms of evaluation, Table 4.1 presents a final brief discussion of the obtained results for vacancy limits detection.

Table 4.1: Evaluation

| **Proposed Method** | **Obtained Features** | **Seconds** |
|---|---|---|
| Manual Detection | Individual vacancies | - |
| Corner Detection | Individual vacancies | 15 |
| Corner + Linear Regression | Region of interest of all vacancies | 20 |
| Selective Search Detection | Individual vacancies | 40 |

As described in Table 4.1, the manual vacancy limits detection proposes a faster and suitable solution that needs to set up each vacancy limit as the first time, and update it vacancy over demand. According to the obtained results for automatic vacancy limits detection, the Corner Detection presents the shortest time required, and when compared the Selective Search method requires an increase of 25 seconds. As described, this method occurs at least once in the beginning of the proposed vacancy state detection method, does not removing the proposed real time feature.

### 4.1.3 Movement Detection Approach

The movement detection process comprises the application on Background Subtraction Method based on Gaussian Distribution Mixture aiming to reduce the computational power consumption through the evaluation of vacancy status alteration candidates. Due to this evaluation, the system has autonomy to consider or not consider a possible vacancy status update, monitoring movement in each vacancy position and comprising the time metric of 5 seconds of movement to being considered a possible candidate. Figure 4.5 describes a real application of a vacancy movement analysis.

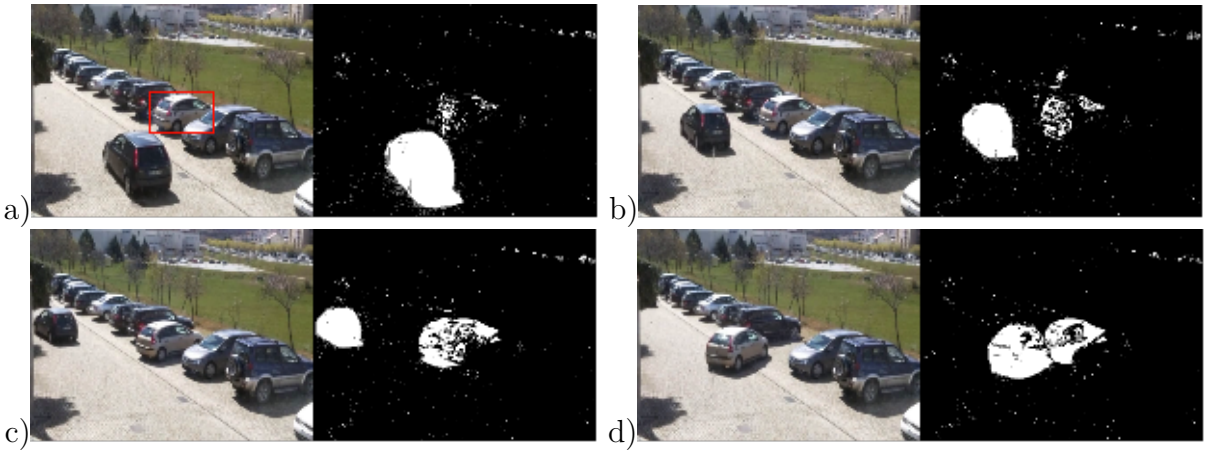


Figure 4.5: a) describes a vacancy with little movement, b) increasing in movement detection at vacancy c) increasing in movement detection at vacancy, and c) decrease in movement detection at the vacancy

Considering the vacancy marked as red (occupied spot) in Figure 4.5, it was possible analyse that the in following figure sections occurs a event characterized in this proposed work as large amount of movement of 5 seconds. From that process, the proposed CNN model recognize if it vacancy has a positive or negative classification, and through this result, the vacancy it is marked as green, in other words, the vacancy state just changed to available spot.

It method allows the system autonomy in terms of enables the model prediction process, filtering the demand of computational consumption to actuate only in real vacancy status update situations, ensuring the vacancy status detector method suitable to embedded solutions, considering a threshold of movement over then 1500 pixels.

### 4.1.4 Convolutional Neural Network Model

In order to represent the results obtained through the proposed Convolutional Neural Network, this section is presented into three steps: the proposed data set, the CNN model structure, and the obtained model results. In what consists of the data set development, it is essential to ensure the balanced it is the set. In this case, as previously described, the data set consists into at least two resources, being the first, the Open Images Dataset

3.0, that contains nine millions of images for train, and about 600 classes, a strongly recommended data set by the literature, and being the second resource, the acquisition of open images through the internet search.

The proposed data set currently contains 41435 images for training patterns, 10491 images for validation patterns, and 10814 images for test patterns, being structured according to the Cross-Validation method, that provides 70 percent of data for training, 15 percent for validation, considering as resources obtained through the training set, and 15 percent of data for test, considering the test data does not belong to the training set. Also, the number of images from each class, it is balanced according to the other classifications, having an average amount of 3000 of positive images.

The proposed model structure is adapted to receive as train input the described data set, and according to the moment of the training process, it was applied significant methods in order to avoid overfitting issues and ensure the loss function presents no significant increase. In the first place, the data augmentation method is provided on the way to generate several possibilities of transformations into the data set, aiming to ensure several possible cases of adaptation of these input images and the environment changes. The data augmentation it is instantiated before the training process, and the generates images remain in the model's training memory, never being digitally saved.

As described, the training process presents four other methods to avoid overfitting issues as well as to maintain the loss range closer to 0, as Kernel Regularization by L2 and L1 paradigms, the Dropout method, the Early Stopping callback, and the Reduce on Plateau callback. These methods provided that the chosen metric, in this case, loss validation metric, does not reach large values keeping as close as possible to 0, and the loss function produced into this work assumes the graphical representation disposed of in Figure 4.6.

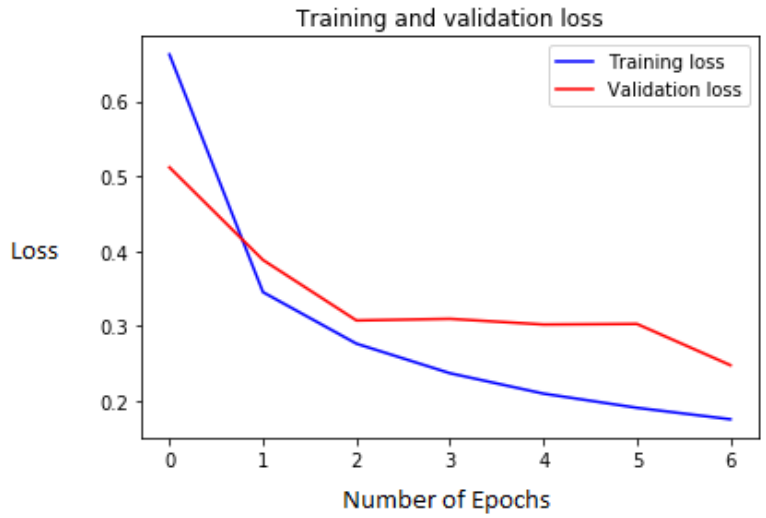


Figure 4.6: Training and Validation loss values.

This graphical representation of loss metric, shown that according to the number of epochs, the model loss evaluation presents two categories, the training loss, and the validation loss metrics. It is essential to realize that the training loss comprises it metric evaluation during uses of the train data set for the training process, and the validation loss metric presents its results to the validation data set, also for the training process.

According to Figure 4.6, it is possible to realize that both the training loss metric reaches large values in the first place, over 0.6, and it finishes the training lower than 0.2, representing 0.4 rates of improvements, and presents again about 66 percent in the training loss evaluation. This process of training loss evaluation usually stay lower if the model structure and the data set structure are appropriate to the designed domain, as large training loss values demonstrate the percentage of error about each 100 rated images.

Therefore, the validation loss metric reaches 0.5 as the first value, and finish its process about 0.25, representing 0.25 rates of improvements, and presenting again about 50 percent in the validation loss evaluation. Furthermore the model validation loss presents a significant feature, in this epoch intervals, the loss never reaches higher values than already seen. This feature demonstrates the Early Stopping, and the Reduce on Plateau

functionalities does not allow the gradual increase of the metric.

The loss metric presents the useful standards, and as closer to 0, the model lies more suitable to the designed domain. Table 7 shows the loss metric more accurately, and Appendix C demonstrate more specific details for each training epoch. Once the training loss allows the evolution of a more suitable model, the accuracy metric presents the model evaluation based on its hits.

It comprises the characteristic of as closer to 1, and the model presents a greater chance of producing true positive results. Figure 4.7 shows the training and the validation accuracy metrics, evaluated according to the training epochs, and as graphical represented, the training accuracy assumes values lower than 0.88 as the first epoch, and finish it higher than 0.96, presenting almost 10 percent of accuracy improvements.

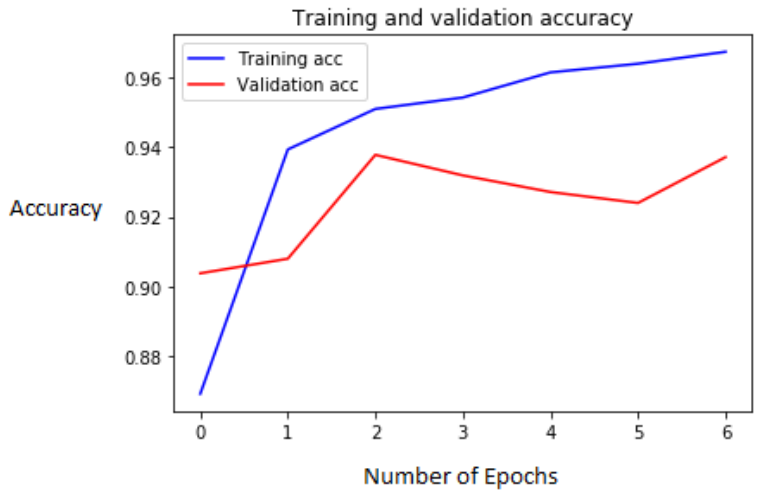


Figure 4.7: Train and Validation Accuracy values.

Furthermore, the validation accuracy metric starts with a rate higher than 0.9 and finishes its process about 0.94, presenting again about 4 percent of accuracy improvements. Its accuracy evaluation graph demonstrates that the validation accuracy is in an acceptable interval, moreover, could be better crafted using methods for accuracy improvements. This proposed work does not present a solution for better accuracy evaluations, being focused on the acquisition of suitable intervals for loss evaluation. Table 4.2 describes more accurately these metrics. Therefore it was possible to compare these results with works

found in the literature to present solutions into the accuracy evaluation.

| **Metrics** | **Loss** | **Accuracy** |
|---|---|---|
| Training | 0.1757 | 0.9673 |
| Validation | 0.2479 | 0.9371 |

Table 4.2: Training and Validation Metrics.

In terms of evaluations through comparative results, in terms of validation evaluation, it was used the Keras API 'predict generator' method, providing the loss and accuracy evaluation, through a randomly set of images based on the validation data set. It process comprises the score evaluation into ten steps, during 12 seconds per stage, according to the used system, and results in 0.21001765 for loss evaluation and 0.96009375 for accuracy evaluation.

In terms of evaluations through comparative results, this work proposes a brief confrontation between some significant models proposed in the literature and the proposed model in this work. Table 4.3 describes three relevant works found into the literature, that represent the use of three different CNN, and it is possible realize the MobileNet not compete in terms of accuracy but presents significant results in terms of networks indicated for mobile devices.

Table 4.3: Model Comparision

| **Model Title** | **Description** | **Obtained Accuracy** |
|---|---|---|
| Parking-stall vacancy indicator system, based on deep convolutional neural networks [45]. | VGGNet-F and Manual Detection of Vacancy Limits | 99% |
| Car parking occupancy detection using smart camera networks and deep learning [7]. | AllexNet and Manual Detection of Vacancy Limits | 90% |
| Vacant Parking Lot Information System Using Transfer Learning and IoT [47]. | MobileNet and Manual Detection of Vacancy Limits | 88% |
| Proposed Model. | MobileNet and Study of Vacancy Limits Detection methods | 94% |

According to this evaluation, the training process presents other important metrics, and it is disposed of in Tables 4.4, 4.5, and Figures 4.8 and 4.9. These metrics are described

as loss, accuracy, precision, recall, f1-score, and confusion matrix rates for validation and test evaluations.

In terms of validation evaluation, it was used the Keras 'predict generator' method, providing the loss and accuracy evaluation, through a randomly set of images based on the validation data set. It process comprises the score evaluation into ten steps, during 12 seconds per stage, and results in 0.21001765 for loss evaluation and 0.96009375 for accuracy evaluation. That is, for every single batch of input images, it was provided an average of losses and accuracies, demonstrating the validation data set evaluation.

Another set of methods used to model evaluation are disposed of in Table 8, and represents the rates of classification errors and hits, considering True Positive, True Negative, False Positive, and False Negative classifications. As the first method, the precision metric represents the number of positive classifications correctly classified; the recall represents the number of positive expected results correctly classified, and the f1-score represents the average between precision and recall.

In this way, according to the Table 4.4, for positive classifications in the validation evaluation, the precision demonstrates 92 and 95 percent, the recall 93 and 87, and the f1-score 93 and 91 percent. Thus, the precision average reaches 94 percent of correctly positive classifications. At the same time, the recall average reaches 90 percent of correctly expected positive classifications, and the f1-score average reaches 92 percent of harmonic mean between the last metrics.

Considering the smart parking domain, classified as 'available spot', an 'occupied spot' represents the most significant error since a park's user will be indicated by the system to park in a vacancy that is already filled than the indication of a filled vacancy when it is available. In the first case, indicate an occupied vacancy could prejudice the user in terms of time costs when the second case prejudices the parking managing with an inoperative vacancy.

Based on its scenarios, the first most suitable factor for the smart parking domain, it is the precision, once false positive situations may be harmful to the user. At the same time, the second most suitable factor analyses the situation of false negatives, where the

case may be harmful to the manager, providing to an available vacancy the classification of occupied, being directly related to the recall metric. According to this discussion, the 94 percent mean for precision and the 90 percent mean for recall are best-suited metrics for this work concerning its domain.

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| Car | 0.92 | 0.93 | 0.93 | 2596 |
| Motorcycle | 0.95 | 0.87 | 0.91 | 1768 |
| Negative | 0.94 | 0.96 | 0.95 | 6127 |
| | | | | |
| Micro AVG | 0.94 | 0.94 | 0.94 | 10491 |
| Micro AVG | 0.94 | 0.92 | 0.93 | 10491 |
| Micro AVG | 0.94 | 0.94 | 0.94 | 10491 |

Table 4.4: Validation evaluation metrics.

Therefore, the model evaluation needs to be performed accordingly to its domain or final objective, providing considerable solutions based on the seriousness of the problem. Moreover, the confusion matrix represents the positive and negative classifications graphically, and the validation confusion matrix it is disposed of in Figure 4.8, considering the true labels and the predicted labels, representing 9831 true positive classifications against 593 false positives and negatives classifications.

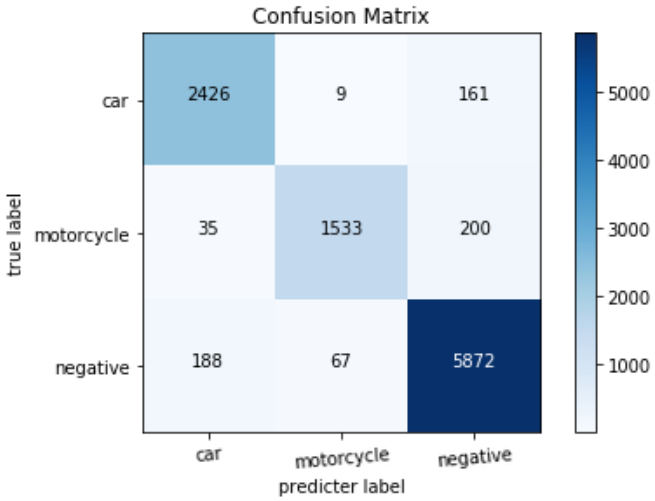


Figure 4.8: Validation's Confusion Matrix.

According to the validation evaluation, the test evaluation provides a new set of metrics based on the test data set, that is, an evaluation over images unknown by the model, evidence to the model a real application. Thus, the loss and the accuracy metrics while analyzed through the test data set, and it is possible to realize that these values approach the validation results, with an absolute difference of 0.03101672 for loss function and 0.02103125 for accuracy function.

Therefore, in terms of suitable metrics according to the designed domain, the precision and recall metrics are disposed of in Table 4.5, demonstrating for positive classifications the values of 0.91 and 0.94 to individual precision rates, and representing 0.94 and 0.88 for recall individual rates. This evaluation set comprises an average of 92.5 percent for precision classifications, and 91 percent for recall evaluation, realizing an absolute difference between validation and test evaluation of 1.5 percent to precision rates and 1 percent to recall rates.

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| Car | 0.91 | 0.94 | 0.92 | 2596 |
| Motorcycle | 0.94 | 0.88 | 0.91 | 1768 |
| Negative | 0.95 | 0.96 | 0.95 | 6127 |
| | | | | |
| Micro AVG | 0.94 | 0.94 | 0.94 | 10814 |
| Micro AVG | 0.94 | 0.92 | 0.93 | 10814 |
| Micro AVG | 0.94 | 0.94 | 0.94 | 10914 |

Table 4.5: Test evaluation metrics.

Thus, the generated confusion matrix for test data set and test evaluation it is disposed of in Figure 4.9, representing 10164 true positives classifications against 650 false positives and negatives classifications. Considering the confusion matrix evaluation for validation and test metrics, it is possible to realize that this metric presents 94,3 percent of correctly classified labels for validation matrix and 93,9 percent of correctly classified labels for test matrix.

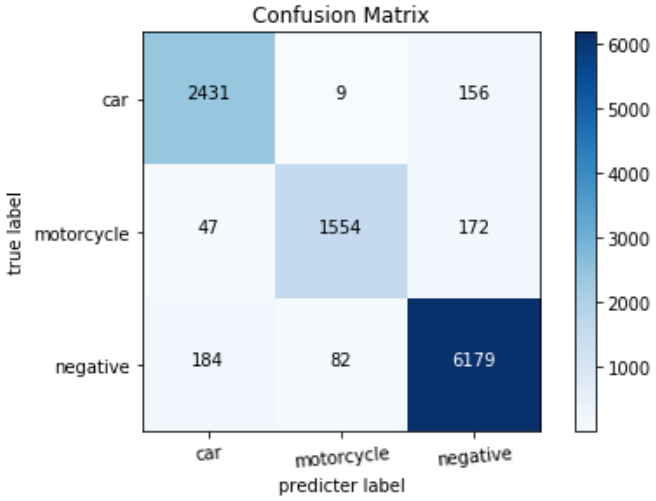


Figure 4.9: Test's Confusion Matrix.

### 4.1.5 Practice Model Evaluation

To demonstrate its discussions in a practice way, Figure 4.10, 4.11 and 4.12 presents the model operation in real visual-based tests. Thus, Figure 4.10 shows the real camera position at ESTIG's park, representing the previously described challenges. In that case, it was disposed three parking spaces acceptable predicted for the model as occupied spots represented in red boxes, and three other positions predicted as available spot represented in green boxes on different backgrounds, obtained through the manual detection of vacancy limits and the proposed CNN model, in order to demonstrate its predicted values.

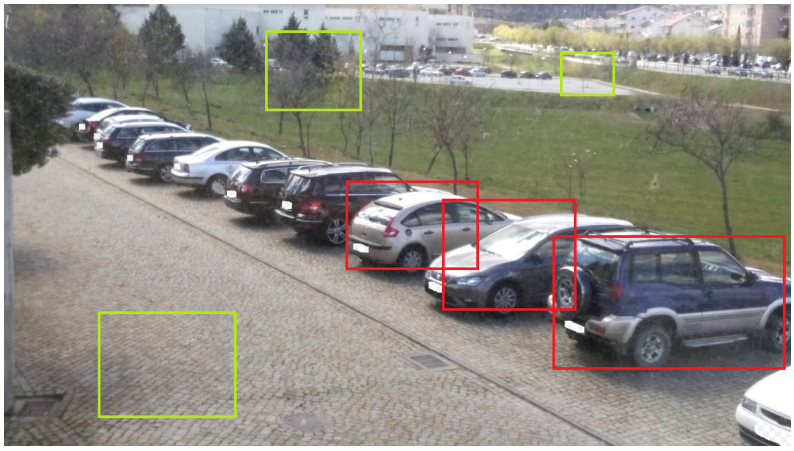

Figure 4.10: Vacancy Detection and its predicted labels.

Furthermore, Figure 4.11 zooming the model predicted accuracy for each previously described position in Figure 4.10, representing a model practice application even considering the difficulties and challenges. Therefore, it is possible analyse in Figure 4.10 a possible available spot marked as blue, in this case, one solution could be uses a depth method to detect if there is a vacancy possibility.


Figure 4.11: Prediction percentages.

As previously described, the challenges comprises the angulation of the disposed vacancies, which could present issues in terms of vehicle or vacancy occlusion, and the no vacancy space demarcations issue, that provides significant difficulties in detecting vacancies according to the users may not park property. Considering these challenges, Figure 4.12 describes a parking space which the is camera slightly positioned to comprises a wide area and spaces without much variation in terms of angle.

In order to filtering the CNN model predictions, the chancing of an object being considered as an object of interest or candidate to positive classification it was evaluated by one different metric, the model's percentage of confidence. It metric proposes to an object being considered as positive classification, its evaluation needs to present values over to 0,7. The source code of its process it was described in Appendix D.

To ensure better solutions in terms of percentage of confidence into the model application, are disposed two significant requirements acquired through the results analysis: the uses of more cameras, and the cameras favorable position study, in order to get a wide

Figure 4.12: Sample of Camera Position.

area as possible, does not allowing situation with vacancy overlapping. Its challenges could be solved based on Figure 4.12, demonstrating an suitable camera position over the parking.

Therefore, this proposed method was studied to get suitable solutions to parking spaces with no vacancy delimitation and in adaptive camera positions, presenting better results when applied using the manual vacancy detection. Also, even though the vacancy state detector model is adapted to reduce the computational expenses, it was needed a structure different of the Rapberry Pi 3B+ used on this work, based on its controlled memory device of 1 GB.

This process could present better solutions if implemented using the Google Coral [61], a USB accessory featuring the Edge TPU that brings ML inference to existing systems, allowing its suitable implementation in Rapberry Pi 4, a newest Raspberry device as substantial difference 4GB of integrated memory.

In order to get around this challenge, the solution applied into this work was the Raspberry Pi 3+, integrated to the the RapCam, get the parking images of a 1 frame per second rate, makes the needed image preprocessing, and report its images to the server, that conclude the solution, reporting to the user. The system uses the Flask Framework and proposes the sharing information of 1 fps through the Base64 conversion method, aiming to doesn't sustain the local images on the device, freeing up disk space.

# Chapter 5

# Conclusions

Chapter present the conclusions of the study conducted in the chapters mentioning the elements that give importance to the work and finally the description of future suggested work.

## 5.1 Final Considerations

In this work it was presented a model to classification of vacancy states in parking spaces, focused on reduce computational power consumption and to be enable to apply in embedded systems. Detailing in special the challenges founded during the development that requires several processes to obtain the final model, since the data acquisition, the Convolutional Neural Network model structure, the model evaluation, the vacancy detection, the middle processing structures and its application on a raspberry pi. Accomplishing the study objectives it was presented the analysis of the model evaluation and its comparison between the literature articles.

As it is possible realize, the section of Analysis and Results Discussion, the proposed model demonstrates a suitable solution to the parking institutions to identifying occupied and available spots. The proposed Convolutional Neural Network model precision presents about 94 percent for almost all evaluation metrics, although the model presents suitable results to the precision and recall metrics, analyzed as the most relevant metrics in order

to avoid the false positives and false negatives situations.

The false positive and false negative situations in the parking domain concluded through this study was considered the situation of an occupied vacancy was characterized as an available vacancy, indicating to the user an erroneously candidate to park, wasting time, and in the same time, the negative situation presents an available vacancy predicted occupied, wasting parking space and money.

The Polytechnic Institute of Braganca disposes its parking space to the exploration and provide the studied challenges through the camera perspective distortion and the no marked positions, allowing study in terms of vacancy position detection, complementing the previously defined objectives, as the Corner Detection, the Linear Regression and the Selective Search methods. With regard to this methods, the study concludes that it is possible implement an integrated or an individual methods for automatic detection in parking situations where the camera position it is favorable.

Thus, the model also presents a solution in term of its lower cost to implementation, as described in the previously sections through the chosed MobileNet Convolutional Neural Network and the Background Subtraction method, focus on detect movement on each vacancy, preventing computational expenses. Also, the model presents a lower cost solution based on its application through Embedded Systems, making the domain an Internet of Things solution. Including, the process of Transfer Learning and stop training which it reaches high values for loss validation, presents a robust method in this domain.

As commonly presented in literature, the solutions in Smart Parking area based on Computer Vision and Embedded Systems keeping the focus on mobile Convolutional Neural Networks, once that solution provides the IoT integration whitin lose so many values in terms of accuracy metrics.

## 5.2 Future Works

Regards the proposed model, it was propose the reformulation of the crated data set including another final classes, aiming to produce suitable and balanced Convolutional

Neural Network input, as the study of CNN model parameters, and updated in terms of the vacancy position detection.

The vacancy position challenge needs to being stable, considering the camera perspective the most relevant proposed increase. This work does not provide solutions in terms of vacancy occlusion by a vehicle in front of the vacancy, being park into the street.

Also, the application of the developed model in a mobile device that provides increases in terms of memory could make that the system just need of a raspberry pi to obtain the expected results, without any server help.

A most ambitious future work provides the integration of this system to a front-end solution, allowing its application in real scenarios through mobile applications to the final user, the driver. For conclude this future work, it was needed a back-end and front-end framework, moreover an special application for mobile devices, proposing a more relevant study.

# Bibliography


[1] R. Giffinger and H. Gudrun, "Smart cities ranking: An effective instrument for the positioning of the cities?", *ACE: architecture, city and environment*, vol. 4, no. 12, pp. 7–26, 2010.

[2] R. Faria, L. Brito, K. Baras, and J. Silva, "Smart mobility: A survey", in *2017 International Conference on Internet of Things for the Global Community (IoTGC)*, IEEE, 2017, pp. 1–8.

[3] A. Koster, F. Koch, and A. L. Bazzan, "Incentivising crowdsourced parking solutions", in *International Workshop on Citizen in Sensor Networks*, Springer, 2013, pp. 36–43.

[4] M. H. Hafezi, A. Ismail, R. A. Al-Mansob, and O. K. Seifabad, "Comparative analysis on bus operation duration light and rush traffic period", *International Journal of Engineering and Technology*, vol. 4, no. 1, p. 97, 2012.

[5] K. Ashton *et al.*, "That 'internet of things' thing", *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[6] M. Fraifer and M. Fernström, "Investigation of smart parking systems and their technologies", in *Thirty Seventh International Conference on Information Systems. IoT Smart City Challenges Applications (ISCA 2016), Dublin, Ireland*, 2016, pp. 1–14.

[7] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning", in *2016 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, 2016, pp. 1212–1217.

[8] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, IEEE, vol. 2, 1999, pp. 246–252.

[9] A. N. Belbachir, *Smart cameras*. Springer, 2010, vol. 2.

[10] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in europe", *Journal of urban technology*, vol. 18, no. 2, pp. 65–82, 2011.

[11] E. Polycarpou, L. Lambrinos, and E. Protopapadakis, "Smart parking solutions for urban areas", in *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, IEEE, 2013, pp. 1–6.

[12] L. Pinheiro and F. Moreira, "Smart parking mobile application", Dissertation, Aveiro University, 2017.

[13] S. E. Shaheen, D. S. Ginley, and G. E. Jabbour, "Organic-based photovoltaics: Toward low-cost power generation", *MRS bulletin*, vol. 30, no. 1, pp. 10–19, 2005.

[14] M. Idris, Y. Leng, E. Tamil, N. Noor, and Z. Razak, " park system: A review of smart parking system and its technology", *Information Technology Journal*, vol. 8, no. 2, pp. 101–113, 2009.

[15] G. N. Hainalkar and M. S. Vanjale, "Smart parking system with pre & post reservation, billing and traffic app", in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, 2017, pp. 500–505.

[16] A. Khanna and R. Anand, "Iot based smart parking system", in *2016 International Conference on Internet of Things and Applications (IOTA)*, IEEE, 2016, pp. 266–270.

[17] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications", *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168–1177, 2008.

[18] R. C. Gonzalez, R. E. Woods, *et al.*, *Digital image processing*, 2002.

[19] R. A. T. d. Costa *et al.*, "Sistema de visão para detecção de pessoas em movimento", 2010.

[20] I. Intel Willow Garage. (2000). Opencv, [Online]. Available: `https://opencv.org`.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[22] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

[23] I. d. Silva, D. H. Spatti, and R. A. Flauzino, "Redes neurais artificiais para engenharia e ciências aplicadas", *São Paulo: Artliber*, vol. 23, no. 5, pp. 33–111, 2010.

[24] T. O. Ayodele, "Types of machine learning algorithms", in *New advances in machine learning*, IntechOpen, 2010.

[25] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial", *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[26] R. Beato and P. Rodrigues, "Smart parking mobile application", Dissertation, Pplytechnic Institute of Braganca, 2017.

[27] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA: 2015, vol. 25.

[28] J. Wang, "Analysis and design of a $k$-winners-take-all model with a single state variable and the heaviside step activation function", *IEEE Transactions on Neural Networks*, vol. 21, no. 9, pp. 1496–1506, 2010.

[29] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function", in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, Citeseer, vol. 181, 1997, p. 185.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] L. S. Feldman, J. Cao, A. Andalib, S. Fraser, and G. M. Fried, "A method to characterize the learning curve for performance of a fundamental laparoscopic simulator task: Defining "learning plateau" and "learning rate"", *Surgery*, vol. 146, no. 2, pp. 381–386, 2009.

[32] K. Yu, W. Xu, and Y. Gong, "Deep learning with kernel regularization for visual recognition", in *Advances in Neural Information Processing Systems*, 2009, pp. 1889–1896.

[33] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, and Q. Yang, "Heterogeneous transfer learning for image classification", in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[34] D. M. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation", 2011.

[35] Y. Sasaki *et al.*, "The truth of the f-measure", *Teach Tutor mater*, vol. 1, no. 5, pp. 1–5, 2007.

[36] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy", *Remote sensing of Environment*, vol. 62, no. 1, pp. 77–89, 1997.

[37] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels", in *Advances in neural information processing systems*, 2018, pp. 8778–8788.

[38] I. Arasaratnam and S. Haykin, "Square-root quadrature kalman filtering", *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2589–2593, 2008.

[39] J. Aneja, A. Deshpande, and A. G. Schwing, "Convolutional image captioning", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5561–5570.

[40] P. P. M. do Nascimento, "Applications of deep learning techniques on nilm", *Diss. Universidade Federal do Rio de Janeiro*, 2016.

[41] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[42] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach", in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 1997, pp. 175–181.

[43] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using kalman-filtering", in *Proceedings of International Conference on recent Advances in Mechatronics*, Citeseer, 1995, pp. 193–199.

[44] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection", in *Video-based surveillance systems*, Springer, 2002, pp. 135–144.

[45] S. Valipour, M. Siam, E. Stroulia, and M. Jagersand, "Parking-stall vacancy indicator system, based on deep convolutional neural networks", in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, 2016, pp. 655–660.

[46] P. R. De Almeida, L. S. Oliveira, A. S. Britto Jr, E. J. Silva Jr, and A. L. Koerich, "Pklot–a robust dataset for parking lot classification", *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937–4949, 2015.

[47] E. K. Jose and S. Veni, "Vacant parking lot information system using transfer learning and iot", *Journal of ICT Research and Applications*, vol. 12, no. 3, pp. 207–218, 2018.

[48] G. van Rossum. (1991). Python, [Online]. Available: `https://www.python.org`.

[49] G. Inc. (2015). Python, [Online]. Available: `https://www.tensorflow.org`.

[50] F. Chollet. (2015). Keras, [Online]. Available: `https://keras.io`.

[51] C. Analytics. (2012). Anaconda, [Online]. Available: `https://www.anaconda.com`.

[52] D. Cournapeau. (2007). Sci-kit learn, [Online]. Available: `https://scikit-learn.org/stable/`.

[53] A. Ronacher. (2010). Flask, [Online]. Available: `https://palletsprojects.com/p/flask/`.

[54] R. P. Foundation. (2012). Raspberry pi, [Online]. Available: `https://www.raspberrypi.org`.

[55] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[56] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition", *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[57] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[58] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.

[59] J. Zhou, "Color separation for background subtraction", 2016.

[60] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[61] G. Inc. (2019). Coral beta, [Online]. Available: `https://coral.withgoogle.com`.

# Appendix A

# Proposed CNN Model

Table A.1: Proposed MobileNet Transfer Learning Model

| **Layer Nº** | **Layer Type** | **Layer Size** | **Layer Size** |
|---|---|---|---|
| 0 | Conv2D | (None, 64, 64, 64) | 2048 |
| 0 | BacthNormalization | (None, 64, 64, 64) | 256 |
| 0 | ReLU | (None, 64, 64, 64) | 0 |
| 0 | ZeroPadding2D | (None, 65, 65, 64) | 0 |
| 0 | DepthwiseConv2D | (None, 32, 32, 64) | 576 |
| 0 | BatchNormalization | (None, 32, 32, 64) | 256 |
| 0 | ReLU | (None, 32, 32, 64) | 0 |
| 0 | Conv2D | (None, 32, 32, 128) | 8192 |
| 0 | BatchNormalization | (None, 32, 32, 128) | 512 |
| 0 | ReLU | (None, 32, 32, 128) | 0 |
| 0 | Conv2D | (None, 32, 32, 128) | 1152 |
| 0 | DepthwiseConv2D | (None, 32, 32, 128) | 512 |
| 0 | BatchNormalization | (None, 32, 32, 128) | 0 |
| 0 | ReLU | (None, 32, 32, 128) | 16384 |
| 0 | Conv2D | (None, 32, 32, 128) | 512 |
| 0 | BatchNormalization | (None, 32, 32, 128) | 256 |
| 0 | ReLU | (None, 32, 32, 128) | 0 |
| 0 | ZeroPadding2D | (None, 33, 33, 128) | 0 |
| 0 | DepthwiseConv2D | (None, 16, 16, 128) | 1152 |
| ... | ... | ... | ... |
| 0 | Pooling2D | (None, 1024) | 0 |
| 0 | Reshape | (None, 1, 1, 1024) | 0 |
| 0 | ReLU | (None, 512) | 512512 |
| 0 | Softmax | (None, 3) | 1539 |

# Appendix B

# Model Evaluation Source Code

```
print('[INFO] Validation Test')
predictions_valid = model.predict_generator(validation_generator,
                        validation_generator.samples/val_batchsize)
y_pred_valid = np.argmax(predictions_valid, axis = 1)

score_valid = model.evaluate_generator(validation_generator,
                        steps = 10, verbose = 1)
print('Scores of Model Evaluate (Validation)')
print('error: ', score_valid[0])
print('accuracy: ', score_valid[1])

print('Classification Report (Validation)')
target_names = ['car', 'motorcycle', 'negative']
print(classification_report(validation_generator.classes,
                y_pred_valid, target_names = target_names))
cm = confusion_matrix(validation_generator.classes,
                y_pred_valid)
```

```
print('Confusion Matrix')
cm_plot_labels = ['car', 'motorcycle', 'negative']
plot_confusion_matrix(cm, cm_plot_labels, title = 'Confusion Matrix')

print('[INFO] Test')
preditctions_test = model.predict_generator(test_generator,
                    test_generator.samples/test_batchsize)
y_pred_test = np.argmax(preditctions_test, axis = 1)

score_test = model.evaluate_generator(test_generator,
                    steps = 10, verbose = 1)
print('Scores of Model Evaluate (Test)')
print('error: ', score_test[0])
print('accuracy: ', score_test[1])

print('Classification Report (Test)')
target_names = ['car', 'motorcycle', 'negative']
print(classification_report(test_generator.classes,
                    y_pred_test, target_names = target_names))
cm = confusion_matrix(test_generator.classes, y_pred_test)

print('Confusion Matrix')
cm_plot_labels = ['car', 'motorcycle', 'negative']
plot_confusion_matrix(cm, cm_plot_labels, title = 'Confusion Matrix')
```

# Appendix C

# Training Process

Table C.1: Training process for train data set

| Epoch | Accuracy | Loss | Learning Rate |
|---|---|---|---|
| 0 | 0.8691203089161432 | 0.6623430979356872 | 1e-04 |
| 1 | 0.9393025220267603 | 0.3453578720287548 | 1e-04 |
| 2 | 0.9509351997147053 | 0.2766191258352226 | 1e-04 |
| 3 | 0.9542174490064623 | 0.23720960743690236 | 1e-04 |
| 4 | 0.9614335706427601 | 0.20998916470221937 | 1e-04 |
| 5 | 0.9639193918185109 | 0.19107530960882618 | 1e-04 |
| 6 | 0.9673464462554856 | 0.17577234012910564 | 1e-04 |

Table C.2: Training process for validation data set.

| Epoch | Validation Accuracy | Validation Loss | Learning Rate |
|---|---|---|---|
| 0 | 0.9037270040987513 | 0.5118281120906145 | 1e-04 |
| 1 | 0.9079210752073206 | 0.38828350548185825 | 1e-04 |
| 2 | 0.9377561719569154 | 0.30757674847138006 | 1e-04 |
| 3 | 0.9318463444857497 | 0.309737091277339 | 1e-04 |
| 4 | 0.9270803545896482 | 0.3021970035831588 | 1e-04 |
| 5 | 0.9239348012582214 | 0.30294722811235575 | 1e-04 |
| 6 | 0.9370889333714613 | 0.24790545452903512 | 1e-04 |

# Appendix D

# Proposed CNN Model Source Code

```
import tensorflow as tf
import keras
from keras.backend import clear_session
from keras.applications import MobileNet
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from keras import regularizers
from keras.utils import multi_gpu_model
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

clear_session()

image_rows = 128
image_cols = 128
image_channels = 3
n_gpus = 2
```

```
print('[INFO] Loading MobileNet Model')
with tf.device('/cpu:0'):
    mobilenet = MobileNet(include_top = True,
                          weights = 'imagenet',
                          input_shape = (image_rows, image_cols,
                                         image_channels),
                          pooling='max')
mobilenet.summary()
for layers in mobilenet.layers[:-5]:
    layer_trainable = False
mobilenet.summary()

print('[INFO] Model Configuration')
model = Sequential()
model.add(mobilenet)
model.add(Dense(512,
                input_shape = (image_rows,image_cols,image_channels),
                kernel_regularizer = regularizers.l2(0.0001),
                activity_regularizer=regularizers.l1(0.0001),
                activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))
try:
    model = multi_gpu_model(model, gpus = n_gpus)
    print("[INFO] Training using multiple GPUs..")
except:
    print("[INFO] Training using single CPU or GPU..")
model.compile(Adam(lr = 0.0001),
              loss = 'categorical_crossentropy',
```

```
                metrics = ['acc'])

%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import Image
from sklearn.metrics import confusion_matrix, classification_report
from keras.preprocessing.image import ImageDataGenerator, image
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
from keras.callbacks import EarlyStopping, CSVLogger
import numpy as np
import keras.utils.np_utils as np_utils
import itertools

print('[INFO] Environment Configuration')
train_batchsize = 64
val_batchsize = 64
test_batchsize = 64
epochs = 30
model_file = 'MobileNet.h5'
log_file = 'mobilenet.log'

print('[INFO] Data Configuration')
train_dir = './data/train'
valid_dir = './data/validation'
test_dir = './data/test'

print('[INFO] Data Augmentation')
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range = 20,
```

```
                                    width_shift_range = 0.2,
                                    height_shift_range = 0.2,
                                    horizontal_flip = True,
                                    fill_mode = 'nearest')
validation_datagen = ImageDataGenerator(rescale = 1./255)
test_datagen = ImageDataGenerator(rescale = 1./255)
train_generator = train_datagen.flow_from_directory(train_dir,
                                target_size = (image_rows, image_cols),
                                batch_size = train_batchsize,
                                class_mode = 'categorical')
val_generator = validation_datagen.flow_from_directory(valid_dir,
                                target_size = (image_rows, image_cols),
                                batch_size = val_batchsize,
                                class_mode = 'categorical',
                                shuffle = False)
test_generator = test_datagen.flow_from_directory(test_dir,
                                target_size = (image_rows, image_cols),
                                batch_size = test_batchsize,
                                class_mode = 'categorical',
                                shuffle = False)

print('[INFO] Callbacks Configuration')
checkpointer = ModelCheckpoint(filepath = model_file,
                                monitor = 'val_loss',
                                verbose = 1,
                                save_best_only = True,
                                mode = 'min')
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                                factor = 0.2,
```

```
                                    patience = 4,
                                    min_lr = 0.0001)
early_stopping = EarlyStopping(monitor='val_loss',
                                    min_delta = 0.2,
                                    patience = 4,
                                    verbose = 1,
                                    mode = 'min')
csv_logger = CSVLogger(log_file, separator = ',', append = True)

print('[INFO] Model Training')
history = model.fit_generator(train_generator,
 steps_per_epoch = train_generator.samples/train_generator.batch_size,
 validation_data = val_generator,
 validation_steps = val_generator.samples/val_generator.batch_size,
 epochs = epochs,
 callbacks = [checkpointer, reduce_lr, early_stopping, csv_logger],
 verbose = 1)
```