

**University Master in Industrial Engineering
Academic Year (2017-2018)**

Master Thesis

Multi-Object Recognition based on Deep Learning applied to Mobile Robots

Jesús Sáez Alegre

Supervisor:

Alejandra Carolina Hernández Silva

Leganés, September 2018

Keywords: Neural Network, Deep Learning, Object Recognition, Robot Navigation, Mobile Robots, Tensorflow.



This work is licensed under Creative Commons **Attribution – Non Commercial – Non Derivatives**

Resumen

Los robots se encuentran cada vez más integrados en el día a día de los humanos. Es por ello que deben ser capaces de interpretar lo que observan, interactuar con las personas y moverse por sus entornos de forma autónoma. Esto ha conducido a diversas líneas de investigación en el campo de la visión, cuyo objetivo es desarrollar el sistema más preciso y rápido de reconocimiento de objetos, siendo Deep Learning la técnica más novedosa.

Las redes neuronales profundas son la base de la inteligencia artificial conocida como Deep Learning. Estas redes pueden analizar gran cantidad de datos y establecer conexiones entre datos de la misma categoría. Deep Learning es una herramienta fundamental para desarrollar redes neuronales capaces de reaccionar ante diferentes entradas, ajustando sus pesos sinápticos, para así mostrar como resultado la salida del sistema más próxima acorde al modelo usado durante el entrenamiento.

El objetivo de este trabajo es analizar el comportamiento, la viabilidad y la versatilidad de los algoritmos más avanzados, en entornos de interior y embarcados en robots móviles del tipo Turtlebot, reconociendo objetos en tiempo real, así como implementar sistemas que permitan reconocer múltiples objetos en una imagen en el caso de no estar la red preparada para ello. Además, este proyecto busca desarrollar una red neuronal profunda y entrenarla para detectar varios tipos de objetos, bajo las mismas condiciones mencionadas previamente. Este desarrollo es llevado a cabo mediante la librería TensorFlow. El rendimiento de estos sistemas es analizado en diferentes entornos cotidianos, comprobando los objetos que se detectan y la clasificación de los mismos, para así determinar que algoritmo es el más robusto a la hora de ser embarcado en un robot móvil real que sea capaz de trabajar en entornos interiores reales. De esta manera se contribuye a la mejora de las capacidades del robot como son la navegación autónoma, la comprensión de los escenarios y la toma de decisiones.

Abstract

Robots are increasingly integrated in humans' life. Consequently, robots must be able to interpret what they see, interact with humans and move around their environment. This has led to several researches in the vision field, which have as an objective the development of the most accurate and fast system to recognize objects, being Deep Learning the newest technique.

Deep Neural Networks are the basis for the type of artificial intelligence known as Deep Learning. These networks can analyze big amounts of data and they establish connections between the data with the same label. Deep Learning is a fundamental tool for developing neural networks which can react to different inputs, adjusting their synaptic weights, in order to bring as a result the closest output according to the model used during the training.

The aim of this work is to analyze the behavior, viability and versatility of the most advanced algorithms in indoor environments while carried by a mobile platform of type Turtlebot, recognizing objects in real time, as well as implementing systems which enable the multi-object recognition in case of not being the network prepared for it. Besides, this project has also the objective of developing a Deep Neural Network and training it to detect various classes of objects under the same conditions mentioned above. This development is carried out with the coding library facilitated by TensorFlow. The performance of these systems is analyzed in different home scenarios, verifying the different detected objects and their classification, so the most robust system for mobile robots capable of working in real indoor environments can be determined. This way, it contributes to enhance the robot's capacities in terms of autonomous navigation, scene understanding and decision taking.

Contents

Resumen	iii
Abstract	v
1 Introduction	1
1.1 Motivation	2
1.2 Objectives	2
1.3 Scope	3
1.4 Structure	3
2 State of the Art of Image Recognition	5
2.1 Main Challenges on Robotics in Europe	5
2.2 Human vs. Robot vision	7
2.3 Object recognition Strategies	9
2.3.1 Model-Based Method	10
2.3.2 Appearance-Based Method	11
2.3.3 Feature-Based Method	12
2.3.4 Artificial Intelligence	13
2.3.5 Multi-Object and Real-Time recognition	22
3 Proposed System	29
3.1 Part 1: Developing a CNN	30
3.2 Part 2: Multi-object recognition and Indoor tests on mobile robot platform	30
3.3 Experimental Platform Software	30
3.3.1 Python	31
3.3.2 OpenCV	31
3.3.3 TensorFlow	31
3.3.4 ROS	32
4 Designing and Training a Convolutional Neural Network	33
4.1 Design and training module	34
4.1.1 Creating network layers with TensorFlow	36
4.1.2 Making the network functional with TensorFlow	42
4.2 Prediction module	44
4.3 Training classes	45
5 Developing Multi-Object Recognition Systems for Mobile Robots	47
5.1 Selected Models Analysis	48
5.2 Multi-Object implementation	50
5.2.1 Method 1: Image segmentation	50
5.2.2 Method 2: Contours Detection	52
5.3 Mobile Robot Platform integration	54

6	Experimental Results	57
6.1	Experimental Platform Hardware	57
6.1.1	Turtlebot Mobile Platform	58
6.1.2	ASUS RGB-D Camera	58
6.1.3	Base Computer	59
6.2	Work environment	59
6.3	Part 1: CNN Results	59
6.1.1	Training Experiments	59
6.1.2	Training Results	61
6.1.2	Developed CNN testing and discussions	66
6.4	Part 2: Multi-Object System	69
7	Socio-Economic Impact and Budget	77
7.1	Socio-Economic Impact	77
7.2	Budget	78
8	Conclusions and Future Works	81
8.1	Conclusions	81
8.2	Future Works	83
	Bibliography	85

List of Figures

Figure 1. Vision Architecture [12]	8
Figure 2. Deep learning representation process [51]	15
Figure 3. Artificial neuron schema	16
Figure 4. Neural network schema	17
Figure 5. AlexNet Architecture [59]	19
Figure 6. VGGNet Architecture [59]	20
Figure 7. Inception Module from GoogLeNet [35]	21
Figure 8. InceptionV3 architecture	21
Figure 9. Results in MS COCO according to SSD paper [36]	23
Figure 10. Results in COCO according to YOLO [21]	24
Figure 11. R-CNN process	24
Figure 12. IoU formula	25
Figure 13. IoU representation	26
Figure 14. YOLO image processing	26
Figure 15. SSD architecture [36]	27
Figure 16. General system schema	29
Figure 17. CNN recognition system	33
Figure 18. Part 1 schema	34
Figure 19. CNN image classifier diagram	35
Figure 20. Convolutional layer structure	36
Figure 21. Convolution process	37
Figure 22. Max Pooling Example	37
Figure 23. Sum Pooling Example	38
Figure 24. Average Pooling Example	38
Figure 25. Max Pooling and Sum Pooling comparison	38
Figure 26. ReLU application example [7]	39
Figure 27. Fully Connected layer schema	40
Figure 28. CNN building process	41
Figure 29. Training process diagram	42
Figure 30. Part 2 schema	48
Figure 31. Models Output example	49
Figure 32. Segmentation classifying process	51
Figure 33. Segmentation output example	52
Figure 34. Contour Detection Process	52
Figure 35. Contour classifying process	53
Figure 36. Contour detection output example	54
Figure 37. ROS system representation	55
Figure 38. Confusion matrix evaluation [28]	61
Figure 39. 2 classes training progress	62
Figure 40. 3 classes training progress	63
Figure 41. 2 classes networks accuracy results	65
Figure 42. 3 classes networks accuracy results	65
Figure 43. Single object detector test	67
Figure 44. 2 contour classifier tests	68
Figure 45. 2 Segmentation classifier tests	69
Figure 46. Kitchen test	70
Figure 47. Livingroom test	71
Figure 48. Bathroom test	72
Figure 49. Office test	74
Figure 50. Bathroom test	75
Figure 51. Gantt Diagram	78

List of Tables

Table 1. Vision classification [12]	9
Table 2. Comparison between brain and computer [26]	16
Table 3. CNN Hyper-Parameters	36
Table 4. Layers Values	42
Table 5. Survey results	45
Table 6. Networks Characteristics.....	49
Table 7. Functional and software specifications [6]	58
Table 8. Constant CNN parameters	60
Table 9. 2 classes precision and sensitivity results	63
Table 10. 3 classes precision and sensitivity results	64
Table 11. Phases of the project.....	78
Table 12. Material Costs	79
Table 13. Total Budget.....	79

1 Introduction

Nowadays, mobile robots are being increasingly integrated into the industry and household appliance. This integration provokes that robotic systems must interact with their surroundings; understanding what they observe in real time and using that information for higher level actions. Mobile robots are systems that integrate perception of the environment, decision making, action and interaction.

The field of vision is in continuous development, during the last decades, a large number of algorithms have been proposed. This is due to the fact that, at a closer look, "object recognition" is an umbrella term for different algorithms designed for a wide variety of applications, where each application has its specific requirements and constraints [64].

Object recognition in real scenes is one the most challenging problems in computer vision, as it is necessary to deal with difficulties such as viewpoint changes, occlusions, illumination variations, background clutter or sensor noise [24]. Thus, there are several methods for object recognition developed based on different techniques such as feature descriptors, shape descriptors, gradient-based, derivative-based, template matching, etc. Besides, this task gets even more complicated when is combined with multi-object detection and real time recognition.

For this purpose, the recognition of objects it is done using machine learning and deep learning techniques, which enables the use of massive data to train a neural network which can predict the objects that are been shown in an image in a fast and accurate way.

1.1 Motivation

Object detection is a computer technology related to computer vision and image processing that deals with detecting, in digital images and videos, instances of semantic objects of a certain class. The interaction of this vision technology with navigation systems can improve the level of integration of robots in humans' life.

Through this combination, mobile robots can understand the place where they are, and that information can be used to trigger different actions regarding the place where they are located and the objects that can be observed, and thus increased levels of autonomy and independent decision-making.

The state of art technique applied to the field of object recognition is Deep Learning. Deep learning goes one step further than machine learning and uses deep neural network to deal with massive data, and it is still a field to be exploited. There are many different neural networks architectures designed and tested in images datasets, each one with their own characteristics and built for specific tasks. The challenge is to understand how they work, analyze how they behave in real indoor environments and use them to recognize objects in real time. Is not only important to know what you see, but to see it on time, and to see as many things as possible in order to take better actions in the right moment. Therefore, these are the reasons from where the motivation of this project arises.

1.2 Objectives

The two main objectives of this work are to build a neural network based on deep learning, and to define which is the most suitable model for multi-object recognition in real time and applied to indoor environments with a mobile robot. These objectives will be accomplished by fulfilling the following sub-objectives:

- Provide a review of the state-of-the-art in object recognition.
- Make use of the already developed best algorithms for image recognition.
- Build a convolutional neural network and train it to classify various objects
- Develop a module to make the Deep Learning systems suitable for the task of multi-object detection based on segmentation and contour detection.
- Develop the previous algorithms to recognize images on real time through an ASUS camera.
- Integrate the object detection systems under ROS environment.
- Analyze the performance of each system in real conditions using a real mobile robot.

1.3 Scope

The scope of this work focused on real time multi-object recognition based on deep learning applied to indoor environments, is to analyze the state-of-the-art algorithms, YOLO, SSD and Inception, and develop a convolutional neural network in order to detect two objects determined by a survey realized about common objects you look for at home. In order to make the multi-object recognition possible with the Inception module, two techniques are implemented, one based on image segmentation and the other on contour detection.

1.4 Structure

This work is divided in 8 chapters. The structure of this work is made as follows:

- Chapter 2 introduces the importance of robotics nowadays, presents the state-of-the-art related to image recognition systems and techniques.
- Chapter 3 explains the proposed system, introducing the employed software frameworks.
- Chapter 4 describes the implementation of the first part of the thesis, building and training a Convolutional Neural Network.
- Chapter 5 describes the implementation of the second part of the thesis, developing multi-object recognition systems for a mobile robot platform.
- Chapter 6 presents the performed experimental tests, the obtained results, and discussions of this work.
- Chapter 7 provides the budget and socio-economic impact of this project.
- Chapter 8 contains the conclusions and the possible future works that might derive from this work.

2 State of the Art of Image Recognition

Image recognition is an important field of study in robotics, which has experimented big changes in the last decades. It is oriented towards the understanding of the scenarios where the robot is and its better comprehension of it by classifying the different elements that surrounds it.

This second chapter exposes the European research challenges on robotics, explains the concept of vision from the biological and the robotic point of view, followed by a review of the different image recognition techniques and its application to indoor environments, with special detail on Deep Learning, as well as a literature review of works where image recognition is used for specific tasks such as multi-object detection and real-time recognition.

2.1 Main Challenges on Robotics in Europe

Horizon 2020, with nearly 80billion euros invested [18], is the biggest Research and Innovation program in EU. With a special emphasis on excellent science and industrial leadership through a continuous transfer of technology and ideas from labs to the market, this project is meant to drive economic growth, create jobs and remove barriers to create a legitimate market for knowledge, research, and innovation.

Information and Communication Technologies (ICT) are included in the actions of Leadership in Enabling and Industrial Technologies Work Program under Horizon 2020. Six main activity lines have been identified in the ICT program: A new generation of components and systems, advanced computing, future internet, content technologies and information management, robotics, micro and nanoelectronics technologies, Photonics [19]. This project is based the activity line of robotics.

Robotics is a field in continuous development. During 2016, 17 robotics projects have been funded under H2020. These projects cover a wide range of research and innovation lines called Societal Challenges which include logistics, health robotics, autonomous cars, industrial robotics and manufacturing market [8]. This can be translated into an impact in society in terms of increasing efficiency, better living conditions and a high-quality growth in job creations.

SPARC, the partnership for robotics in Europe, is not only the agent in charge of implementing robotics strategy for Europe within Horizon 2020, but also the largest research and innovation program in civilian robotics in the world. Its vision of Robotics Technology in the Strategic Research Agenda (SRA) is that robotics technology will become dominant in the next decade, and that its influence will be in every aspect of home and work, transforming lives practices by raising efficiency and safety levels, creating jobs and enhancing levels of service. Furthermore, as the interaction between humans and robots will grow, this impact will do so [42].

The SPARC has determined Mechatronics, Human-Robot Interaction, Systems Development, Navigation, Perception, and Cognition as key technology areas. Through the Multi-Annual Roadmap (MAR), a detailed guide of strategic and technical characteristics of the European robotics field, SRA connects the technology with the consumer needs. On it, the system abilities that robots can possess are determined, and a classification of robots according to the operation environments is done. The core system abilities listed in the MAR are configurability, adaptability, interaction ability, dependability, motion ability, manipulation ability, perception ability, decisional autonomy and cognitive ability [41].

The robot developed by the group belongs to the category of robots on the ground in indoors surroundings. It must be emphasized that this work is focused on the perception and cognition applied to the area of navigation.

2.2 Human vs. Robot vision

Vision is one of human main faculties. Providing to robots a similar vision as humans have, is one of the main challenges of the artificial intelligence [12].

To understand this issue, first it is needed to understand how human vision works. In its simple form, vision can be explained into four stages [4]:

- **Perception:** The first step of the process is optical. The light goes into the eye through the transparent organs (cornea, aqueous humor, crystallin and vitreous humor), where the image is searched, followed and focused.
- **Transformation:** The luminous energy gets to the retina, where sensorial cells are activated, and the light is transformed into nerve energy.
- **Transmission:** The nerve impulses go along the optic nerve until they reach the cerebral cortex.
- **Interpretation:** In the cerebral cortex the impulses are interpreted, recognized and processes in order to know what it has been seen.

This interpretation task is not trivial; it is the process that consumes more resources in the human brain, occupying almost a 40% of the total cerebral capacity. Many different tasks are carried out during these processes such as recognizing objects, textures, distances between objects, creating 3D images from the images of both eyes, movement direction if it exists, creating clear images from blurry images. All these at an almost instantaneous velocity, which according to Simon Thorpe among the other researches from the Centre de Recherche Cerveau & Cognition of Toulouse, France, it allows to interpret the reality with 150 milliseconds of delay [88].

That a computer can perform all these tasks in such a short space of time is something complicated with today's technology, thus, robot vision is still far away from human vision. Nevertheless, different algorithms are being developed for each of the previously exposed task.

Human vision can be translated into robot vision in this way, with a slight change in the stages order.

- **Perception.** Usage of cameras which allow capturing luminosity, colors, contrast, movement and distortions.
- **Transmission.** Sending those images from the camera to a computer through some communication protocol.
- **Transformation.** Transform those images into an array which contains the main features of the images in terms of numbers.
- **Interpretation.** Make use of the different techniques and algorithms to analyze the images and get the desired results.

But indeed, robot vision can be depicted in its own way as it can be seen in Figure 1. To understand robot vision is needed to have clear the difference between robot vision, computer vision, machine vision, image processing and pattern recognition. In basic terms, Robot vision involves a camera hardware and computer algorithms to process visual data.

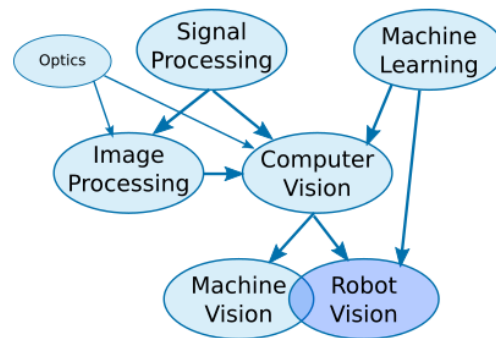


Figure 1. Vision Architecture [12]

Signal processing involves processing electronic signals to extract information or prepare them for future processing. Among some of signals that can be processes there are analog electrical signals, digital electronic signals and frequency signals. But images are not signals, so signal processing doesn't involve image processing.

Images processing involves techniques which are primarily used to improve the quality of an image, convert it into another format, or change some parameters for further processing.

Computer vision is the art of extracting information from images to make sense of them. Images processing and computer vision are very close terms, which they complement each other to detect objects within an image. Both influenced by the domain of Optics.

Object detection is mainly done through pattern detection, or in general terms Machine Learning and Deep Learning. This branch is specialized on recognizing patterns in data, so the object recognition is performed by detecting if the object it sees is similar to previous objects.

Machine Learning and Deep Learning are a complement to Computer Vision. In practice, these domains are combined in such a way that Computer vision detects features and information from an image, and then this information is used as an input to the Machine Learning algorithms, which they will later decide whether those type of data are useful or not to classify the image according to the learned knowledge about how that object should look like in terms of data and key features.

The next term related to vision is Machine Vision. It refers to the industrial use of vision for automatic inspection, process control and robot guidance. It's part of the engineering domain.

Finally, it is Robot Vision. It involves techniques from all the mentioned terms and incorporates aspects of robotics into its techniques and algorithms, such as kinematics, reference frame calibration and robot's ability to physically affect the environment. It also belongs to the domain of engineering but there are a few subtle differences which make Robot Vision different from Machine Vision. A clear example is parts inspection; the part is simply placed in front of the sensor which looks for faults in an automated way.

To simplify the concepts in terms of inputs and outputs of the different terms explained above, look at Table 1.

Table 1. Vision classification [12]

Technique	Input	Output
Signal Processing	Electric signals	Electric signals
Image Processing	Images	Images
Computer Vision	Images	Information/features
Pattern Recognition/Machine Learning	Information/features	Information
Machine Vision	Images	Information
Robot Vision	Images	Physical Action

2.3 Object recognition Strategies

There are many different algorithms when it comes to object recognition. In this section the general strategies in which these algorithms are based will be explained. The

first two methods, appearance-base and feature-base, are used combinedly in lots of algorithms. Then the most revolutionary one is explained, artificial intelligence, in its two variances, Machine Learning and Deep Learning. A detailed explanation of the main Deep Learning algorithms will be given.

2.3.1 Model-Based Method

In this method a 3D model of the object being recognized is available. It contains high detailed information concerning shape of the structure, spatial relationship between parts and its appearance. For this purpose, generative models are commonly used. Through mathematical functions and operators these models are described with such detail that is possible to generate images of target objects

Two approaches can be used solve the object recognition problem. The first one is obtaining 3D information of the object from images through specialized hardware such as stereo vision camera, for a later comparison with the object models. On the other hand, it's possible to obtain 2D images of the object and compare them with the 2D projections of the generative model [91].

The main algorithm is CBIR (Content-Based image Retrieval).CBIR uses the visual properties of an image such as color, shape and texture to represent and index the image, and these visual contents of the image are automatically extracted and described as multi-dimensional feature values [79].

This method is normally used for object Recognition for Service Robots in the Smart Environment [91]. A smart environment is developed for light-weight service robots, full of smart items with radio frequency identification (RFID) tag and smart devices with a RFID reader. John Wiley & Sons in their work propose a new method for object recognition in these environments, which is robust to environment constrains. This system uses two visual descriptors of MPEG-7, dominant color descriptor (DCD) and edge histogram descriptor (EHD) [55].

Furthermore, Park J. et al. have proposed a new approach base on a robot full of RFID tags and connected by wireless networks. RFID tags provide the presence and key information of the object and the robot should be able to recognize the object through vision processing based on a detected RFID code and previously stored visual descriptor information, inherent to the model-based methodology.

2.3.2 Appearance-Based Method

This method is also known as View-Based Recognition. Opposite to the previous method, in appearance-based object recognition, 3D model of the object is not available. The information available are images of the object viewed at different angles and distances.

The work methodology of these systems is comparing 2D images with the model 2D images and looking for the more similar one. This matching is simpler than a 3D matching but has the disadvantage of the requiring much space.

To do this matching there are also several techniques. One of the most common ones is to extract salient information such as regions, edges, corner points, etc. Another popular approach is to extract translation, rotation and scale invariants features. The purpose of these techniques is to reduce the dimensions to obtain a lower dimension subspace, eigenspace [27]. These approaches require both extractions, from the images from the database, and from the image to be compared, so that the new input images can be projected on the eigenspace and then correspondence is examined [44].

Following this methodology Wenyi Zhao et al. use PCA (Principle Component Analysis) not recognize faces [94]. Thanks to this statistics procedure is possible to reduce the dimensionality of a collection of observed data. In general, PCA is an orthogonal linear transformation which transforms data to a new coordinate system. It is a suitable technique for object recognition but for classification, its performance is not so good [86].

Another important algorithm is LDA (Linear Discriminant Analysis), which is used in statistics for dimensionality reduction or classification. Mean and same covariance represent each class. The algorithm minimizes the intra-class variance, while the inter-class variance is maximized. Ming-Li et al. use this approach for image matrix analysis [68], while Li-Fen et al. have developed an algorithm based on it to recognize faces [61].

When talking about indoor environments, scene understanding has been dealt with from different approaches. For example, with the Hierarchical Bayesian Model proposed by Steinberg et al. [89], which use unsupervised or visual-data techniques to understand scenes. Similarly, Changhai et al. [46] on their work use Bayesian filtering with motion

cues for the same purpose of indoor scene understanding. In this case motion cues are used to compute likelihoods of indoor structure hypothesis and the Bayesian filter is used to give probability distribution over the set of hypotheses.

2.3.3 Feature-Based Method

The algorithms based on this strategy recognize objects base on specific features, understanding by feature a characteristic of the object, which by itself does not describe an object, but along with others they do. The main attributes that characterize an image are colors, geometry, edges, contour lines, gradient of pixels intensities.

Features can be classified into global and local features. A global feature is the one which can be found considering the whole image, while a local feature is found in small parts of the image. The main example for a global feature is a histogram of the pixel intensity or color

A specific type of feature is searched in every input image, this feature is then compared to a database containing models of the objects, and it is verified if there are recognized objects.

If global features are used, illumination, position and rotation may affect the results as the image will present significant differences. That's why descriptors of local features are more robust against these problems. Thus, they lead to better results [56].

The first time this feature-based Scale-Invariant Feature Transform (SIFT) algorithm was presented was in 2004 by Lowe [39]. The general SIFT method consist in four major stages: scale-space extrema detection, keypoint localization, orientation assignment, keypoint descriptor. This method is used to recognize objects from local scale-invariant features [43], or for more specific application as the one that Apostolos P. Psyllos et al. propose on their paper "Vehicle Logo Recognition Using a SIFT-Based Enhanced Matching Scheme" [31]

Based on SIFT, Bay et al. used it as a model to build a more advance Speeded-Up Robust Feature algorithm (SURF). It consists of the same steps with some modifications [47]. Drew Schmitt et al. have developed an algorithm based on it to classify and located objects [40].

For indoor environments, other approaches try to overcome visual polysemia and

concept polymorphism (VPCP), such as Li et al. work, that use location-sensitive hashing (LSH) to improve understanding in large images [23]. It is also usual to use RGB-D images along with bottom-up segmentation as Saurabh et al. propose in their paper to understand indoor scenes [84]. They train RGB-D object detectors by analyzing and computing Histogram of Oriented Gradients (HOG) [84]. Another technique is using 3D Bag-Of-Words model for categorization of objects and scenes, as Redondo-Cabrera et al propose [81].

The proposal by Espinance et al. includes high-level semantic information in the recognition process [73]. They propose a new approach for indoor scene recognition based on generative probabilistic hierarchical model that uses common objects as an intermediate semantic representation. Thus, low-level features are associated to objects thanks to objects classifiers, and objects can be associated to scenes using contextual relations. Quattoni and Torralba [25] propose an indoor scene recognition algorithm based on combining local and global information. They test their approach using 67 indoor image categories with results that outperform current approaches for the case of indoor scenes.

2.3.4 Artificial Intelligence

Artificial intelligence is the intelligence based on computer systems. It is capable of perform any intellectual task that humans can do. Furthermore, it has to be able to learn, show knowledge, plan, make decisions under certain uncertainties, communicate in natural language and use all these abilities for a common goal [37]. Thus, the field of vision has moved towards this revolutionary line of investigation.

Following this investigation line two techniques have been developed, Machine Learning and Deep Learning.

2.3.4.1 Machine Learning

Machine learning is a type of artificial intelligence where are no written rules to generate intelligence, but an algorithm is created, and it can learn from the data and information [72]. Inputs are introduced along with their associated outputs, and the system itself generates a code to make them coincide. Once the learning process is complete, the system can be used to produce outputs from another set of inputs.

It differs from data mining in the sense that with data mining it is possible to extract

knowledge from data, while Machine Learning consist in learning to work and predict future data from the data available that moment.

In the vision field, machine learning is understood as the efficient development of a method capable of recognizing patterns in images. SVM (Support Vector Machine) is an algorithm based on Machine Learning [71]. It can be used to classify objects in indoor environment as Alejandra C. Hernández et al. expose on its work [24]. In addition, Rezaul K. Begg et al. apply the same methodology for the automatic recognition of gait changes due to ageing. For it, the gaits of 12 young and 12 elderly participants were recorded and analyzed, and the test results indicated an accuracy of 91.7% in its capacity to distinguish the two gait patterns from video recordings [78]. It can also be use Machine Learning with other algorithms for detection purposes. Edward Rosten et al. explain how to detect corners in high-speed using SLAM (simultaneous localization and mapping) and Machine Learning [82].

In terms of image recognition, Machine Learning has experienced a big development in driving applications. J.Stallkamp et al. propose several machine learning algorithms, based on LDA, for traffic signal recognition and provides a traffic sign dataset with around 50,000 images of German roads signs classified in 43 classes. The results of the work showed an accuracy of 95% in comparison to the 98% accuracy of human's performance in the same test [52]. Furthermore, Savaram Ravindra exposes on her work the machine learning algorithms such as k-means, AdaBoosting or Decision Matrixes, used in self-driving cars applications [16]. These applications include evaluation of driver condition or driving scenario classification through data fusion from different external and internal sensors.

AdaBoosting is a meta-algorithm that can be used in conjunction with many other types of learning algorithms to improve performance, where he output of the other learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier [3]. Based also in this algorithm, M.S.Barlett et al. present system classifies 7 expressions, neutral, anger, disgust, fear, joy, sadness, and surprise; whether they occur singly or in combination with other actions, with a mean agreement rate of 93% with human FACS (facial action coding system) [63]. In contrast, P.Viola et al. used this methodology along with their new mage representation approach "integral image" for rapid face detection, being able process a 384 by 288 pixel image in about 0.067 seconds with an average accuracy of 90% [74].

2.3.4.2 Deep Learning

Deep Learning is a type of Machine Learning that includes block, function compositions, which can be adjusted at each step to obtain better results. This is done adjusting the blocks that are far from the desired output [51]. The difference between Machine Learning process and Deep Learning process, as it can be appreciated in Figure 2, lies on the features extraction process. In the first method, the features are designed manually by the user, and from them the mapping is done for a later output obtainment. On the counter, deep learning chooses automatically the main features and it is even possible to extract more abstract features with the usage of additional hidden layers, which will be explained later on this chapter, and from them the mapping is done, and outputs are obtained similarly as in machine learning.

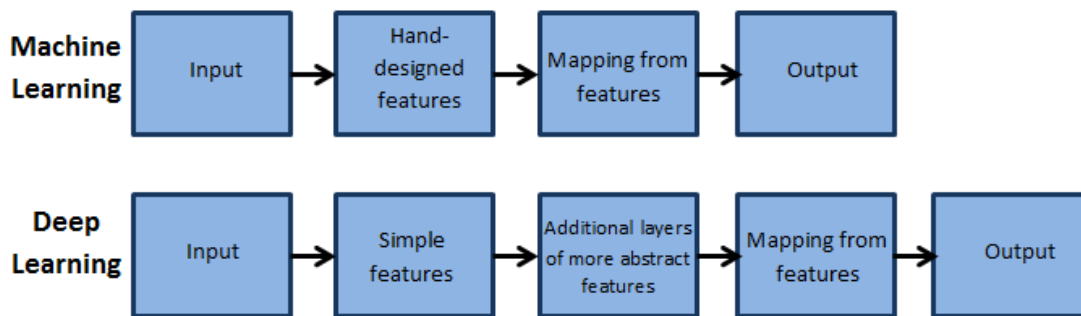


Figure 2. Deep learning representation process [51]

This method is based artificial neural networks. They try to imitate human neurons applying the same rules as in human brains to generate then artificial intelligence. Networks are usually implemented with electronic components or with a software simulation on a computer. It is then a massive processor of parallel distribution elaborated by simpler processing units which have a natural propensity to store experimental knowledge and have it available for its use. In Table 2 the potential of computers compared to a human brain is shown.

It is similar to a human brain in two main characteristics; the knowledge is acquired by the network through the surrounding and by a learning process. The process used to realize this learning process is called learning algorithm, whose function is to modify the synaptic weights in an ordered way to obtain the desired design. Its correspondence to a biological neuron can be seen in Figure 3.

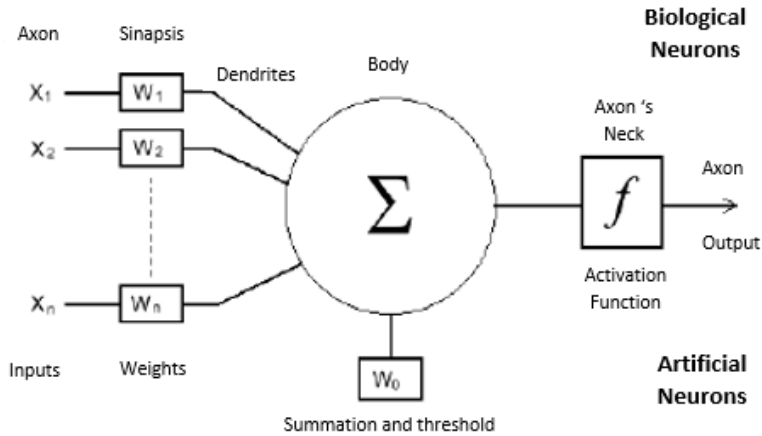


Figure 3. Artificial neuron schema

Neurons are activated if the sum of the input signals is above a certain threshold, so a simple way explained, artificial neural networks recognize objects after analyzing in deep the image and analyzing the activation state of the network which is triggered by the activation function and depending of this state, the output is one or another.

New inputs go through the same path, some neurons might be activated based on the trained network and finally, this leads to the most suitable classification [85][34].

Table 2. Comparison between brain and computer [26]

	Brain	Computer
No. of processing units	$\approx 10^{11}$	$\approx 10^{11}$
Type of processing units	Neurons	Transistors
Type of calculation	Massively parallel	Usually serial
Data storage	Associative	Address-based
Switching time	$\approx 10^{-3}s$	$\approx 10^{-9}s$
Possible switching operations	$\approx 10^{13} \frac{1}{s}$	$\approx 10^{18} \frac{1}{s}$
Actual switching operations	$\approx 10^{12} \frac{1}{s}$	$\approx 10^{10} \frac{1}{s}$

Networks consist of several layers, in which neurons with the same function are grouped (see Figure 4). Thus, they can be distinguished 3 types of neurons:

- **Input Neurons.** They are neurons which function is to receive the stimuli from the outside (inputs) and transmit them to the next layer. They form the input layer, and they are also denominated “source nodes”, they can be referred to the number of pixels of an image.
- **Hidden Neurons.** They can constitute one or several layers of one or several neurons. They are called hidden computational nodes because this part of

the network cannot be seen directly from the entrance nor the exit. These neurons are in charge of processing the information and realize the intermediate calculations in an automatic way. The presence of these layers is what allows the network to work with more complex data and go deeper into the analysis of information. Thus, they are the key ingredient in deep learning. The term “deep” refers to the depth of the networks when using hidden layers to compute the data.

- **Output Neurons.** They conform the output layer, and their function is to take the already processed information of the neural network, make the final calculations and provide the proposed solution to the system. It is linked to the number of classes able to be recognized.

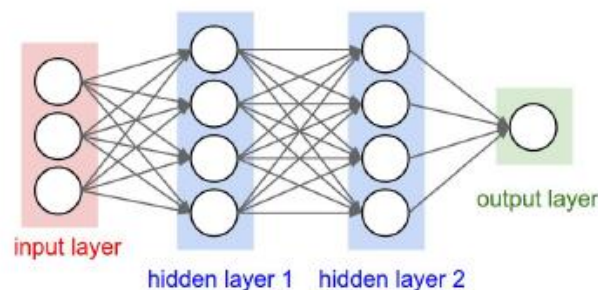


Figure 4. Neural network schema

There is a type of network specially used for image processing and analysis purposes for the little preprocessing of the image needed, it's the CNN (Convolutional Neural Network), which will be furthered explained in chapter 4.

This method has revolutionized the field of image recognition and classification. And based on the CNN principles, many groups of researchers have developed their own networks to recognize objects.

Some basic examples are MNIST or CIFAR10. MNIST consist of a first approach to image recognition based on convolutional neural network. Its main goal is to recognize handwritten digits in what is called the MNIST dataset, which comprises 60.000 test examples of the handwritten digits 0-9, formatted as 28x28-pixel monochrome image [49]. While CIFAR10 is dataset of 60.000 32x32 color images used to train computer vision algorithms based on CNN to detect 10 different classes represented by airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks, with 6.000 images of each class [22].

Since 2012, deep learning has brought big breakthroughs in developing models for

the task of image classification. In the next bullet points, it's going to be shown the progress of deep learning on this task through some of the major architectures that made that progress possible, which are AlexNet, VGGNet, GoogLeNet and its inception module. These challenging algorithms were developed to classify a large number of objects, with the highest accuracy possible and in the less time possible, and to measure its performance in image recognition challenges. They all have the same characteristic, they classify one object at a time and they provide the top 5 predictions for it.

Based on this revolutionary method, researchers have tried to develop their methods for indoor scene recognition. Lucas Pinheiro uses deep residual networks to analyze and detect anomalies in surveillance videos [62].

Tung and Little employ a concatenated version of object detectors produced by Never-Ending Image Learning (NEIL) and ImageNet models to learn the objects and their relationship in the scenes [92]. Furthermore, Rangel, J.C. exploits deep learning techniques for scene understanding, both indoor and outdoor through 3D data [57].

- **AlexNet**

The algorithm exposed by A. Krizhevsky et al. used a Deep Convolutional Neural Network (CNN) for the task of image classification. It was the first time to successfully use these types of networks for large scale image classification. It takes the name of the main author Alex Krizhevsky and it achieved nearly a 50% reduction in the error rate in the image recognition challenge ImageNet [59]. AlexNet set the bar, providing the baseline and default techniques of using CNNs for computer vision tasks. The main contributions that came from this development were the following:

1. The model was trained using parallel computations on two GPUs with large amounts of labeled data from ImageNet.
2. For the non-linearity activation functions the algorithm used ReLU, which lead to a decrease in the training time in comparison to other activation functions.
3. To process the images, it used data augmentation techniques that consisted of image translations, horizontal reflections, and mean subtraction.
4. In order to combat the problem of over-fitting to the training data, it used dropout layers.
5. The designed layer configuration consists of successive convolution and pooling layers, followed by fully-connected layers. In Figure 5 the architecture

of AlexNet can be appreciated. This design is still the basis of many state-of-the-art networks.

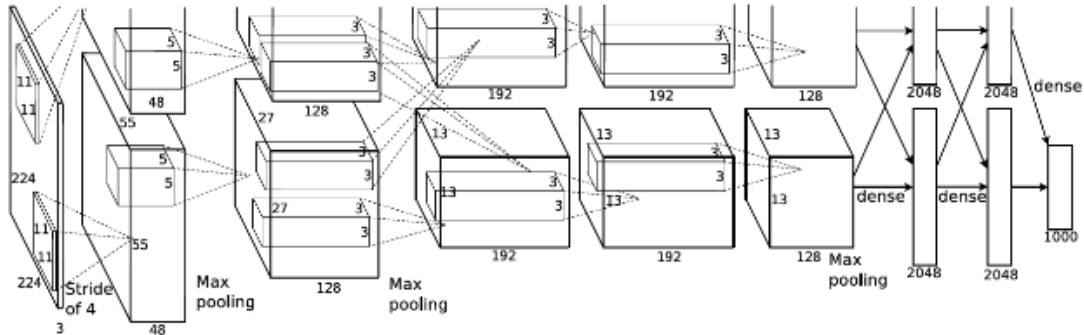


Figure 5. AlexNet Architecture [59]

This network configuration has been used for multiple applications. Kevin Lin et al. proved its efficiency on hashing codes for fast image retrieval over a dataset of 1 million clothing images and outperforming several hashing algorithms on the CIFAR-10 and MNIST datasets [58]. Another example is its use for medical purposes; as Marios Anthimopoulos et al. exposes, it can be applied to classify interstitial lung diseases. With a dataset of 14696 images patches, derived from 120 CT scans, this approach proved its effectiveness against previous methods not based on neural networks by reaching 85.5% of accuracy in the classifications [65].

• VGGNet

The next advances in convolutional neural networks were presented in the in the paper “Very Deep Convolutional Neural Networks for Large-Scale Image Recognition” by the researchers at Visual Graphics Group at Oxford [59]. The main idea is that to get high accuracy it is just needed a deep network with lots of small 3x3 convolutions and non-linearities. Thus, the main contributions of VGGNets are:

1. The use of only 3x3 sized filters (see Figure 6) instead of the 11x11 used in AlexNet. The advantage of this is that it simulates a larger filter while keeping the benefits of smaller filter sizes. The first benefit of smaller filters is a decrease in the number of parameters. The second is being able to use a ReLU function in between each convolution, that introducing more non-linearity into the network which makes the decision function more discriminative.

2. As the spatial information decreases (from the down-sampling down by max pooling), it should be encoded as more discriminative features to use for accurate and highly discriminative classification. Thus, the number of feature maps is increased with depth to be able to capture these features to be used for classification.
3. It introduced scale jittering, a new kind of data augmentation.

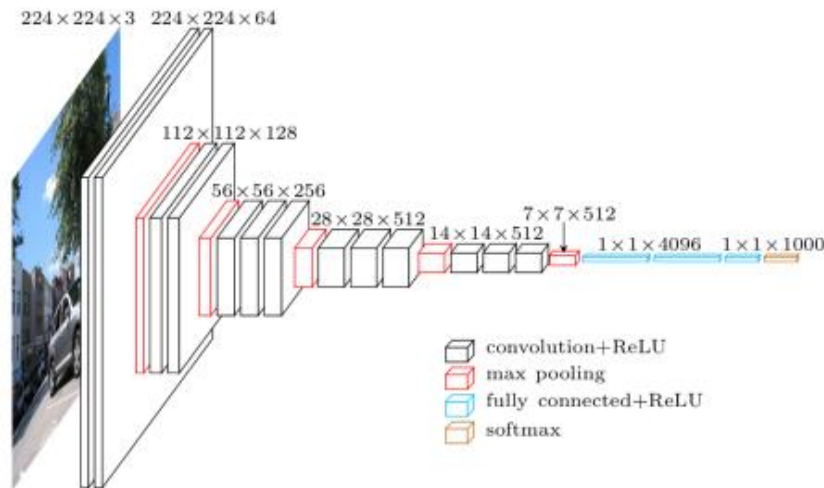


Figure 6. VGGNet Architecture [59]

Algorithms based VGGnet Architecture have been developed to recognize facial emotions in real time. Dam Duncan et al. achieved a 90.7% of accuracy during the training and a 57.1% while the testing face, being able to recognize anger, disgust, fear, happiness, sadness, surprise and contempt over a live video stream [38]. Hung Nguyen et al. have used this model for an Efficient and reliable monitoring of wild animals in their natural habitats. Using a single-labeled dataset from Wildlife Spotter project, the network presents an accuracy of 96.6% for the task of detecting images containing animal, and 90.4% for identifying the three most common species among the set of images of wild animals taken in South-central Victoria, Australia [50].

• GoogLeNet and its inception module

The GoogLeNet architecture, designed by researchers at Google, was the first to really address the issue of computational resources along with multi-scale processing in the paper “Going Deeper with Convolutions” [35]. They design the inception module (Figure 7), a new style of building neural networks which is de base of the InceptionV3 algorithm (see its architecture in Figure 8). InceptionV3 was the winner of the ImageNet 2014, being able to classify 1000 different objects and giving the top 5 predictions for

each object. The inception module and GoogLeNet made the following contributions:

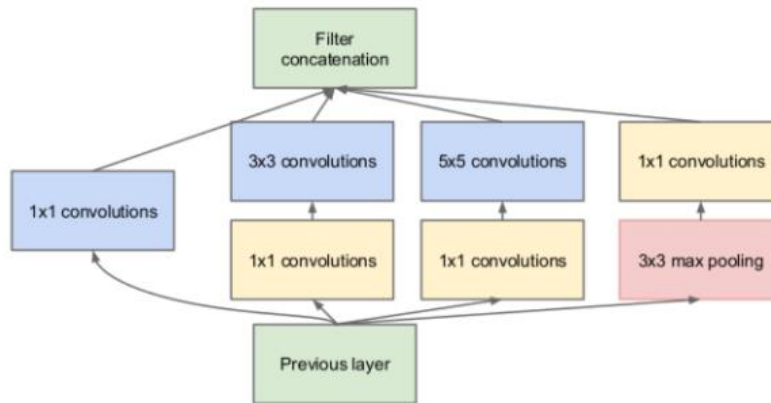


Figure 7. Inception Module from GoogLeNet [35]

1. Use of 1x1 convolutions before each 3x3 and 5x5, which reduces the number of feature maps passed through each layer, and so it reduces computations and memory consumption.
2. The inception module has 1x1, 3x3, and 5x5 convolutions all in parallel. By this way and through training, the network can decide depending on what information would be learned and used. Furthermore, it allows for multi-scale processing. The model can recover both local features via smaller convolutions and high abstracted features with larger convolutions.
3. It introduced the idea that CNN can also be increased in for better performance and not just depth.

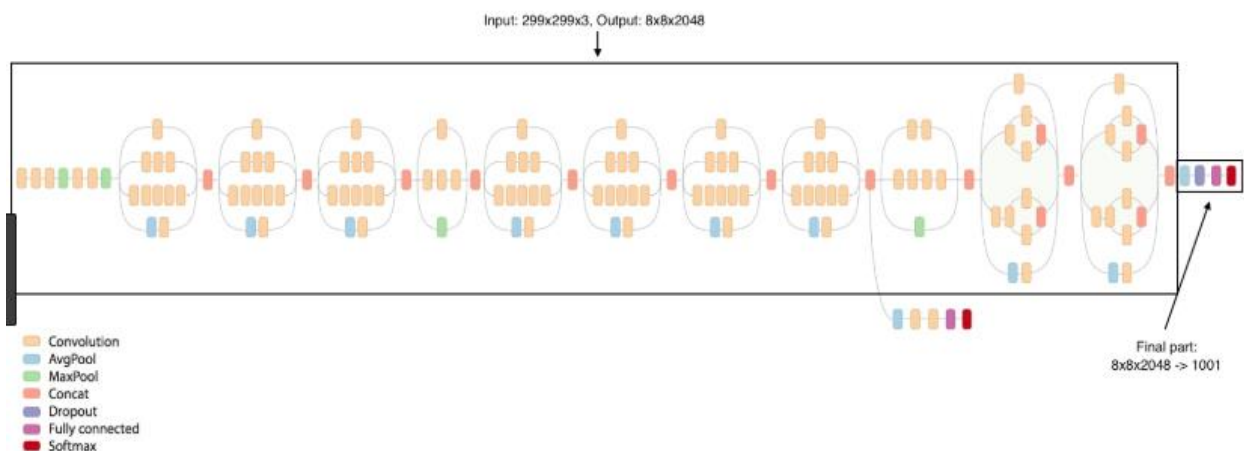


Figure 8. InceptionV3 architecture

Using a pre-trained GoogLeNet model, Ashutosh Singla et al., in the field of image-based dietary assessment, achieved on experimental results a high accuracy of 99.2% on the food/non-food classification and 83.6% on the food category recognition [32]. By contrast, Hervé Goëau et al., training a GoogLeNet model with a dataset of more than 110K images illustrating 1000 plant species living in West Europe, can identify plants in an open-world [48].

The performance of this network configuration has been compared with others in the image recognition field. For example, Sang-Geol Lee et al. uses variation of AlexNet and GoogLeNet models to recognize Korean characters. 2,350 Korean character classes for a total of 5,139,450 data samples were used for the training, while the testing is done using different character fonts, showing an average of classification success rates of 90.12% and 89.14% respectively, and a faster classification speed in the case of the GoogLeNet model [83].

Comparisons between Inception module and VGGnet are also carried out. James Carroll developed two systems based on these models for distracted driving recognition, classifying safe and unsafe driving images [53]. Both systems were able to consistently recognize distraction such as texting or talking on the phone, but failed classifying distractions like drinking or applying makeup.

2.3.5 Multi-Object and Real-Time recognition

In this section the main multi-object recognition systems are presented and its performance while working with real-time images is commented. The most conventional algorithms are based on artificial neural networks to perform the multi-object detection task although it can be also done through contour detection and hierarchical image segmentation as Pablo Arbaláez and Charles Fowlkes propose on their paper [75].

Talking about neural networks, one of the fundamental problems with such type of task is that you can't apply the fundamental CNN to figure out objects within these, because the traditional CNN tend to get confused when there are multiple labels associated with an image. A traditional approach to multiple object classification in an image would be use object detection algorithm and run CNN on top of that as its done in RCNN models, which will be explained later on this section.

As solution to this problem, YOLO and SSD, the main systems designed for multi-Object recognition and the state-of-the-art in terms of multi-object recognition and real

time recognition, they both proposed the usage of a single neural network to perform both the object detection and the object recognition, as it will be detailed in the following subsections.

The main advantage of multi-object recognition algorithms is that they can recognize and classify more than one object in an image, drawing bounding boxes around the different objects. On the other hand, the disadvantage is that the output for each prediction comes alone, just showing the top prediction and no other probabilities.

In terms of real time object recognition, the main feature needed is speed of processing along with accuracy. Both related is what makes a system robust for this task, speed without accuracy is useless for having realistic results, and accuracy without speed is not useful either because it can lead to wrong environment understanding.

That's why; the main algorithms that overcome the others are YOLO and SSD. Both of them were design for real time object recognition applications and that's why they perform better than the others. Nevertheless, there are other models that can perform real-time object recognition after modifying the code to accept frames from a camera as input, but these 2 systems will make faster predictions compared to other developments (see Figure 9).

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Figure 9. Results in MS COCO according to SSD paper [36]

On the paper, in terms of speed YOLO is the best model, but SSD performs better when it comes to accuracy. But depending on who makes the evaluation, one model is better than the other one as it can be seen in Figure 10 and Figure 9. (For further model's comparison check bibliographic reference number [9]).

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	-	-	7.07 Bn	200

Figure 10. Results in COCO according to YOLOError! Reference source not found. [21].

- **R-CNN**

R-CNN (Region-based convolutional neural network) is an algorithm that uses CNN principles to classify objects. Its peculiarity is that it performs a preprocessing on the image, it applies the model to an image at multiple locations and scales, which are the result of segmenting the input image, so object detections are considered as result of high scoring regions of the image [45]. This procedure can be seen in Figure 11.

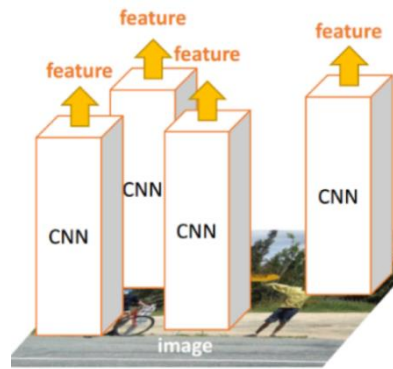


Figure 11. R-CNN process

Each part of the image is classified, so at the end you get an output map with results for each portion. Those regions with the same output result are considered as one object and thus a bounding box is drawn. A threshold value to detect objects is also applied in this algorithm, to discriminate between false positives.

On the one hand, this system has the advantage of giving several predictions for each segmented image, but on the other hand, these types of object detectors are very slow as they have to process many data. The more accurate the system is made, into smaller images the main image is subdivided and thus the slower the system it is.

R-CNN has successfully been proved to detect scene text with both high accuracy and efficiency, taking only 0.09s per image in a fast implementation, as Minghui Liao exposes. Furthermore, combined with a text recognizer, the system significantly outperforms state-of-the-art approaches on word spotting and end-to-end text recognition tasks [69]. In contrast, Alvaro Fuentes et al. use this model along with a GPU to recognize tomato plant diseases and pests in real-time. The results show a 75.37% mean accuracy for this task using a R-CNN model, and 82.53% accuracy and faster speed using SSD model [29], which will be explained later in this section.

• YOLO

You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP (mean Average Precision) of 57.9% on COCO test-dev, a dataset of common objects [21].

Opposite to other traditional multi-object algorithms, YOLOs approach is totally different. A single neural network is applied to the full image, and it divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities, and the results are controlled with a threshold value that can be selected by the user.

To detect objects, YOLO divides the input image into an NxN grid and each grid cell predicts only one object (see Figure 14). For each grid cell the model predicts B boundary boxes with its own box confidence score, it detects one object only no matter how many B boxes they are and predicts C conditional class probabilities.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally, the confidence prediction represents the IOU (Intersection Over Union, see Figure 12 and Figure 13) between the predicted box and any ground truth box.

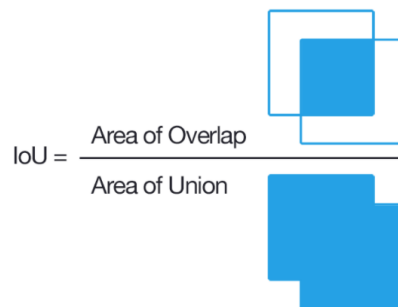


Figure 12. IoU formula

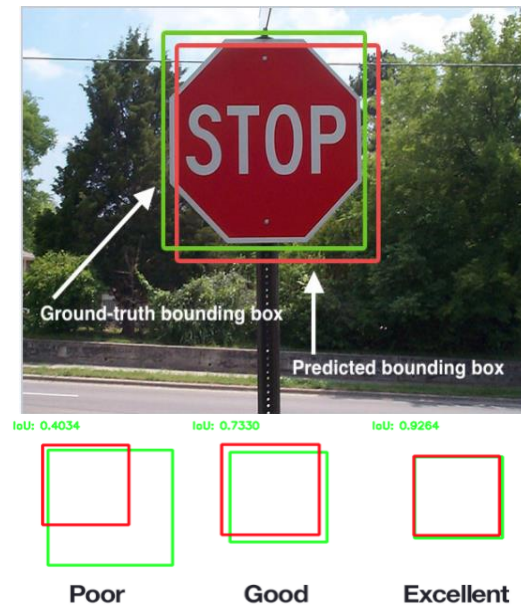


Figure 13. IoU representation

In YOLO, confidence reflects how confident the model is that the box contains an object and how accurate it thinks the box is that it predicts. It is defined as described in equation 1 [80].

$$\text{Confidence} = \text{Pr}(\text{object}) * \text{IOU} \quad (1)$$

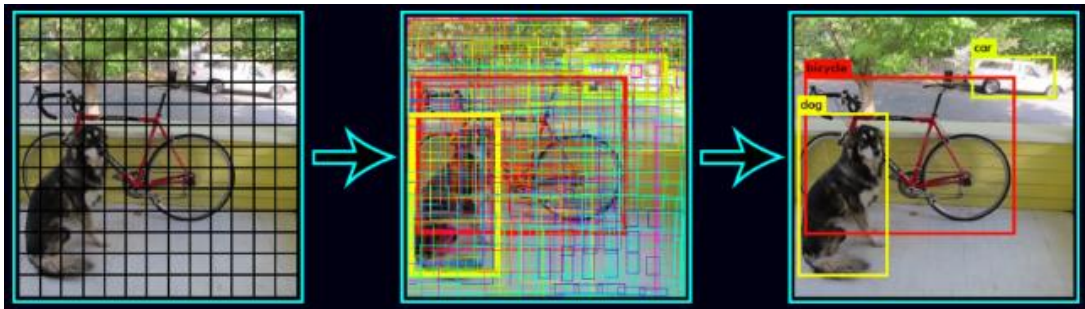


Figure 14. YOLO image processing

The main advantage that this model has over classifier-based systems like R-CNN, is that it looks at the whole image at test time, and that predictions are done with a single network evaluation. This provides an output more influenced by global context and makes it extremely faster in comparison to R-CNN [80], but only one label per box.

YOLO has been implemented for multiple applications. Matija Radovic et al. system can detect and classify objects in aerial images taken from unmanned aerial vehicles (UAVs). The training was performed with Google Maps images in order to classify the object class “airplanes” with an accuracy in the test phase of 97.5% [66]. Similarly,

Sushma Nagaraj trained this model with traffic data captured at California and Nebraska for edge-based street object detection. The model detects 14 objects with 25% average accuracy on NVIDIA Jetson TX2 [90]. Michal Busta et al. research is focused in scene text localization and recognition using YOLO architecture running at 10 frames per second on a NVidia K80 [67].

• SSD

SSD (Single Shot MultiBox Detector) was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as COCO [36].

The bounding box regression technique of SSD is inspired by Szegedy's work on MultiBox, a method for fast class-agnostic bounding box coordinate proposals presented in the paper "Scalable, High-Quality Object Detection" [33].

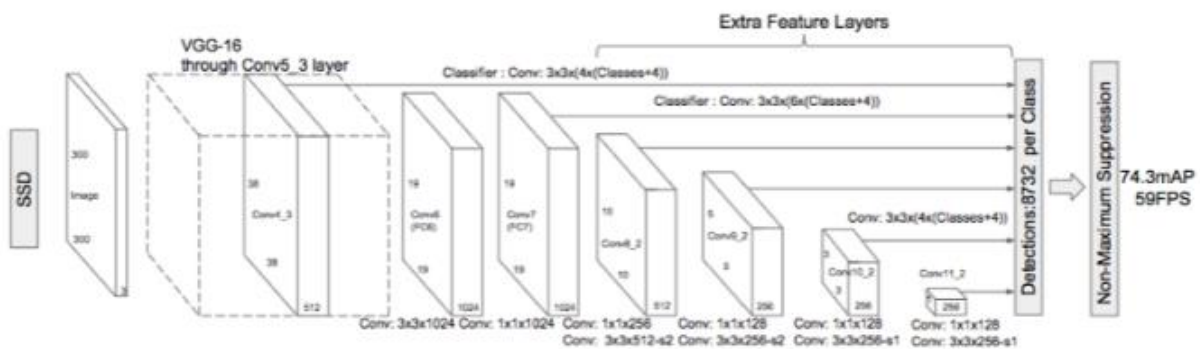


Figure 15. SSD architecture [36]

This method, like YOLO, uses a single deep neural network for detecting objects (see its architecture in Figure 15). SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. Thus, the process can be differentiated in 2 parts, multi-scale feature maps for detection and applying convolution filters to detect objects.

To create multi-scale feature maps, convolutional feature layers are added so predictions of detections at multiple scales are possible. In opposition to YOLO, which operate on a single scale feature map, SSDs convolutional model for predicting detections is different for each feature layer.

At prediction time, to improve accuracy small convolutional filters are added to predict

object classes and offsets to default boundary boxes. Thus, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network.

SDD loss function for multibox detection expresses how far off the prediction made is from reality. Besides, it combines two critical components, confidence loss and location loss; as it is shown in equation 2.

$$SDD\ loss = confidence\ loss + \alpha * location\ (2)$$

The alpha term helps us in balancing the contribution of the location loss. As usual in deep learning, the goal is to find the parameter values that most optimally reduce the loss function, thereby bringing our predictions closer to the ground truth.

Some of the applications for which this architecture has been implemented are face recognition and pose estimation. Peiyun Hu et al. have developed a system to find tiny faces of 3px tall on images, producing a precision of 81% while prior art technique ranges from 29-64%, and being capable of finding around 800 faces out of 1000 in an image [77]. On the other hand, Patrick Poirson et al. use SSD to detect and estimate the 3D pose of objects for navigation and robotics applications [76].

In addition, Vicky Kaloageiton et al. compare SSD architecture and R-CNN for the purpose of spatio-temporal action localization with an action tubelet detector. Video datasets were used for the training and testing, a sequence of images is analyzed to classify the type of action, and in the three different datasets used SSD outperforms R-CNN and previous state-of-the-art systems in terms of accuracy and speed [93].

3 Proposed System

In this chapter a general overview of the proposed system is presented. The main steps and parts of the system are exposed for a better understanding in the next chapters, where they are described more in detail.

This thesis can be divided in two main parts as it can be seen in Figure 16. A first part where a Convolutional Neural Network is built and trained on various classes of objects, and a second part where different state-of-the-art image recognition modules, as well as the built CNN, are developed for the task of multi-object recognition in real time for its later use, testing and comparison in an indoor environment for a mobile robot application.

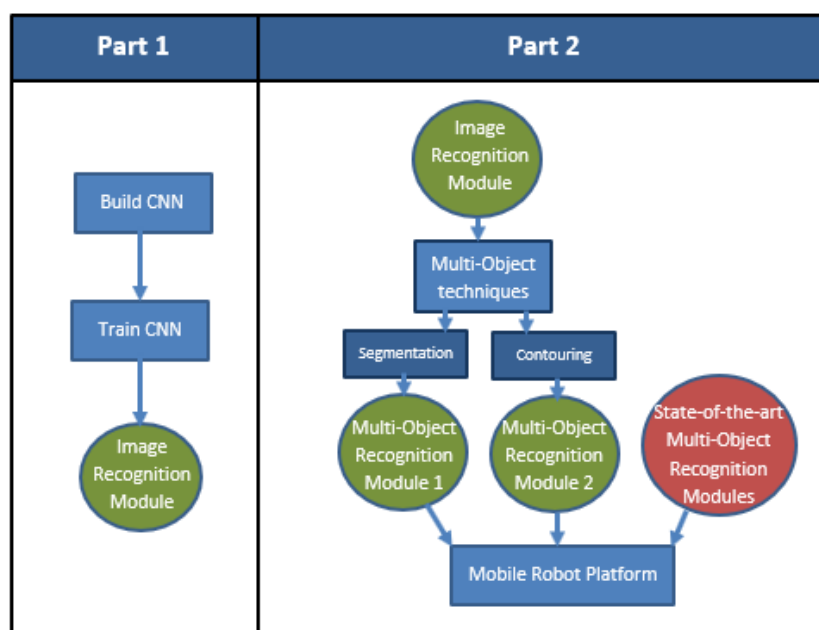


Figure 16. General system schema

3.1 Part 1: Developing a CNN

In this part, information about the process of creating a CNN and its characteristics is explained. The system consists of two main modules that will be implemented, a first module where the CNN is built and trained, and a second module which constitutes the image recognition system. A total of 12 tests will be performed in order to select the best configuration.

3.2 Part 2: Multi-object recognition and Indoor tests on mobile robot platform

The second part of the work constitutes the development of the image recognition modules for the multi-object task. YOLO and SSD are already prepared for it so no development in this field has to be performed.

Inception module and the built CNN module can just recognize one object at a time. Thus, two different techniques are developed over this system for detecting multiple-objects in an image.

One technique is based on image segmentation, and the other in contour detection. The different segmented images or contours are passed to the one-object recognition module so multiple objects can be classified. The outputs of this second part are 4 different modules: 2 multi-object Inception modules, each one applying one technique; and 2 multi-object for the developed CNN, based on the same principle.

The next step consists of developing the previously 4 modules and the state-of-the-art multi-object detection modules, YOLO and SSD, to be embedded in a mobile robot platform. For it, they have to be implemented for real-time image recognition and adapted to the robot operating system, ROS.

All the different modules are tested in a not previously prepared nor conditioned indoor environment for a further analysis and a later selection of the best system for the desired task of real-time multi-object recognition in indoor environments.

3.3 Experimental Platform Software

The software used to develop the different models of this project are presented in this section. The mobile robot platform works with ROS and Linux, and all the

systems are developed with Python programming language. OpenCV is the software with which the image processing is done and for the neural network building and usage the software used is TensorFlow.

3.3.1 Python

Python is an interpreted high-level and open source programming language for general-purpose programming created by Guido van Rossum and first released in 1991[60]. It features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python in this work is used for developing the different image recognition systems and converts them into a system that can operate in real time and upgraded to multi-object, as well as for building, training and evaluating the neural network and its later set up. For these tasks Python makes use of external programming libraries, OpenCV for the image processing, and TensorFlow for the neural network management.

3.3.2 OpenCV

OpenCV (Open Source Computer Vision Library) was designed for computational efficiency and with a strong focus on real-time applications, besides its library can take advantage of multi-core processing. Therefore, it is used for a wide range of applications, from interactive art, to mines inspection, stitching maps on the web or through advanced robotics [10].

In this thesis, OpenCV is used to processes real time images through a camera and apply them the needed conversion to make it usable for the rest of the coding. With OpenCV is possible to capture frames from a video, both real time or saved file, apply filters to the images, introduce text on it, crop the image, display images or videos on the laptop screen, detect contours, etc.

3.3.3 TensorFlow

TensorFlow is an open source software library for high performance numerical computation. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning [14]. Its computations are expressed as stateful dataflow

graphs. Its name derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors". It has its own logic of computational processing and processes distribution which makes Python multiprocessing option inviable.

TensorFlow in this work is used for two different tasks: as the working environment of the image recognition systems and to build and train a neural network based on deep learning from scratch. The image recognition systems use TensorFlow as the platform to load the neural network model, load the synaptic weights, the classes' labels and to pass the input image through the net in order to get the desired output.

3.3.4 ROS

Robot Operating System (ROS) is one of the principal frameworks in robotics these days. It offers tools and libraries for the development of robotic system. It is an open source software released under the BSD license. It was created in 2007 by Willow Garage in Stanford Artificial Intelligence Laboratory, and now the Open Source Robotics Foundation continues its development [70].

All the programs are free to use, reusable, implementable and integrable with all current and future robotics software. ROS provides solutions to the users in terms of hardware abstraction, communication mechanism between processes, distributed processes and low-level device control.

ROS is based in a graph architecture where processing occurs in nodes that can receive, publish or multiplex messages on actuators, sensors, control, states, and planning.

4 Designing and Training a Convolutional Neural Network

This chapter corresponds to the first part of the proposed system. Here, not only the main features of Convolutional Neural Networks will be explained, but also how to build one in Python [11], making use of TensorFlow library [14], and how to use that network to throw predictions over new input images. Thus, there are two different modules, one is in charge of building and training the network, and other whose mission is loading the trained network and its weights and make predictions for the input images (see Figure 17). Furthermore, the hyper-parameters chosen are detailed in addition to the training process carried out.

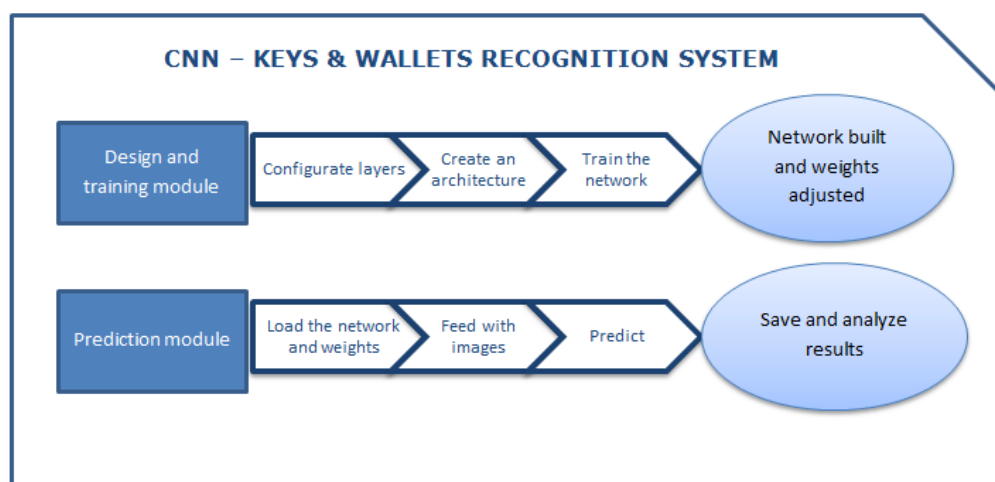


Figure 17. CNN recognition system

Standout that the system has been elaborated using TensorFlow tutorials [13] and further developed to perform the required tasks and save the information about both the design and architecture selected, annotating the different layers created and their parameters; and the training process, where is possible to see the learning path and the accuracy of the CNN.

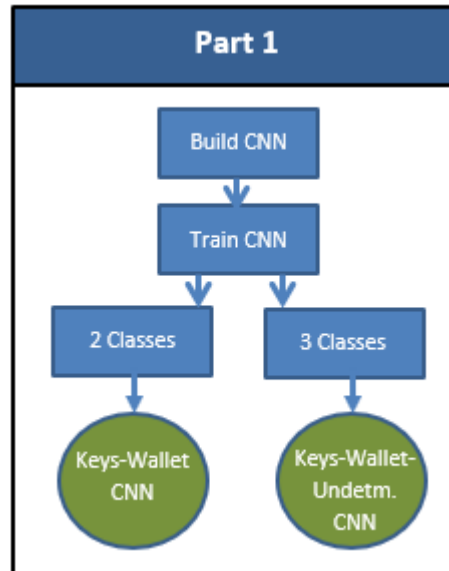


Figure 18. Part 1 schema

The process described in Figure 17 is carried out two times to build two different CNN trained to detect 2 classes or 3 classes of objects. The reason for this duplicity is explained later on this chapter as well as the different classes selected for the training. This first part of the thesis has as output two image recognition modules, as it can be observed in Figure 18.

4.1 Design and training module

In this subsection it will be dealt with the first module of the system, where the network is built and prepared to be trained.

Deep Convolutional Neural Networks are one type of artificial neural networks. The basic principles of CNNs are inspired by the biological visual cortex of humans, that's why they differentiate themselves for its great effectiveness while recognizing and classifying images, as it was design to receive images as inputs. The term deep comes from the presence of at least one hidden layer, and convolutional means the use of convolution layers. These types of neural networks are very effective when detecting faces, objects and traffic signs, that's way they are part of the robotics field

of interest, to develop vision of robots.

Due to the good performance of CNN when dealing with images, this network is the one chosen to be implemented. Building a neural network is not an easy task; there are many parameters to consider and many variables to define.

A CNN consists of an input and an output layer, as well as multiple hidden layers. As it can be seen in Figure 19, the function structure of a CNN is divided into two main blocks and a final part where the different labels are located. The main features of the process are convolution, pooling, activation function and fully connected layer. Thus, in order to build a CNN, is needed to replicate these blocks.

It has to be pointed out that the final output of these networks consists of a set of probabilities for each class in which the network has been trained. So, it does not only give you the most probable solution for the input image according to the knowledge acquired, but also which other objects it can be and with its confident.

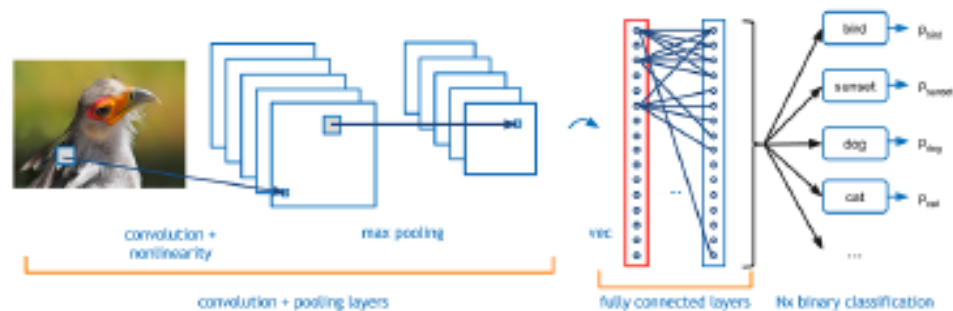


Figure 19. CNN image classifier diagram

In order to make an efficient network, its parameters have to be optimized. Optimizing a CNN for a specific task is a costly procedure and many iterations are needed to find the perfect architecture of the network. Nevertheless, once the main features are defined, optimizing the network is a matter of varying the hyper-parameters till the training gives the best outputs.

It is understood as Hyper-Parameter the parameter in deep learning whose value is initialized before the learning takes place. After having explained the main blocks of the CNN and how they operate, in order to optimize the system some parameters of these blocks have to be dealt with carefully.

The selection of these parameters is not trivial, each case has to be studied individually, and there are no generic rules to follow. It is mainly done through experience or through a try and failure process. The only rule that intuition gives is that

the complex the functionality that the network should have, the deeper and wider this one should be. These parameters are shown in Table 3.

Table 3. CNN Hyper-Parameters

CNN Hyper-Parameters	
Nº of hidden layers	
Nº of hidden neurons	
Activation function	
Loss function	
Learning rate	
Number of filters	
Filter shape	
Max pooling shape	
Nº of iterations/epochs during the training	

4.1.1 Creating network layers with TensorFlow

In this subsection the different layers that define a CNN are explained, as well as their main parameters and options. Furthermore, the TensorFlow commands to implement each layer are mentioned.

1. Convolution Layer

Convolutional layers are formed by three main blocks: a convolutional process, a pooling stage and an activation function (see Figure 20).

Convolutional layer	Convolutional Process	<code>tf.nn.conv2d</code>
	Max-Pooling Process	<code>tf.nn.max_pool</code>
	ReLU Activation Function	<code>tf.nn.relu</code>

Figure 20. Convolutional layer structure

The core building block of a CNN is the convolutional layer. The parameters of this layer are a set of learnable filters or kernel. During the convolution process the filter is applied all across the height and width of the input matrix, computing the dot product between the entries of the filter and the input. This will produce what is called an activation map and it is a characteristic of the image. The convolution process can be observed in Figure 21.



Figure 21. Convolution process

Depending on the **filter** applied one or other characteristics can be identified. **Padding** can be applied on the border of the input volume to control the output spatial size. The depth of a convolution indicates the **quantity of filters** applied in the process; while the **stride** means every how many pixels the filters are applied. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. It forms the full output volume of the convolution layer by stacking the activation maps for all filters along the depth dimension.

To build a convolutional layer, the TensorFlow function to be used is **tf.nn.conv2d**. It defines the previous outstanding parameters, and takes as input the output of the previous layer or in case of being part of the first layer, the images are the input.

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. It is used to reduce the dimension of each feature map and keep just the most important data. After each convolution, it is normal to apply pooling.

For it, it is needed to group the samples in an ordered way to apply pooling afterwards. The window that groups the samples is called aggregation, and from that window there are several non-linear functions to implement pooling and select the information.

1. **Max Pooling:** The windows group the input image samples into a set of non-overlapping rectangles and, for each such sub-region, the characteristic of highest value is selected. (see Figure 22)

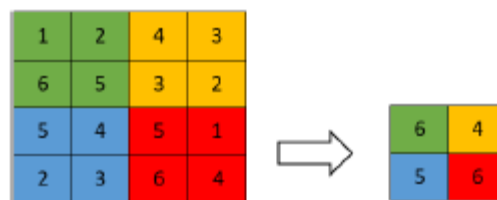


Figure 22. Max Pooling Example

2. **Sum Pooling:** This method applies sum as operation. The result of the sum is

composed by all the values of the characteristics grouped by the window. (see Figure 23).

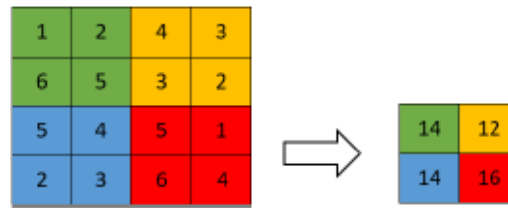


Figure 23. Sum Pooling Example

3. **Average Pooling:** Using this method, the result is the arithmetic mean of the values located inside the window. (see Figure 24)

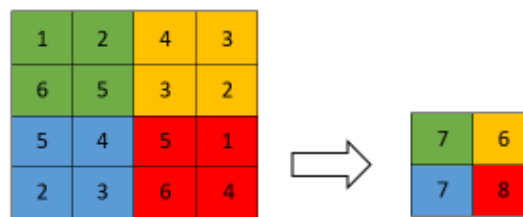


Figure 24. Average Pooling Example

The pooling layers reduce progressively the spatial size of the representation; reduce the number of parameters and thus the calculations in the network. Normally pooling layers are introduced periodically between successive convolutional layers in the CNN architecture. Besides, it provides certain control over errors, making the network invariant to certain small transformations, distortions or translations in the input image.

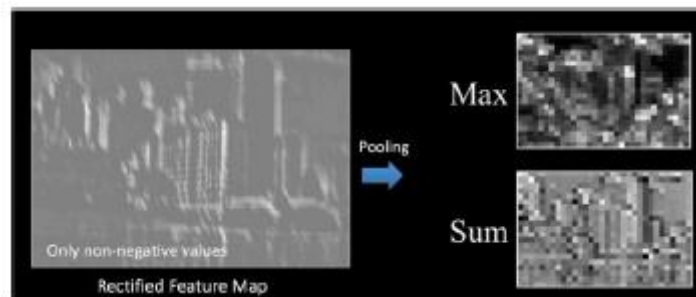


Figure 25. Max Pooling and Sum Pooling comparison

Max pooling is the most common (see pooling comparison in Figure 25) and the most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down-samples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations [6]. Thus, in the system to be built, max-pooling is applied through the function ***tf.nn.max_pool***.

The next step is the activation function. This block is applied after each convolution block and limits the output amplitude of the neuron to finite values and

determines the activation state, which indicates up to which point a neuron is activated. The main activation functions are:

1. **Threshold function.** Close to the limit value, the activation function of the neuron reacts in a very sensitive way. It represents the limit from which a neuron starts transmitting impulses. Each neuron has its own value, and it indicates the position of the gradient maximum value of the activation function. Only two values can be taken, which are constant.
2. **Sigmoid function.** It's the most common activation function of artificial neural networks. It is a strict increasing function which shows an equilibrium between lineal and non-lineal behaviors. It allows to take values among range, $[0,1]$ if it's a logic type, and $[-1,1]$ if it's a hyperbolic tangent type.
3. **ReLu function.** Rectified Linear Unit. This function applies an initial limit in such a way that all negative values are zero. This introduces no-linearities to lineal values, making the network a non-lineal system.

ReLU is the activation function of CNN [13], and its objective is to add non-linearities to the decision function and of the overall network and to reduce the variation between the extrema of the function without affecting the receptive fields of the layers. It is often preferred to other functions, because it trains the neural network several times faster without a significant penalty to generalization accuracy [59].

This activation function is added with the command `tf.nn.relu`. In Figure 26, it can be observed a visual explanation of how an image changes when applying ReLU.

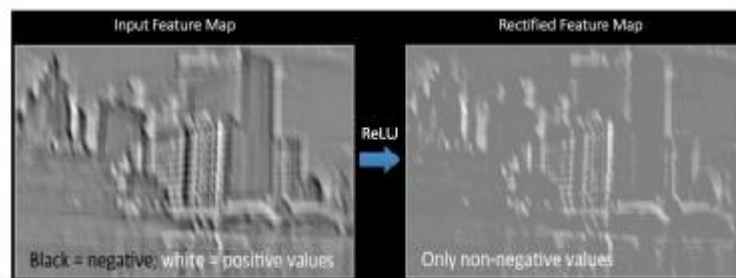


Figure 26. ReLU application example [7]

2. Flattening Layer

The result of a convolutional layer is a multi-dimensional tensor. The desired output is a single dimensional tensor. Thus, a Flattening Layer is used to reshape the

tensor to convert it into a one-dimension tensor.

In order to make this, the function used is ***tf.reshape***, where the input layer is reshaped to a one dimensional tensor of the number of features selected.

3. Fully Connected Layer

After the convolutional and max pooling stages, now the next layer is analyzed. In this block is where the different neurons of the Convolutional Neural Network are interconnected to each other, thus the high-level reasoning in the neural network is done via fully connected layers.

In a fully connected layer, the neurons have connections to all activations in the previous layers (see Figure 27). Their activations can hence be computed with a matrix multiplication followed by a bias offset. These interconnections are what allow the information to flow throw the network.

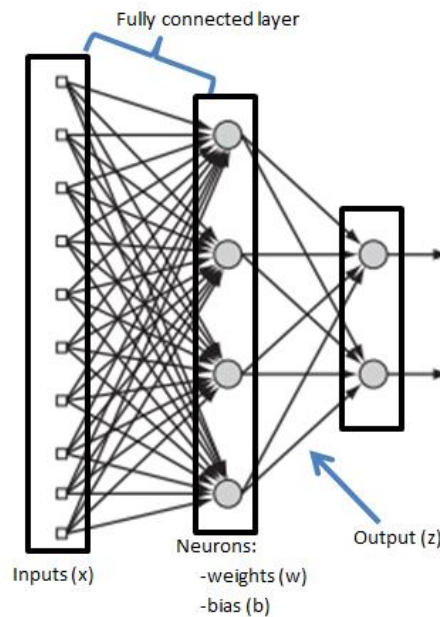


Figure 27. Fully Connected layer schema

The main characteristic in fully connected layers is that on all the inputs is applied the operation shown in equation 3. And all the inputs refer to all the neurons from the input layer. This is done with TensorFlow with the ***tf.matmul*** function.

$$z = w * x + b \quad (3)$$

Where: z =output
 w = weights
 x = inputs
 b =bias

Weights and biases are declared as random normal distributions. Besides, it is also possible to add non-linearities in this layer applying ReLU function.

4. Network Design

Once all the functions to create layers are design, it is just needed to call them and connected them in the proper way, respecting inputs and outputs sizes. It can be seen in Figure 28 how a 3 convolutional neural network is created, but following the same principles, a deeper neural network can be built, or a simpler one.

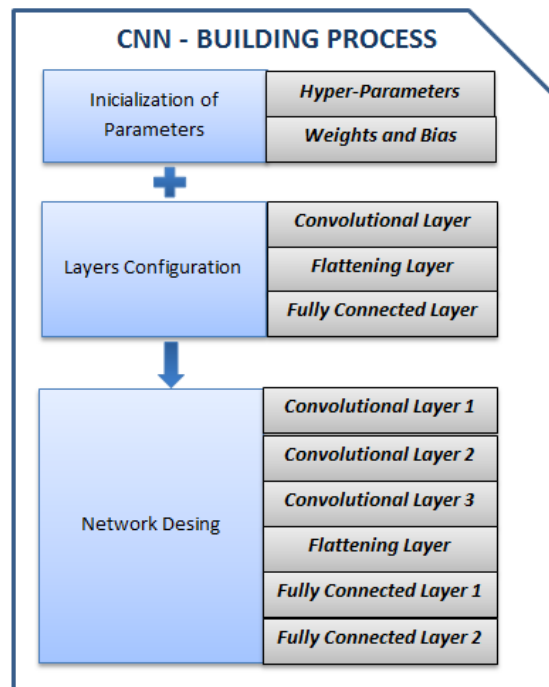


Figure 28. CNN building process

Table 1 shows the values selected for the filters for this application, following the recommendations of Tensorflow [14]. The values selected for stride and max pooling are the most common ones so that the output at the end is exactly half of the input [13]. Point out that a second fully connected layer is needed in other to match the size of the network to the size of the number of output classes that the CNN can classify.

Table 4. Layers Values

Parameter	Value
Filter size – convolutional layer_1	3
Nº Filters – convolutional layer_1	32
Filter size – convolutional layer_2	3
Nº Filters – convolutional layer_2	32
Filter size – convolutional layer_3	3
Nº Filters – convolutional layer_3	64
Fully connected layer size	128
Stride	2
Pooling size	[1,2,2,1]

4.1.2 Making the network functional with TensorFlow

The first step was to build the convolutional neural network. Now, once that it is done, the network is ready to be used, thus, the next step is making it functional for to the problem it has to solve. In a general overview, this means managing the inputs to the network and the output that comes from it.

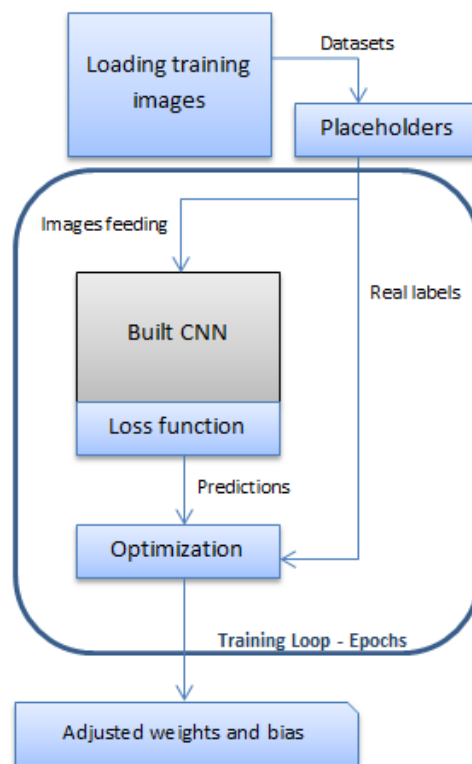


Figure 29. Training process diagram

To make the network functional, as it can be seen in Figure 29, it has to be defined the way of feeding the network with inputs, the way it predicts and how it optimizes the results. This last step of optimization is referred to the learning process. Here the weights and biases are adjusted to try to make the output predictions coincide with the labels of the input images.

1. Feeding and place placeholders

Images are read from a directory and stored in a dataset along with other information. These images are divided into training images and validating images and grouped in batches. The batch size determines the number of images that will be passed to the system in each iteration.

Placeholder will hold the input images for the training. Once the images are read, the input placeholder is created to group and feed with images according to the batch size, the images size and their channels. Likewise, another placeholder is created to store the labels of the inputs.

2. Predictions

To obtain the predictions for the images a loss stage must be added to the output of the fully connected layer. It is normally the final layer. How training penalizes the deviation between the predicted and true labels is specified in this layer. There are different loss functions appropriate for different tasks.

1. **Softmax loss.** It is used for predicting a single class of K mutually exclusive classes.
2. **Sigmoidcross-entropy loss.** It is used for predicting K independent probability values in $[0, 1]$.
3. **Euclidean loss.** It is used for regressing to real-valued labels $(-\infty, \infty)$.

The loss function that adapts better to the desired task for the built network is softmax, which function in TensorFlow is implemented with the command ***tf.nn.softmax***. Thus, for each image there will be as many outputs in form of probabilities as classes and it is interesting to store the highest value.

3. Optimization

During the optimization the goal is to reach the optimum value of weights. For it a cost is calculated using the function ***softmax_cross_entropy_with_logits***. This function takes the output of last fully connected layer and the real labels to calculate the cross entropy. The batch cross entropy average determines the cost.

Then, the cost needs to be optimized. This can be done with an optimization function implemented by Tensorflow, ***AdamOptimizer***, which is useful for gradient calculation and weigh optimization [13]. Here the learning rate is specified, which is a parameter controls how much the weights are adjusted of the network with respect to the loss gradient.

Other important parameter to calculate in order to check the progress of the network is the accuracy on the validation set and the training set. This is done by comparing true labels and predicted labels. As, training images along with labels are used for training, so in general training accuracy will be higher than validation. Training accuracy is reported to know the network is moving in the right direction and is at least improving accuracy in the training dataset. After each Epoch, we report the accuracy numbers and save the model.

4.2 Prediction module

Here the second module of the system will be explained. In this module the necessary steps to carry out predictions over new images are implemented, and a function to save the results automatically in a text file is also developed.

First is needed to load the files created during the training, which will be used to recreate the network and use the trained weights. Then, the input image is read and pre-processed in the same way as in the training. After that, is possible to feed the network with the input and receive the output predicted by the CNN.

Last, the predictions are stored and ordered to create confusion matrixes out of the results and storing the class and the probability of the highest prediction for each image for an easier later analysis.

4.3 Training classes

As the thesis is based on indoor environments, the classes in which the network is trained have to be common objects that can be useful to detect in any indoor environment. For this CNN the learning process carried out during the training is done with a teacher, which means that there are labeled examples of the function to be learned by the network.

To select the classes for training the network, a research elaborated among 20 students in the age range [20,26] was elaborated. In this survey they were asked to say two objects that they search for frequently in their places or other environments where they spend time. The results can be observed in Table 5.

Table 5. Survey results

Object	Selection
Phone	45%
Wallet	35%
Keys	55%
Charger	15%
Remote	10%
Sunglasses	20%
Laptop	20%

Due to the fact the class phone is already taken into account in the dataset used to train the networks that will be used in the second part of the thesis, the previously built CNN will be trained to classify wallets and keys. For this purpose, 273 images (downloaded from Google) of each class are used, in batches of 11. Nevertheless, the network accuracy is closely related to the quality and number of images, and for training a network in the object classification task is needed at least 1,000 images per class [15]. The input data is normally divided into 3 sets:

1. **Training data.** 80% of the total images will be used for this purpose.
2. **Validation data.** The resting 20% of the images will be used to calculate accuracy independently during the training process.
3. **Testing data.** It consists of 20 images for each class. Its purpose is, after the training, to obtain results of the performance of the network with images from an independent source.

Training data and validation data are used during the building and training of the network, while the testing data is used on the prediction module.

Because the final purpose of the thesis is to analyze the systems in real time and in indoor environments, an extra class for the training will be included, “Undetermined (UNDETM.)” or also called “Background”. The aim of this extra class is to help the algorithm to classify better images where there are no objects of the trained classes. This way, a neutral response from the network will be 33% for each class with 3 classes trained, while if there are only 2 classes the neutral response is 50% for keys 50% for wallets, and thus it can be harder to set a threshold value to discriminate false predictions.

The “background” image’s sets are made of the same number of images as the other classes and it contains images of kitchens, living rooms and offices. This should also help the system to classify more precisely the objects and recognize easier the images where there are no objects of the desired ones, keys and wallets.

In order to verify this approach, both trainings will be done, with 2 and 3 classes, as well as a study of the optimal number of layers according to the number of classes, and the best learning rate for the task. Thus, the final result will be two different neural networks that will be called “Keys-Wallet CNN” and “Keys-Wallet-Undetm. CNN” that can classify 2 and 3 classes of objects respectively.

5 Developing Multi-Object Recognition Systems for Mobile Robots

In this chapter, the second main part of the thesis will be explained. Taking the state-of-the-art models for image recognition, the aim is to develop them to recognize multiple objects in an image and to work with real-time images from a camera. These tasks have to be performed while embedded in a mobile robotic platform.

For it, after analyzing the state-of-the-art, the models selected to be developed and later compared are:

- YOLO
- SSD
- Inception

Furthermore, the CNN implemented in the previous chapter will also be developed to perform these tasks to prove how it works on real environments and to check if the network trained on 3 classes does a better job recognizing objects than the network trained just on 2 without a background class.

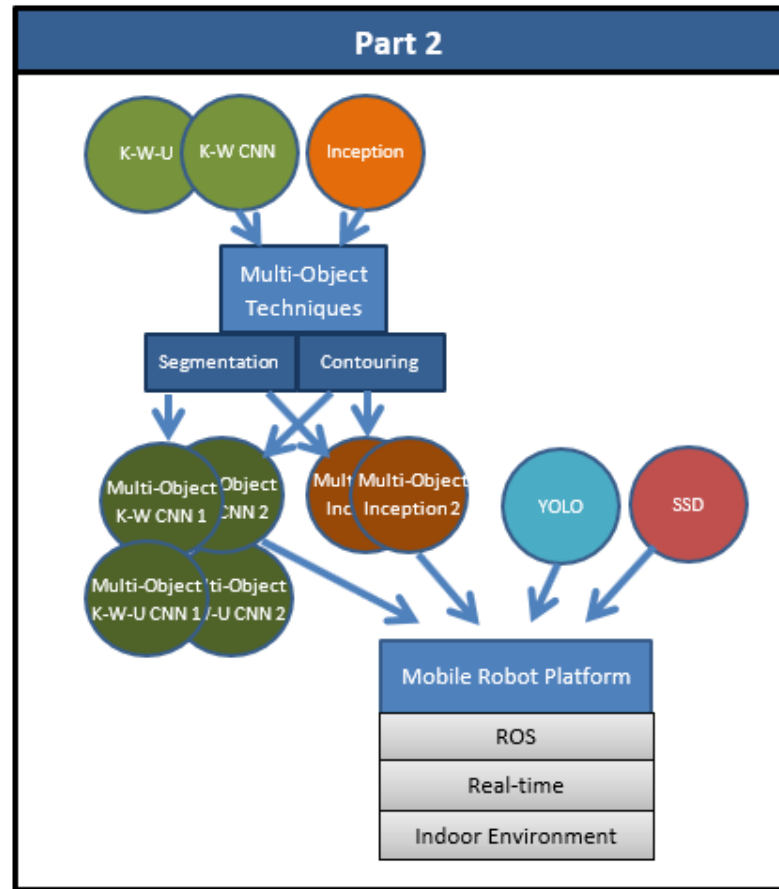


Figure 30. Part 2 schema

Figure 30 shows the state-flow of this second part. Starting with the single object recognition networks and developing them for multi-object detection through two different techniques; then the next step is to adapt all the systems considered for the thesis to a mobile robot platform, which implies making them work under ROS operating system, in real-time and in an indoor environment.

5.1 Selected Models Analysis

The basic models of the already presented network architectures are already implemented and trained. As this field of study is open-source, most of the models can be easily downloaded and implemented for image classification. The main webpage to find information and projects of this kind is <https://github.com/>.

Not all the models give as output all the desired parameters, but after understanding the system and performing simple developments over the image

recognition module, it is possible to return bounding boxes, labels, probabilities and positions of the objects.

Table 6. Networks Characteristics

	Multi-Object*	Real-time*	Classes trained	ROS*	Predictions	Object Location
YOLO	✓	✓	91	✗	1	✗(**)
SSD	✓	✓	91	✗	1	✗(**)
INCEPTION	✗	✗	1001	✗	5	✗
Created CNN	✗	✗	2	✗	2	✗

(*Prepared for it. **Raw Model not returning positions, but locates them)

As it can be seen in Table 6, YOLO and SSD modules are trained in fewer classes than InceptionV3. Thus, the idea behind the comparison is verify if InceptionV3 can overcome YOLO's and SSD's performance despite isn't prepared for real-time nor multi-object recognition but thanks to its more accurate object classification.

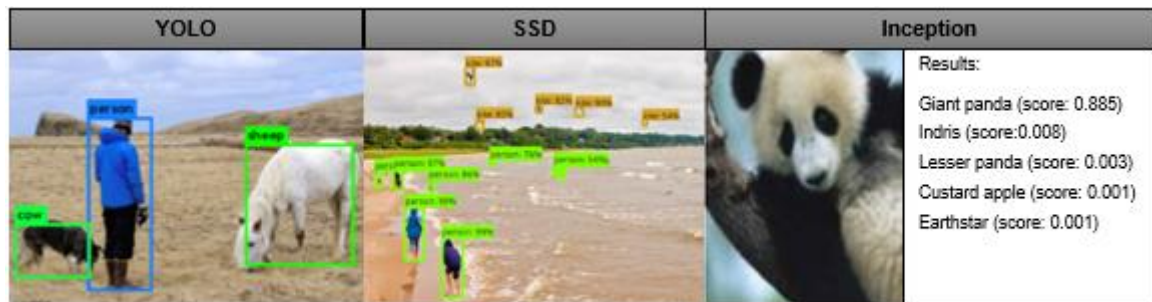


Figure 31. Models Output example

The version used for YOLO is "YOLOV2". This system is originally written in C++ with its library DarkNet, but as this project used Python, the corresponding library in this programming language and translated into Tensorflow commands is called DarkFlow. For the SSD network, the version considered is "SSD_MobileNet_V1", an SSD network developed for mobile and embedded vision applications [30]. Meanwhile, the exact version of the Inception module used is "inception-2015-12-05", which corresponds to the InceptionV3 architecture.

YOLO and SSD are trained with COCO dataset. COCO is an image dataset designed to spur object detection research with a focus on detecting objects in context. The annotations include pixel-level segmentation of object belonging to 80 categories, keypoint annotations for person instances, stuff segmentations for 91

categories, and five image captions per image [5]. The main advantage of this dataset is the use of non-single object image. The objects are labeled inside an image which contains more objects, so the training is more realistic. Due to the fact that the images contain many labeled objects, this dataset is especially useful for algorithms designed to detect multiple objects and classify them.

On the other hand, Inception is trained with ImageNet dataset. ImageNet is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images. The data for the image classification task is collected from Flickr and other search engines. It consists of 150.000 photographs manually labeled by humans into one of the 1000 different object categories that contemplate the datasheet [1]. Due to the presence of one object per image, this dataset is great for algorithms that can classify single objects, producing 5 class labels in decreasing order of confidence.

The organizations behind the creation of these two datasets also launch competitions for computer vision systems. In the field of image vision, the main advances are shown and proved in this kind of image recognition contests.

5.2 Multi-Object implementation

The target networks for this development are InceptionV3 and the built CNN. The aim of this work is to make the network detect different objects within one image and locate them. From the state-of-the art two main ideas are considered to accomplish this task.

The first one consists of segmenting the input image in several of them, principle in which R-CNNs are based [45]. The second approach encompass preprocessing of the input image, manipulating colors and other parameters to detect contours and thus different possible objects in the image. This technique is based on the ideas developed by Pablo Arbeláez et al. in their work “Contour Detection and Hierarchical Image Segmentation” [75], and a similar procedure was used by Jerome Paul N. Cruz et al. to recognize and detect objects by shape and color pattern [54].

5.2.1 Method 1: Image segmentation

The first approach to realize multi-object recognition consists of dividing the input image in 8 quadrants, crop the images of each quadrant and pass them to the object classification module (see process in Figure 32).

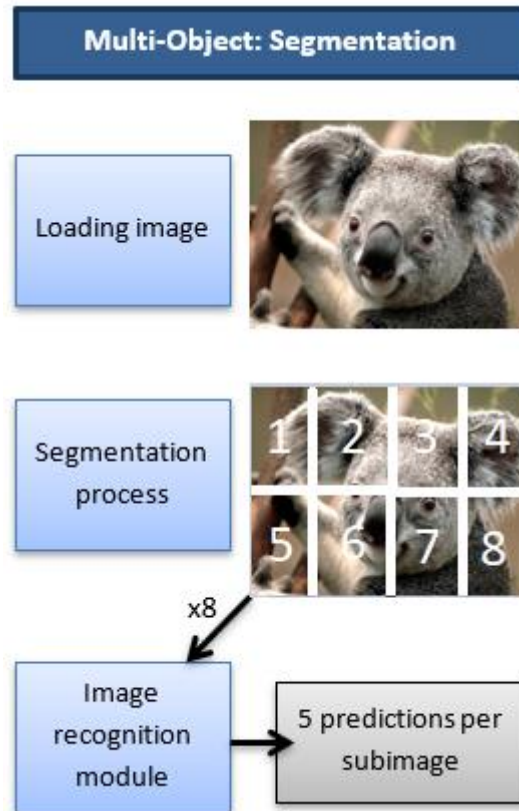


Figure 32. Segmentation classifying process

The process of loading the image and cropping it is done with OpenCV. With this technique 8 objects with their 5 highest probabilities of classes can be recognized and located in their respective quadrant. This method allows the system to easily overcome the multi-object issue with InceptionV3 module without having the main problem of RCNN, which is the processing time.

Intuitively the system will run 8 times slower than the normal one, as 8 images are passed instead of one to the image recognition module. But as the images are smaller, and size is a parameter that influences the processing time, the system will not be as slow as it could be expected, so it can still be used later for real-time image recognition.



Figure 33. Segmentation output example

An example of the multi-object recognition modules output can be observed in Figure 33. For every quadrant 5 predictions are given with their probabilities ordered from bigger to smaller, the same way that each quadrant is identified for a possible localization of the objects inside the original input image.

5.2.2 Method 2: Contours Detection

This second technique is based on detecting contours in the input image. For every image processing OpenCV library is used. To make this method comparable with the previous one, the number of contours to detect will be limited to eight.

After exploring different ways to identify contours, the one that brought better results in a preliminary study was the one that is going to be explained next. It is based on filtering by color range and then searching for contours.

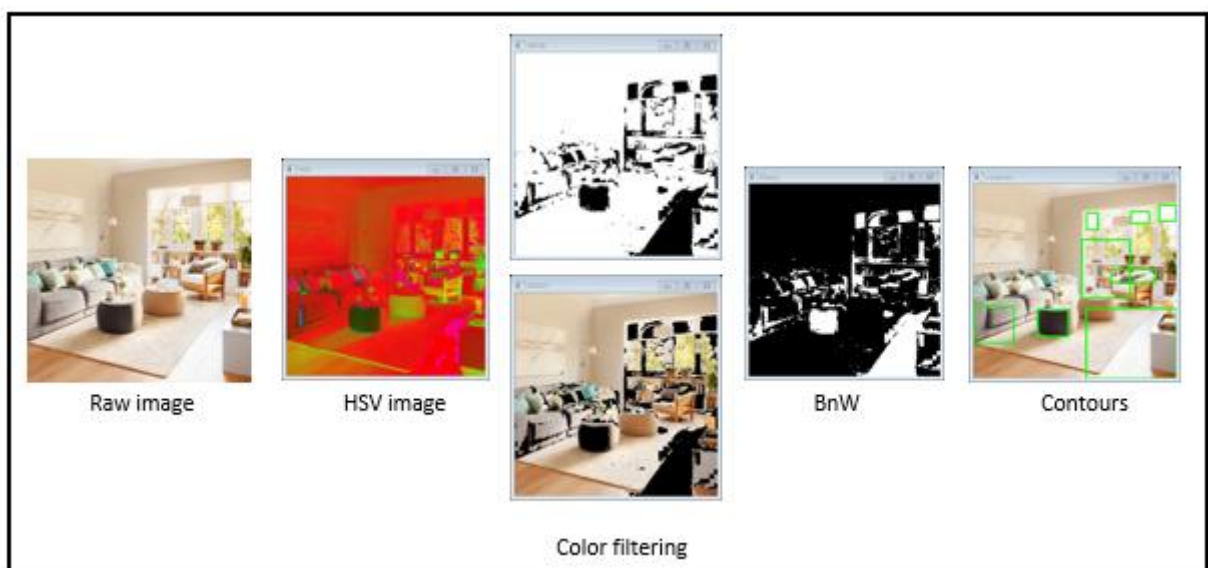


Figure 34. Contour Detection Process

In first place, the image is loaded and sized down. Then, the color scale is changed from RGB to HSV. The next step is filtering this pre-processed image according to upper and lower color intervals, for later being transformed into a binary color image. Finally, over that black and white image contours are found. The process can be observed in Figure 34.

The algorithm is designed to store the 8 biggest contours detected and its location, discarding the contour that englobes the whole image. This can lead to inaccurate object detections as the small objects are not prioritized. Besides, light and brightness have a big influence when detecting contours with this methodology, which it can result in false positives.

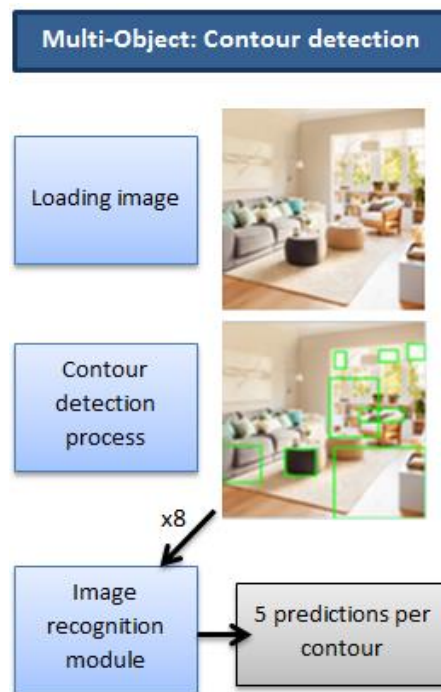


Figure 35. Contour classifying process

The contours detected from the original input image are cropped and sent to the image recognition module to obtain their predictions. In the same way as it occurs with the previous method, this technique allows to classify multiple objects without transforming the system into a very slow recognition module. A general overview of the system functionality can be seen in Figure 35.

Compared to the segmentation process, the contour detection method presents acceptable times of analysis to be used in further real-time applications, although due to the greater complexity of the pre-processing of the images to find the contours, this technique is slightly slower than segmentation. Nevertheless, the output

predictions give the same information, being the contour detection more precise with the object localization coordinates (see Figure 36)



Figure 36. Contour detection output example

The information is presented by contours, identifying each contour in the image and in the output window. For each contour it can be observed that the top five predictions are shown with their confident. In addition, by the contour number label, the localization of it is described by four coordinates that indicates the position of the top left bounding box corner with respect to the X-axis and the Y-axis, as well as the width and height of the box, [x,y,w,h].

5.3 Mobile Robot Platform integration

The last step to make the system fully operative to work in a mobile robot platform is to implement the object recognition systems into ROS and make operate in real-time. For it, is needed to take the previous ideas into a new operating system.

First is needed to adapt all the multi-object recognition modules to read from the mobile robot camera instead of reading feed static images from a folder. This is done with OpenCV, it connects the module with the camera and captures frames from it at a certain rate. This rate is known as FPS (frames per second) and is important to control the FPS with which the camera is operating and the FPS that the system is analyzing. To achieve a better performance is important to balance both numbers; otherwise the FPS of the image recognition system will not be as fast as it could be.

Each frame is pre-processed to be compatible with the multi-object recognition module and furthered analyzed. This process is inside a loop that repeats while the camera is sending images and the predictions for the frames are stored, but before the code gets into the loop, the Tensorflow session and graph must be initialized,

otherwise the system will consume too many resources and time creating new ones for every new image.

So far, the systems are able to operate in Ubuntu and Window environments. Now ROS comes into play and its way of working is different, it is based on nodes, and thus the code has to be adapted to work with nodes.

For each system 2 nodes will be used:

- **Camera Node.** This node has the information that the camera gets through its lens.
- **Object Recognition Node.** This node gathers information about the nodes interaction and the functions needed to perform the desired tasks. It determines the topics to be published and the nodes to which be subscribed.

The Object Recognition Node consists of 3 main parts. The first part is where the main parameters of the node are established, here the TensorFlow session and graph are created, values to variables are assigned, and the subscriber and publisher topics are declared as well as the type of message that they carry and the actions that they trigger. The subscriber topic is design to trigger the next part of the node, the classification function. In the second part the image processing and analysis is carried out, all the needed functions developed are used or called in here to perform the multi-object classification from the images coming from the subscriber node. Finally, the third part which creates a loop, so the node keeps operating till the system is shutted-down.

The Object Recognition Node is subscribed to the Camera Node Topic “Camera/rgb/image_raw” that feeds the system with images, and it publishes the topic “Results”, so that other nodes can subscribe to it (see Figure 37).

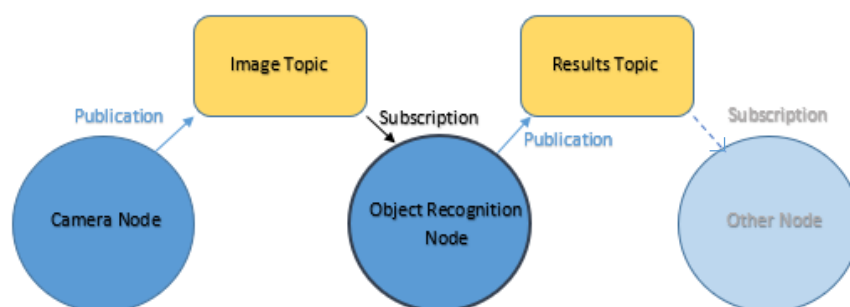


Figure 37. ROS system representation

6 Experimental Results

This chapter evaluates the developed systems by checking its performance in different scenarios. These experiments will help to determine the correct functioning of the systems of this work, as well as finding out which systems perform better in a mobile robot platform for the desired task, multi-object recognition in real-time applied to indoor environments.

The division of this chapter into four sections provides a better comprehension of the results of this work. In the first place, the hardware of the experimental platform is described separately. Secondly, the work environment is presented by explaining scenarios of the experiments. Later, the experimental results of the two main parts of the work are presented, the first part where the YOLO, SDD and InceptionV3 are tested, and a second part where the two bests design of the built CNN is evaluated, one design for the 2 classes classifier and another for the 3 classes classifier.

6.1 Experimental Platform Hardware

In this section the robotic platform used in this work is presented. The hardware elements used for the experiments of this master thesis are described as follows.

6.1.1 Turtlebot Mobile Platform

Turtlebot is a low-cost mobile platform which provides a personal robot kit with open-source hardware and software. It allows access to lower information level, so it is especially indicated for algorithm research and testing. This mobile platform was developed in November 2010 at Willow Garage by Melonee Wise and Tully Foote [17].

The second version of this robot will be used, Turtlebot 2 [17]. The Turtlebot 2 kit is composed by a kuboki base, 2C/3D sensor camera, laptop computer or single board computer, a 2200mAh battery pack, Kinect mounting hardware, fast charger, charging dock, Turtlebot structure and Turtlebot module plate with 1-inch spacing hole pattern. In addition to the hardware, the robotic software development environment is included. The software and functional specification of the Kobuki base can be seen in Table 7.

Table 7. Functional and software specifications [6]

Functional specification		
Maximum translational velocity		70 cm/s
Maximum rotational velocity		180 deg/s (>110 deg/s gyro performance will degrade)
Maximum payload		5 kg (hard floor), 4 kg (carpet)
Inclination threshold	Negative	5 cm
	Positive	12 mm
Expected operating time		3/7 hours (small/large battery)
Expected charging time		1.5/2.6 hours (small/large battery)
Software specification		
C++ drivers for Linux and Windows		
ROS nodes		
Gazebo simulation		

6.1.2 ASUS RGB-D Camera

Xtion PRO Live by Asus is the depth sensor camera used for the experiments of this master thesis, as well as an integrated device in the robot. This camera is design to detect movement and broadcast live video, providing depth images, RGB images, and a microphone array. For that reason, it possesses the functionalities of gestural detection and full body detection, RGB, and audio, and it has the advantage of being powered by USB connection [20].

6.1.3 Base Computer

The Turtlebot 2 Kit provides a base computer, a Toshiba KIRA laptop with a core i7 processor. This laptop is strongly suitable for this application as it has small dimensions to fit in the mobile platform and a good processing and graphical capabilities. For this work, this base computer works under ROS Inigo Igloo and Linux with Ubuntu 14.04 LTS (Trusty Tahr).

6.2 Work environment

To test the different systems, the mobile robot platform will be moved around the indoor environment of a house. The experiments will be performed in five scenarios of the house: Kitchen, Living room, Bathroom, Bedroom and Office.

The illumination conditions or object placement has not been modified for a better performance. During the experiments realization, there was no human interaction with the objects, the scenario is static, but the mobile robot platform moves around it.

6.3 Part 1: CNN Results

In this section the results regarding the CNN are presented. First, the results that corresponds to the training are shown. Twelve experiments are carried out in order to select the best network configuration, one for the Keys-Wallet CNN and another one for the Keys-Wallet-Undetm. CNN.

Secondly, with these best networks' configurations, real-time tests are done embedded in the mobile robot platform. The performance is evaluated for the task of single-object recognition and multi-object recognition.

6.1.1 Training Experiments

This section details the different experiments done during the training and its purpose. A total of twelve experiments are done and analyzed. Keeping constant the parameters shown in Table 8, the aim of these tests is to evaluate the following fields:

- **Design.** CNNs of one, two and three convolutional layers are tested.
- **Learning Rate.** The learning rate is varied in order to see how it affects to the performance of the network and its training.
- **Number of classes.** Systems trained in 2 and 3 classes are compared on the testing data set.

Table 8. Constant CNN parameters

Parameter	Value
Batch_size	11
Validation_size	0.2
Img_size	128
Num_channels	3
Filter_size_conv	3
Num_filters_conv	32 (64 for conv3 layer)
Fc_layer_size	128
Layer_conv_strides	[1,1,1,1]
Layer_maxPool_strides	[1,2,2,1]
Layer_maxPool_ksize	[1,2,2,1]
Layer_padding	'SAME'
Weighs & biases_stdDev	0.05
Epoch	50

Thus, the experiments encompass six tests for each functionality, where for each network configuration of one, two and three layers, two tests are carried out with different learning rates (LR). Is understood as functionally the ability of the network to classify 2 or 3 different objects classes.

The results compare the performance of the different networks configurations in three categories:

- **Training progress.** Training accuracy, validation accuracy and validation loss progress during the training are compared to verify which architecture has the best knowledge acquisition and to check how the Learning rate influences these curves.
- **Precision and sensitivity.** Evaluating the results obtained from the images belonging to the testing dataset is possible to elaborate confusion matrixes for each configuration to compare them and to observe how the networks classify. Furthermore, with that information, it's possible to evaluate the

precision and sensitivity of the network.

- **Accuracy results.** From the previous results over the training dataset, Box-and-Whisker plots can be created to measure statistically the accuracy of the classes predicted. Thus, it can be obtained an approximate idea of the needed threshold value for the predictions in order to eliminate false positives.

		True condition	
		Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive, Power	False positive, Type I error
	Predicted condition negative	False negative, Type II error	True negative

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad \text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Figure 38. Confusion matrix evaluation [28]

From Figure 38 it can be observed how a confusion matrix can be formed ordering the predicted conditions in the corresponding true conditions columns. Besides, the basic terms to identify the different predictions cases can be easily recognized in function of the predicted condition and the true condition, and from these results the equations to calculate precision and sensitivity are exposed.

6.1.2 Training Results

Here the results of the CNN training and offline testing are presented for the different experiments carried out. The analysis es done for each of the previously mentioned categories, to finally choose the best network configuration for the Keys-Wallet CNN and for the Keys-Wallet-Undemt. CNN

Firstly, analyzing the training progresses in Figure 39 and Figure 40, where the training accuracy, validation accuracy and validation loss percentages evolution are plotted in the Y-axis during the different iterations or epochs, it can be observed how in general, the more the layers the networks have, the longer it takes to reach high training accuracy. The learning slope is steeper when using a bigger learning rate.

Furthermore, this learning process takes longer when training three classes in comparison with 2 classes and the validation accuracy oscillates more, and so the validation loss does.

In terms of time spent during the training phase, as a general observation, it can be said that the more layers the network has, the training takes longer. This time increases also when decreasing the learning rate. The same happens when the number of classes trained are incremented. This last fact is directly related with the time spend because it increases the number of images to process and the clustering of the features of the images from the network becomes a more complex task. The test performed during this thesis took between 15 and 30 minutes to be completed.

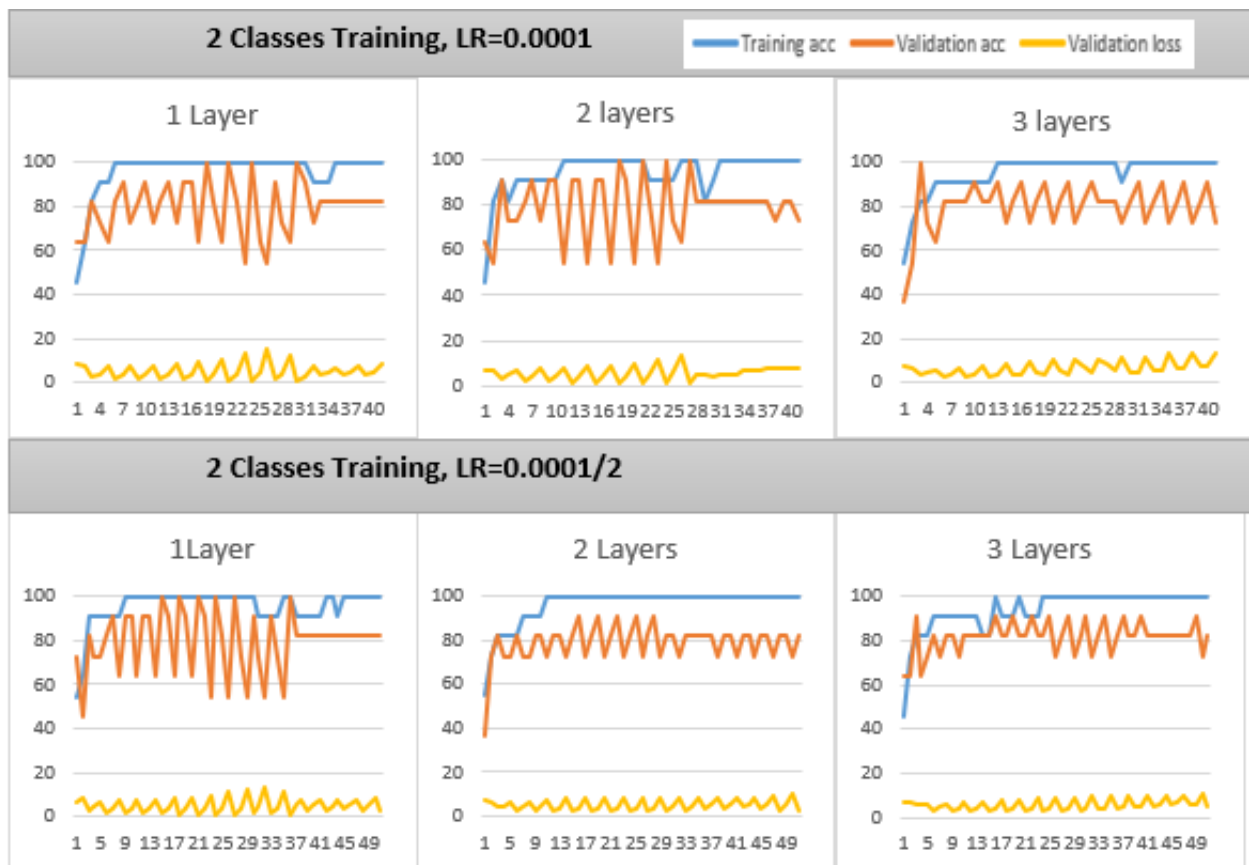


Figure 39. 2 classes training progress

Point out that in all the cases the training accuracy reached the 100%, and the validation accuracy oscillates around the 80%, stabilizing close to that value for the 2classes network, while in the 3classes network does not seem to stabilize in the training period evaluated.

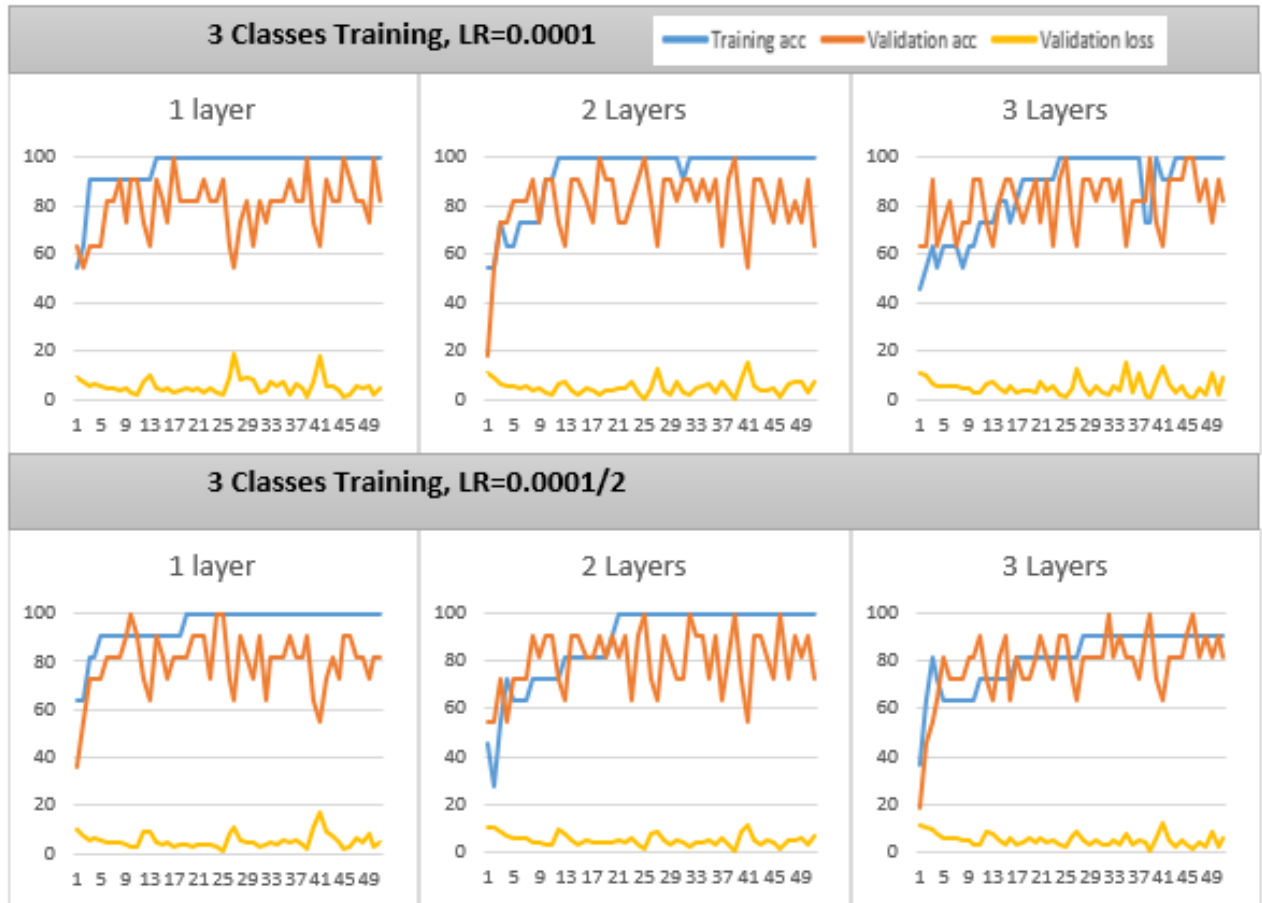


Figure 40. 3 classes training progress

When analyzing the confusion matrixes precision and sensitivity, the design that better results gives for the 2 classes classification, considering the sum of both learning rates results, is the 2 layers network results (see Table 9). More specifically, the 2 layers network trained with the lower learning rate gave better overall accuracy and sensitivity results than the network with a greater learning rate. Furthermore, the network had better result classifying keys than wallets in all the tests, as it can be seen in higher keys sensitivity and thus, influences to have high wallet precision.

Table 9. 2 classes precision and sensitivity results

		LR=0,0001			LR=0,0001/2		
		Layers			Layers		
		1	2	3	1	2	3
Wallet	Precision	100%	100%	80%	93%	94%	89%
	Sensitivity	70%	70%	60%	70%	80%	80%
Keys	Precision	77%	77%	68%	76%	83%	82%
	Sensitivity	100%	100%	85%	95%	95%	90%

For the 3 classes configuration the conclusions are different (see Table 10). The 3 layers configuration with a the higher LR, overcomes the other setting. For that

learning rate, it can be observed how the results improve when adding layers, while with decreasing the learning rate, the results get worse with a more complex network.

In this case, the class with more true positives is the Background/Undetem. Class, which can be understood from a high sensitivity and precision percentage average; and opposite to the 2 classes network, wallets are better classified than keys. This can be observed in a higher percentage of wallet sensitivity than keys sensitivity in almost every test performed.

Table 10. 3 classes precision and sensitivity results

		LR=0,0001			LR=0,0001/2		
		Layers			Layers		
		1	2	3	1	2	3
Wallet	Precision	82%	71%	90%	100%	90%	89%
	Sensitivity	70%	75%	90%	70%	90%	80%
Keys	Precision	68%	70%	88%	65%	83%	71%
	Sensitivity	75%	70%	75%	80%	75%	75%
UNDETM.	Precision	86%	89%	87%	83%	81%	81%
	Sensitivity	90%	85%	100%	95%	90%	85%

After evaluating all the results and considering the conclusions exposed before, the best designs and the ones that will be tested on the mobile robot platform are:

- **2 Classes Network:** 2 layers and 0.0001/2 learning rate.
- **3 Classes Network:** 3 layers and 0.0001 learning rate.

Comparing both of them in terms of keys and wallet classification, the network that presents better overall accuracy and sensitivity results is the 2 classes Network.

While the network configurations that present worst results of precision and accuracy of the classes to be detected are:

- **2 Classes Network:** 3 layers and 0.0001 learning rate.
- **3 Classes Network:** 2 layers and 0.0001 learning rate.

For these four configurations, the accuracy results will be analyzed. These results are achieved by ordering the predictions accuracy of each image depending on their class dataset. The results have been organized in box and whisker plots in order to analyze statistically the accuracy, where each sub-plot corresponds to a dataset, and the Y-axis represents the percentage of each prediction, and the X-axis the classes predicted.

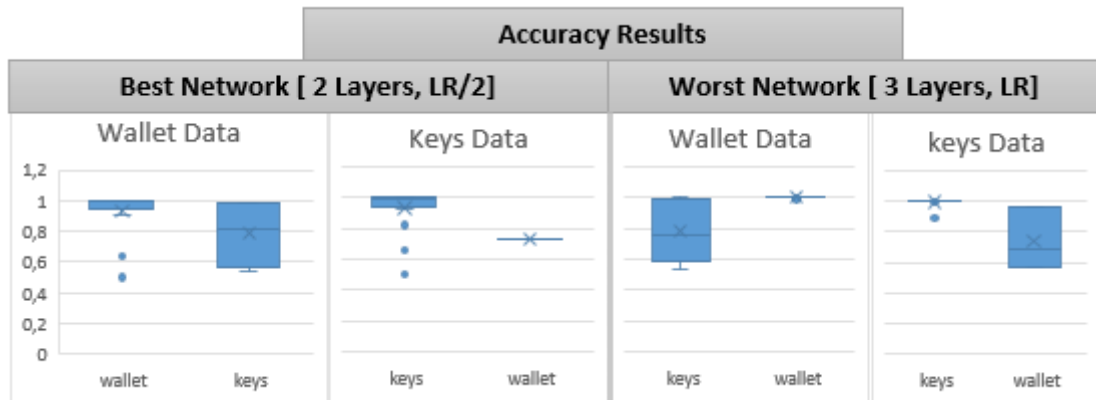


Figure 41. 2 classes networks accuracy results

By looking at the best networks accuracy results in Figure 41 and Figure 42, it can be seen that using a threshold equal to the value of the 1st quartile, most of the false positive results are dismissed. The targeted classes present higher accuracy and precision results with some point exceptions, while the other classes accuracy occupies a wider and lower range.

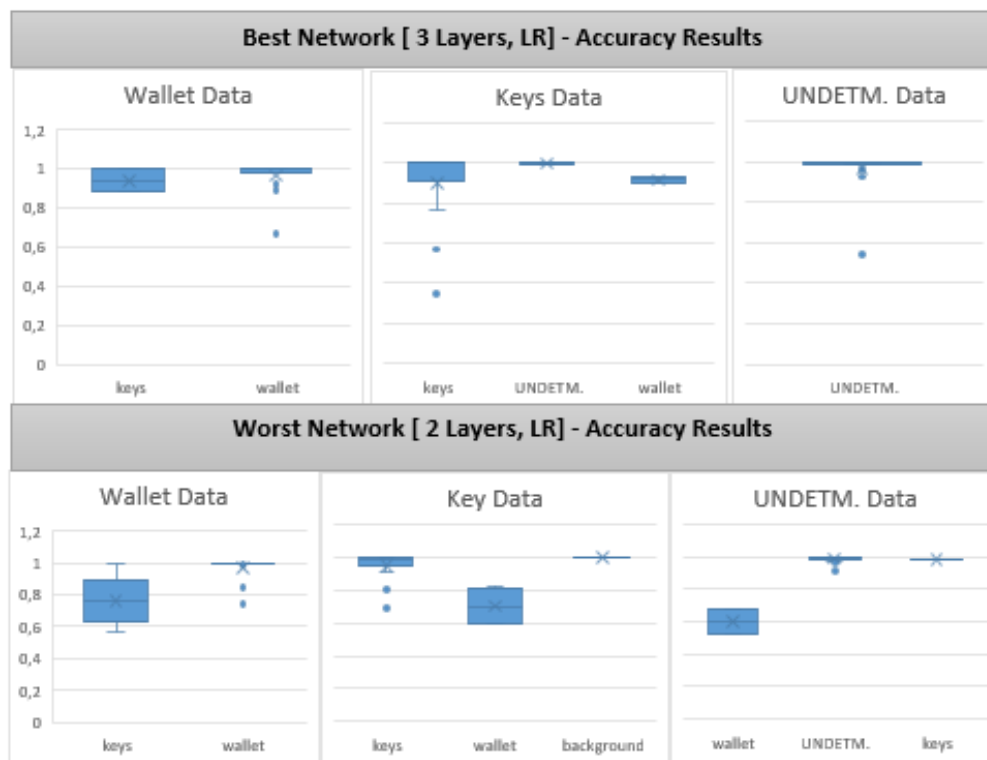


Figure 42. 3 classes networks accuracy results

Comparing the accuracy of the best and worst network configurations, it can be observed how in both image recognition networks, the worst network presents better accuracy and precision for the targeted class when it manages to classify it correctly.

Besides, similarly occurs with the not targeted classes, which is an indicator of a worst separation of features of the images for its classification, and thus it can easier lead to misclassifications.

Furthermore, it can be observed how the wallet data sub-plot of the keys-Wallet-Undetm. CNN does not present the predicted class of “UNDETM.”. This is explained because no images of that dataset had an “UNDETM.” output. This phenomenon occurs also in the Undetm. Dataset, which lead to the 100% of sensitivity reached for this class (Table 10), and along with the previous fact, it leaves the undetm. false positives obtained in the keys dataset as the only responsible of the precision decrease of the undtem. Class.

6.1.2 Developed CNN testing and discussions

Here the performance of the built and trained networks to classify keys and wallets, and its variant, which detects keys, wallets and background, is compared on real time images. This comparison is done with 3 tests, the first one where the objects trained to detect are shown to the camera, as well as a background image without any of those objects. A second test where the contour detector is implemented and the performance with the network is checked, and finally the segmentation technique for multi-object detection is also verified.

In Figure 43 is possible to observe how both configurations classify successfully the wallet with high confident, the keys is also correctly labelled although the 3 classes network gives better accuracy, and when coming to an open scenario, the 2 classes network interpret it wrongly as a 93 % wallet, while the other network recognize it as a background. This is important to notice, as images without the trained objects may be misinterpreted, but when adding a neutral class, the probabilities to have false positive classification decreases as it can be observed in this test.

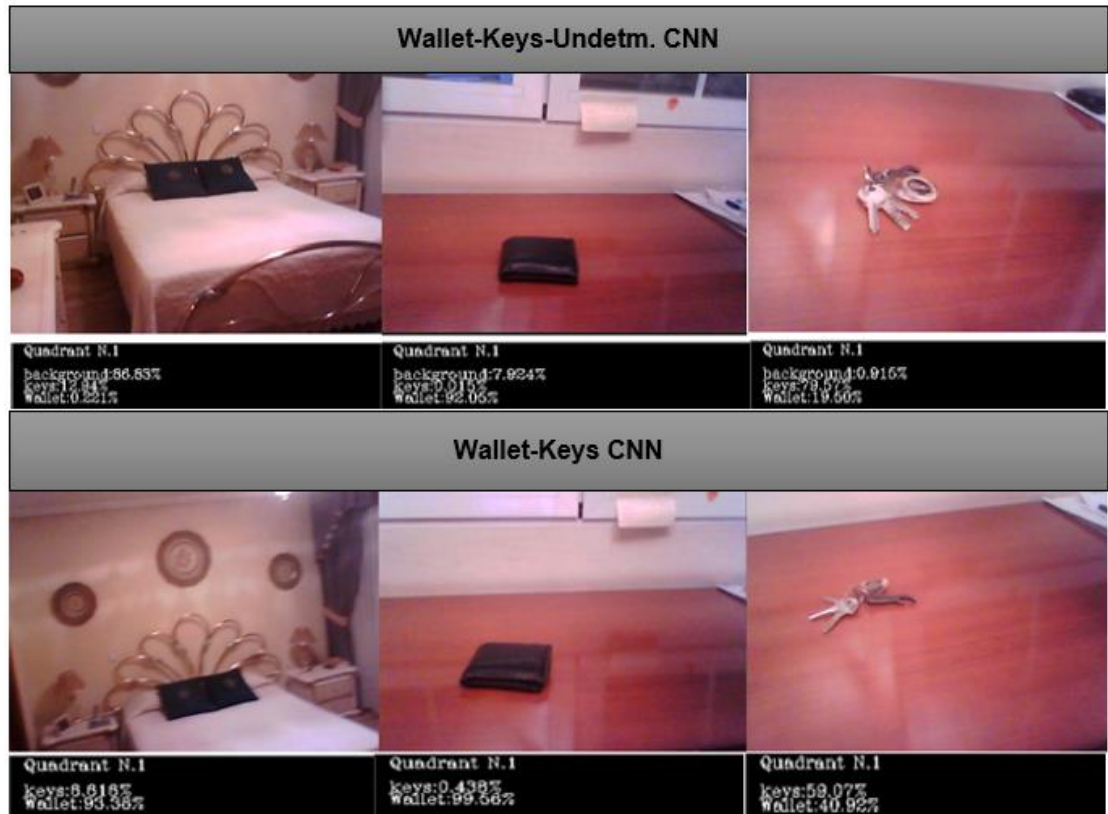


Figure 43. Single object detector test

When comparing the networks with the contour detector over a scenario with both target objects and another scenario without them, the results obtained can be seen in Figure 44. The 3 classes network in both images detects and classify properly the wallet and the background with the contours number one. Point out that even if the other contour results are wrong, its confident is lower than the first contour confident, so they can be easily dismissed with a higher threshold value. Nevertheless, the system fails to detect the keys and so to classify them.

With the other network, the wallet is properly classified on the second contour with a really with probability, while the key is not even detected by the system. On the bathroom scenario, the system identifies wrongly wallets and keys with high confident, which makes impossible to set a threshold value high enough to discard those outputs without discarding also the correct ones.

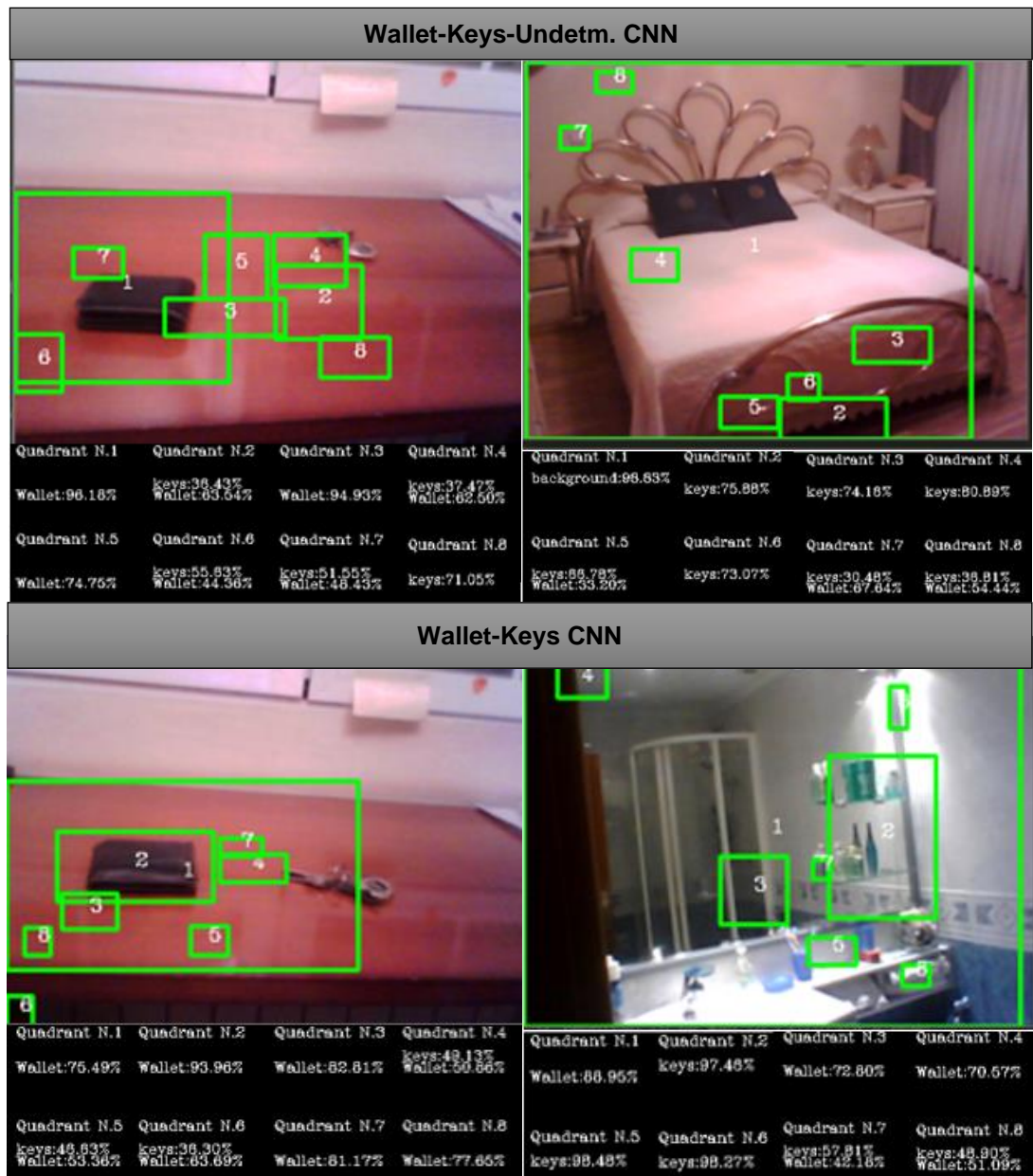


Figure 44. 2 contour classifier tests

In Figure 45 the results obtained for the segmentation module of both networks can be observed. The output from both network configurations doesn't bring good results, quadrants without the objects are recognize as the objects with higher confident than the quadrants with them, so there is no possibility of eliminating the false positive results with a threshold value. Nevertheless, the 3 classes network recognize properly some background images, and the other parts of the main images don't throw as high probabilities for the other 2 classes as the other network configuration.

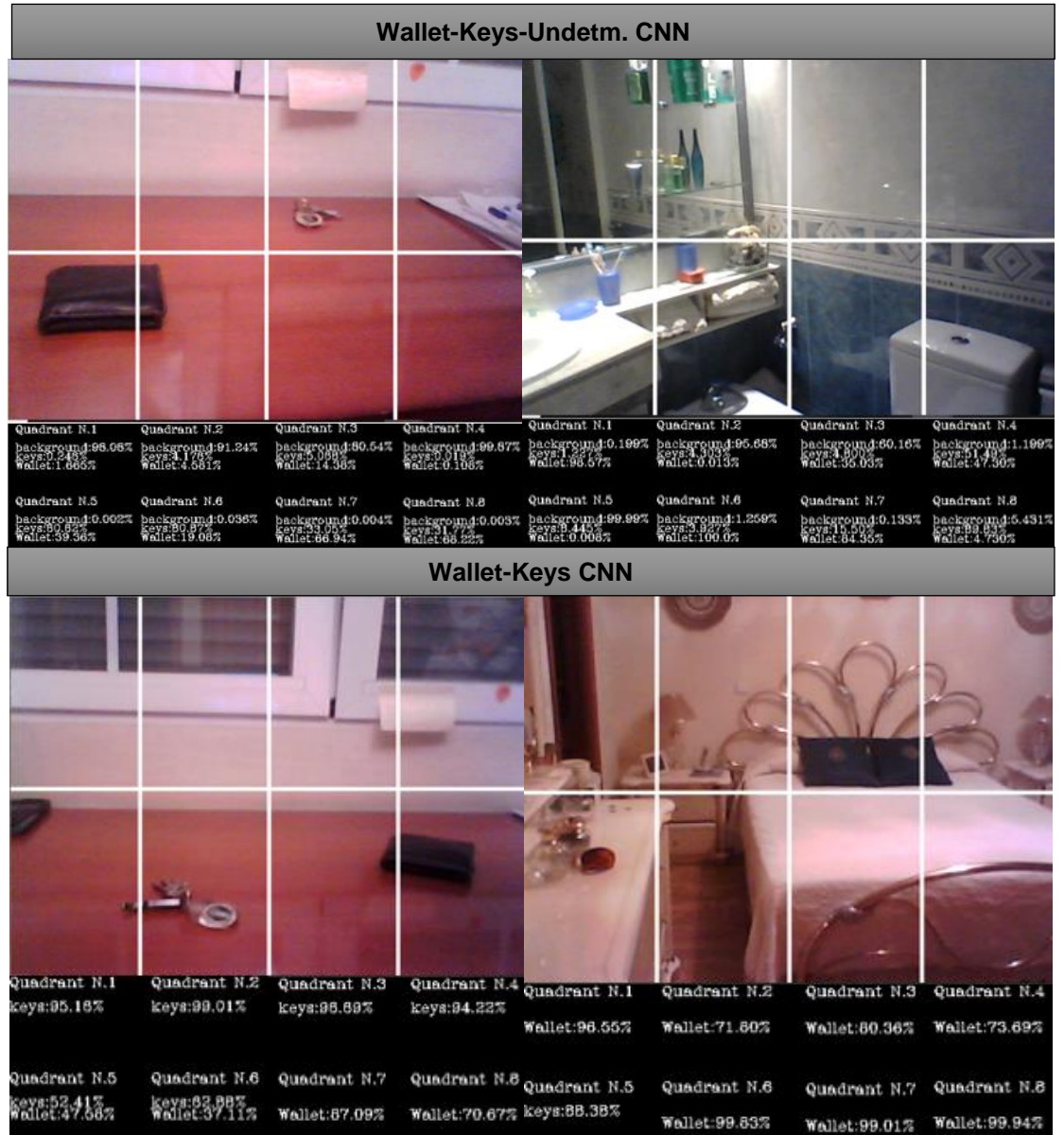


Figure 45. 2 Segmentation classifier tests

After analyzing the different results of both networks in different scenarios and applications, it can be said that adding a third class to classify objects as undetermined when the network can not classify them has advantages when working in real environments.

6.4 Part 2: Multi-Object System

In this section the results of the second part of the thesis are evaluated. Divided in the five scenes that conform the house indoor environment, each system is analyze and compare to verify which one obtains better results. The different images shown in the next sections corresponds to screenshots of the live video that the mobile robot platform is analyzing.

- # Kitchen

Firstly, comparing YOLO and SSD networks, which are trained on the same dataset, COCO dataset, it can be observed in Figure 46 how YOLO detects more objects and with a higher confident, an average of 60% in contrast of an average of 30% for each prediction.

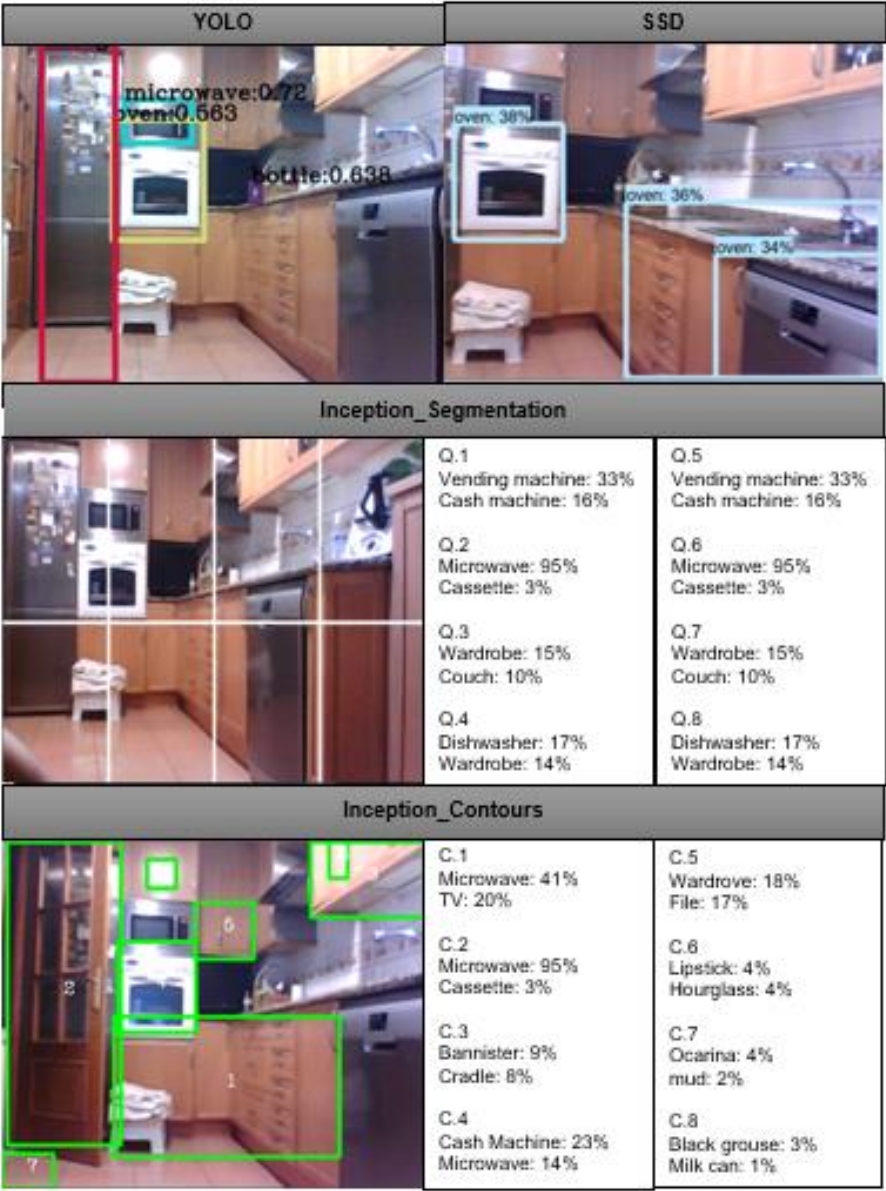


Figure 46. Kitchen test

Analyzing both InceptionV3 multi-object detectors, the first conclusion that is possible to obtain is that the contour detector doesn't recognize properly the different objects, it just bound correctly the door and the oven, besides it classifies it as a cabinet and a vending machine. On the other hand, the segmentation system throws better results detecting and classifying correctly the wardrobe, microwave and dishwasher.

Between YOLO and InceptionV3_segmentation the system with the best performance is YOLO. It detects objects with higher confident and with just one mistake. Still the objects detected are enough to classify the scene as a kitchen, while with InceptionV3, the low confident and misclassification may lead to scene misunderstandings.

• Livingroom

Following the same comparison procedure as before, it can be seen in Figure 47 how SSD is not capable of detect as many objects as YOLO does. Furthermore, its confident is lower and it classifies wrong more objects. YOLO's detections are more accurate and greater in number and most of the mistakes have less than 50% of confident.




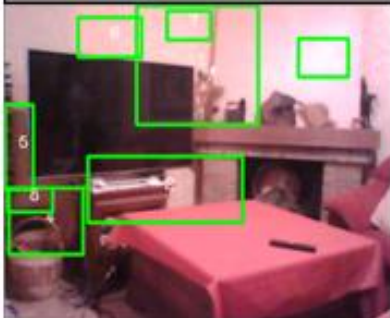
YOLO	SSD	
		
Inception_Segmentation		
	Q.1 Missile: 10% Half Track: 5%	Q.5 Couch: 12% Stretcher: 5%
	Q.2 Plate Rack: 15% Guillotine: 10%	Q.6 Couch: 93% Throne: 2%
	Q.3 Television: 35% Couch: 7%	Q.7 Restaurant: 60% Home Theater: 11%
	Q.4 Couch: 61% Television: 16%	Q.8 Couch: 89% Quilt: 2%
Inception_Contours		
	C.1 Microwave: 27% Sewing Machine: 10%	C.5 Whiskey jug: 12% Milk can: 5%
	C.2 Couch: 47% Barber chair: 4%	C.6 Face powder: 3% Nematode: 2%
	C.3 Velvet: 13% Geyser: 3%	C.7 Tick: 9% Black grouse: 3%
	C.4 Oxygen mask: 4% Oboe: 4%	C.8 Curtain: 20% Barrel: 6%

Figure 47. Livingroom test

InceptionV3_segmentation doesn't perform well at all in the livingroom environment, classifying correctly only the sofas and the table, and confusing the painting with a TV, as SSD does. The contour detector version of this system fails to detect properly the objects and thus the classification of them is wrong and with low confident.

For a living room environment, it can be said that YOLO is the best network. The objects detected and the confident with which they are classify can easily lead to understand that surrounding environment is a living room.

• Bathroom

When the mobile robot platform is taken to a bathroom environment, the results obtained with the different systems coincide in a high confident toilet detection and classification.





YOLO	SSD
	
Inception_Segmentation	
	<p>Q.1 Speedboat: 15% Trimaran: 3%</p> <p>Q.2 Medicine chest.: 16% Stove: 13%</p> <p>Q.3 Trailer Truck: 16% Halftrack: 10%</p> <p>Q.4 Soap dispenser: 9% Washer: 6%</p> <p>Q.5 Upright: 8% Microwave: 5%</p> <p>Q.6 Bathtub: 13% Washbasin: 9%</p> <p>Q.7 Toilet seat: 63% Washbasin: 22%</p> <p>Q.8 Toilet seat: 79% Soap dispenser: 7%</p>
Inception_Contours	
	<p>C.1 Mosque: 2% Harmonica: 2%</p> <p>C.2 Phone: 20% Cash Machine: 8%</p> <p>C.3 Toilet seat: 61% Soap dispenser: 5%</p> <p>C.4 Mousetrap: 9% Prison: 7%</p> <p>C.5 Face powder: 31% Oil filter: 15%</p> <p>C.6 Face powder: 3% Nematode: 2%</p> <p>C.7 Pedestal: 7% Wardrobe: 5%</p> <p>C.8 Geyser: 63% Fountain: 33%</p>

Figure 48. Bathroom test

SSD and YOLO are both capable of recognizing bottles in the image. Furthermore, YOLO performs great detecting 2 toothbrushes, while SSD doesn't but instead it detects correctly the sink. Again, the confident shown by YOLO is greater than SSD's.

The contour detector works correctly detecting the toilets, but the other contours are quite unprecise or irrelevant. Moreover, the segmentation system for the InceptionV3 module has the same problem for classifying the quadrants where toilets are not present.

Both SSD and YOLO are appropriate systems for this environment, presenting similar results, while InceptionV3, despite its wider range of objects knowledge, fails to detect more or different objects with precision.

• Office

In an office environment, SSD is clearly overcome by YOLO. This last system detects correctly the chair, books and monitor, and has a comprehensible mistake confusing part of a printer with a cell phone, while SSD only detects the chairs and not with a high confidence (see Figure 49).

Illumination again plays an important role in the contour detector system, which it can be appreciated with the detection of two contours on the window spot. Furthermore, the object detection in general doesn't perform too good, detecting only properly the contour of the chair and the table, but failing classifying the first while succeeding with the second.

The segmentation system fails also to recognize the objects in the different quadrants. Despite that, the confident of its results are at least not really high so they can be easily dismissed just by adjusting the threshold value, but in that case nothing would be detected.




YOLO	SSD
	
Inception_Segmentation	
	<div><div>Q.1 Pool table: 7% Barber chair: 6%</div><div>Q.2 Projector: 11% Home theater: 6%</div><div>Q.3 Stove: 9% TV: 6%</div><div>Q.4 Balance beam: 5% Parallel bars: 2%</div></div> <div><div>Q.5 Chest: 12% Plane: 9%</div><div>Q.6 Tape player: 20% Stove: 11%</div><div>Q.7 Stove: 45% Tape player: 14%</div><div>Q.8 Desk: 11% Upright: 7%</div></div>
Inception_Contours	
	<div><div>C.1 Desk: 31% File: 16%</div><div>C.2 Beer glass: 5% Violin: 3%</div><div>C.3 Ocarina: 14% Ruler: 8%</div><div>C.4 Fountain: 99% Fireboat: 1%</div></div> <div><div>C.5 Slide ruler: 9% Panpipe: 3%</div><div>C.6 Geyser: 34% Beaker: 3%</div><div>C.7 Geyser: 20% Beaker: 11%</div><div>C.8 Black grouse: 64% Chicken: 9%</div></div>

Figure 49. Office test

For these reasons, the systems that better results throw in an office environment is YOLO. Besides, the different classified elements can be typically found in and office scene.

• Bedroom

The results obtained after moving the mobile robot platform around a bedroom show how YOLO and SSD success in detecting and classifying the bed, while both developed InceptionV3 systems fail in that task, as it can be observed in Figure 50.

Due to the special form of the bed, the first two system detect scissors where they are not. On the other hand, Inception system throw more variated results, the contour system interprets the bed as a couch, while the segmentation locates and classifies properly the night desk. Nevertheless, Inception results are not accurate with reality.

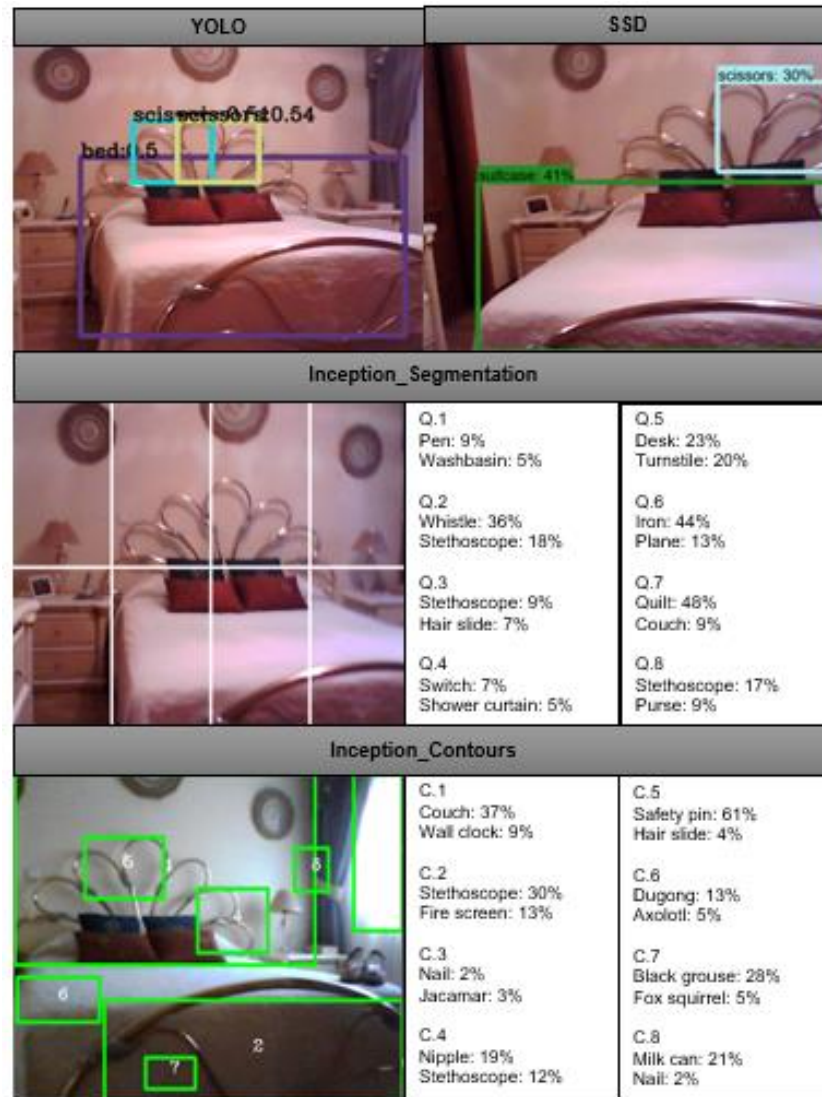


Figure 50. Bathroom test

In this case, setting a threshold value of 35% the false positive of the scissors can be eliminated from the SSD output and just have the bed classification, while in YOLO this can not be done because the wrong prediction has higher confident than the correct one. Nevertheless, the bed is classified with a 50% of probability in YOLO and with a 41% in SSD. So, in other circumstances with a more classical bed image, it can be understood that YOLO would detect and classify it easier.

7 Socio-Economic Impact and Budget

This chapter presents an analysis of the work from a socio-economic point of view. Furthermore, an estimated budget of the project is given.

7.1 Socio-Economic Impact

The main area of use of robots in the past year has been the industrial sector. This paradigm is changing due to new needs in the services sector and certain industrial activities which has led to exploit this technology to improve their processes. Healthcare, domestic and military sectors are the ones who lead this change, a change that provoked the appearance of the robotic mobile platforms.

In the research area, the use of mobile robots for remote environment explorations is very useful and exploited. In the domestic area, is increasing the use of mobile robots to help people with reduce mobility and not only to do tasks such as cleaning.

Robots are starting to be introduced in many different environments which they should be able to understand and recognize the objects that surround them in order to facilitate better information or take better decisions. Therefore, the importance and

usefulness of mobile robots with effective object recognition systems to be introduced in the society explains the great research effort in this field over the last years.

7.2 Budget

Here a brief study of the project cost is presented. To determine the hours of work is necessary to explain the different phases that led to the development of this thesis work, which are exposed in Table 11. First, the familiarization with the working environments and tools was performed. Almost at the same time the literature search was started by gathering information about the topic and the state-of-the-art, which led to select the different tools and thus its learning process. Then, the project continued with the research to find suitable approaches for the proposed tasks and, subsequently, the system implementation. As the last step, this report was written.

Table 11. Phases of the project

Phase 1	Familiarization with the working environments and tools	180 hours
Phase 2	Literature search	150 hours
Phase 3	Research	180 hours
Phase 4	Implementation of algorithm	200 hours
Phase 5	Creation of the document	140 hours

Some parts of these phases were developed simultaneous, taking a total of 850 hours to finish it. The duration for each phase can be found represented in the Gantt diagram (see Figure 51).

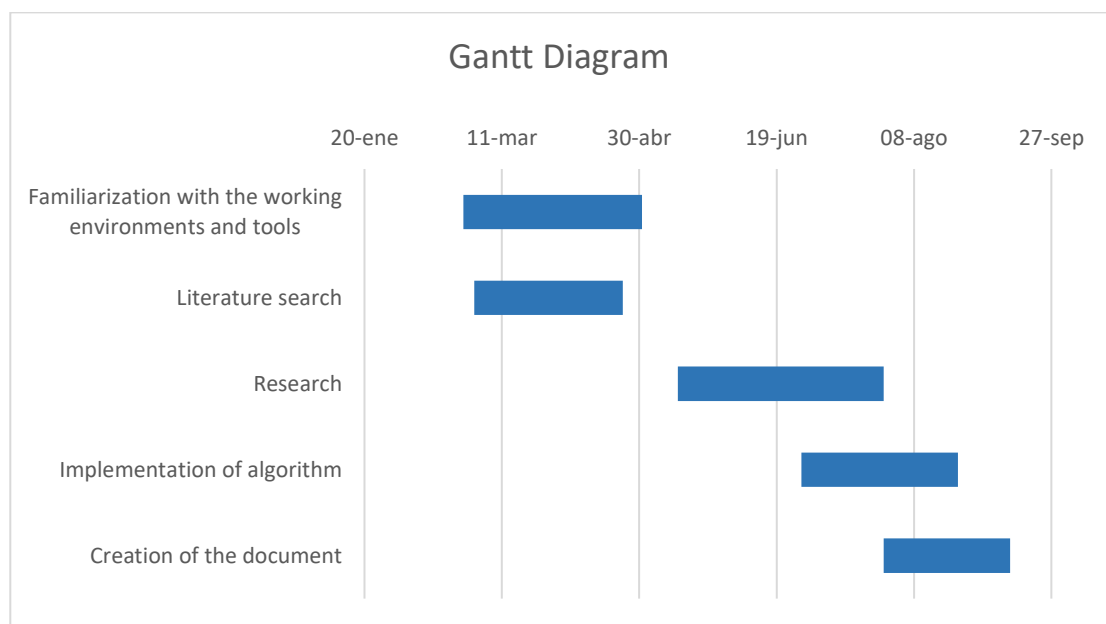


Figure 51. Gantt Diagram

The material that is part of the robotic system constitute themselves mainly most of the material costs as it is shown in Table 12. No software costs are taken into account because of the open-source nature of the programs and systems used. A 10% of depreciation of the material is considered for the total costs' calculations.

Table 12. Material Costs

	Cost (€)	Amortization (€)
<i>Turtlebot</i>	800	80
<i>ASUS PRO LIVE camera</i>	200	20
<i>Laptop</i>	800	80
<i>Router</i>	50	5
<i>Office material</i>	5	5
<i>TOTAL</i>	1855	190

The total budget is deducted by adding the cost of working hours (15€/h), the material costs and indirect costs, which represents the 15% of the total cost of the project. Thus, the total amount needed to carry out this work is approximately 14881 Euros (see Table 13).

Table 13. Total Budget

	Cost (€)
<i>Material cost</i>	190
<i>Working hours cost</i>	12.750
<i>Indirect cost</i>	1.941
<i>TOTAL</i>	14.881

8 Conclusions and Future Works

Finally, the conclusions extracted from the experimental data are summarized in this chapter, and possible future works are presented.

8.1 Conclusions

The main objectives of this thesis were to develop a convolutional neural network, develop systems capable of doing multi-object recognition and prove their functionality in real-time embedded in a mobile robot platform.

The conclusions that can be obtained out of building, training and testing the CNN exposed in this work are the following. Analyzing the training results, it can be observed how a more complex desired task needs a complex neural network configuration. This fact is present in the election of the best architectures for each network, when the CNN was trained on 2 classes, the best configuration was the 2 convolutional layers network with an average precision of 88.5% and an average sensitivity of 87.5%, while the 3 classes network present better results with 3 layers with an average precision of 88.3% and an average sensitivity of 88.3%.

Both results during the test present lower accuracy results than what the training showed, with a 100% training accuracy and 83% validation accuracy for both networks, the Keys-Wallet-Undetm. CNN and the Keys-Wallet CNN. Furthermore, the learning rate varies also the output, but its effect is not notable when using a

large enough number of iterations or epoch during the training.

When analyzing these two CNNs on real images, not coming from the training and verification tests, and implemented over the mobile robot platform using ROS, the results obtained clearly shows an advantage of using a third neutral class to recognize backgrounds/undetermined objects, despite the accuracy decreases with respect to the offline tests in every object classification.

This class helps the system to identify where there are no objects of the targeted classes and thus eliminate many false positive or decrease the confident of the predictions making possible to set a threshold value to filtrate them. Nevertheless, when it comes to real-time multi-object recognition, the results are not as good as one could expect. One solution for this is incrementing the number of images for training purposes, on this work a small number of them were used, and the results reflect that. The larger the number and variety of images of each class, the better the results that can be obtained.

From the second part of the thesis the following conclusions can be extracted. The experimental results have shown that the implementation of real-time multi-object recognition neural networks in a mobile robot with ROS was successfully done. The precision of these systems differs from one to another, but the general conclusion obtained from all of them is that their performance decreases when dealing with real-time information. Besides, for an indoor environment the systems still have a big gap to fulfill in terms of classes able to classify.

Due to the nature of the InceptionV3 network, despite its ability to classify a large number of classes, is not a suitable network to implement for multi-object detection in real time. Both methods developed to detect objects on these images fail to throw good results. The first method, the image segmentation technique, does not give good results when more than one object is present in a quadrant. The second method, the contour detector, has the problem of not detecting properly the main objects. The main reasons for this problem are the light and brightness condition, which affects directly to the contour detection process leading to misdetections.

The other two systems have better results despite they are trained to classify less objects. Nevertheless, the objects detected in the different indoor scenes are key objects to know the possible scenario where the mobile robot is. Out of both methods, SDD and YOLO, this second one presents a better performance, detecting in most of the cases more objects and with higher confident.

For these reasons, the conclusion extracted out of the experimental results is that the best object recognition system based on deep learning for real-time multi-object recognition for indoor environments is YOLO. Its performance was the best in 5/5 scenarios tested, with an average confident of recognition of 60%. Despite it can classify less different objects, the classification of the objects is more accurate and with less false positives than Inception systems, which in many cases give unconnected or nonsense objects classifications.

8.2 Future Works

Future research branches for future works would focus:

- YOLO network training on more specific classes for indoor environments.
- Deeper analysis of CNN hyper-parameters influence during the training.
- Scene classification from the output given from the object recognition system.
- Vision and navigation interaction.

To obtain better object classification, the network YOLO could be trained to detect the desired classes and not just the object presents in COCO dataset. For it, creating a dataset would be needed as well as a computer with GPU to analyze and compute all the operations in an acceptable time.

Regarding the second point, a deeper understanding of the influence of hyper-parameters can help to build a better CNN or to improve the performance of other network configurations by adapting the training to the necessities.

The last two points are related because they are both based on sharing the information obtained from the developed module explained in this work. With the results of the image recognition system, the navigation system could use that information to automatically move towards a desired target. Depth cameras along with the object location provided by the system can enable this process. Besides, from the objects detected, another system could use the output to evaluate them and their confident and thus stablish a possible scenario where the mobile robot platform could be located. Furthermore, the understanding to the environment can enable higher level tasks like automatic decisions taking, or it can trigger certain functionalities of the robot depending on the scenario where it is placed.

Bibliography

- [1] "About ImageNet" (Online). Available: <http://image-net.org/about-overview>. [Accessed: 05-09-18]
- [2] "About Kobuki," KOBUKI, 01-August-2018
- [3] "AdaBoost" [Online] Available: <https://en.wikipedia.org/wiki/AdaBoost> [Accessed: 07-09-2018]
- [4] "Anatomía y Fisiología Ocular: Mecanismo de la Visión" Online. Available: http://www.ite.educacion.es/formacion/materiales/129/cd/unidad_1/mo1_mecanismo_de_la_vision.htm [Accessed: 04-09-2018]
- [5] "COCO, Common Objects in Context" Available: <https://places-coco2017.github.io/>. [Accessed: 20-08-2018]
- [6] "Convolutional neural Network" Online. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network#Design [Accessed: 22-08-2018]
- [7] "Neural Networks" (Online). Available: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf [Accessed: 22-08-2018]
- [8] "New Horizon 2020 robotics projects from 2016 - Horizon 2020 - European Commission," Horizon 2020. [Online]. Available:
- [9] "Object Detection: Speed and Accuracy Comparison".[Online] Available: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359. [Accessed: 20-08-2018]
- [10] "OpenCV". Available: <https://opencv.org/>. [Accessed: 02-August-2018].
- [11] "Python". Available: <https://www.python.org/about/> [Accessed: 02-August-2018].
- [12] "Robot Vision vs Computer Vision: What's the Difference?" Alex Owen Hill. Available: <https://blog.robotiq.com/robot-vision-vs-computer-vision-whats-the-difference> [Accessed: 07-08-2018]
- [13] "Tensorflow Tutorial 2: image classifier using a CNN" (online). Available: <https://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/> [Accessed: 23-08-2018]
- [14] "TensorFlow". Available: <https://www.tensorflow.org/>.

[Accessed: 02-August-2018].

[15] "Tensorflow: image recognition" (Online) Available: https://www.tensorflow.org/tutorials/images/image_recognition [Accessed: 24-08-2018]

[16] "The Machine Learning Algorithms Used in Self-Driving Cars" Savaram Ravindra. (online) Available: <https://www.kdnuggets.com/2017/06/machine-learning-algorithms-used-self-driving-cars.html> [Accessed: 06-09-2018].

[17] "TurtleBot" [Online]. Available: <http://www.turtlebot.com/>. [Accessed: 01-August-2018].

[18] "What is Horizon 2020? – Horizon 2020 – European Commission," Horizon 2020.[Online]. Available: <https://ec.europa.eu/programs/horizon2020/en/what-horizon-2020>. [Accessed: 03-August-2018]

[19] "What is Horizon 2020? – Horizon 2020 – European Commission," Horizon 2020.[Online]. Available: <https://ec.europa.eu/programmes/horizon2020/en/h2020-section/information-and-communication-technologies> [Accessed: 03-August-2018]

[20] "Xtion PRO LIVE | Sensores de movimiento," ASUS España. [Online]. Available: https://www.asus.com/es/3D-Sensor/Xtion_PRO_LIVE/. [Accessed: 01-August-2018].

[21] "YOLO". Available: <https://pjreddie.com/darknet/yolo/>. [Accessed: 20-08-2018]

[22] " Hope, Tom; Resheff, Yehezkel S.; Leider, "Learning TensorFlow: A Guide to Build Deep Learning Systems Itay. 2017

[23] " Li, L.J., Socher, R., and Fei-Fei, L. "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework (2009) IEEE Conference on Computer Vision and Pattern Recognition.

[24] A. C. Hernández, C. Gomez, J. Crespo, and R. Barber, "Object Detection Applied to Indoor Environments for Mobile Robot Navigation," Sensors, vol. 16, p. 1180, Jul. 2016.

[25] A. Quattoni and A.Torralba, "Recognizing indoor scenes," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[26] Addison-Wesly, "Simulation Neuronaler Netzwe". 1994. German.

[27] Aleix M. Martinez and Avinash C. Kak. Pca versus Ida. In IEEE "Transactions on pattern analysis and machine intelligence" 2001.

[28] Alessandro Iván Fava Fernández, "Big Data y Deep Learning para la detección de objetos en imágenes" Trabajo de Fin de Grado. Universitat Politecnica de Catalunya.

- [29] Álvaro Fuentes, Sook Yoon, Sang Cheol Kim, Dong Sun Park, "A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition" *Sensors* 2017, 17(9), 2022; doi:10.3390/s17092022.
- [30] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" 04/2017. eprint arXiv:1704.04861
- [31] Apostolos P. Psyllos, Christos-Nikolaos E. Anagnostopoulos, and Eleftherios Kayafas, "Vehicle Logo Recognition Using a SIFT-Based Enhanced Matching Scheme" *IEEE Transactions on Intelligent Transportation Systems* (Volume: 11, Issue: 2, June 2010)
- [32] Ashutosh Singla, Lin Yuan, Touradj Ebrahimi. "Food/Non-food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model" *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, Pages 3-11. Amsterdam, The Netherlands — October 16 - 16, 2016
- [33] C. Szegedy, S.Reed, D. Erhan "Scalable, High-Quality Object Detection" 3 Dec 2014, v1, arXiv:1412.1441
- [34] Christian Szegedy, Alexander Toshev, and Dimitru Erhan. "Deep Neural Networks for object detection". In "Advances in Neural Information Processing Systems" 2013
- [35] Christian Szegedy, Wei Liu, Yangqing Jia, "Going Deeper with Convolutions" Google Inc. University of North Carolina. 2015.
- [36] Christian Szegedy, Wei Liu, Dragomir Anguelov, "SSD: Single Shot MultiBox Detector" *Computer Vision – ECCV 2016*. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham, 8 Dec 2015
- [37] D. Kriesel, "A brief introduction to Neural Networks"
- [38] Dam Duncan, Gautam Shine, Chris English. "Facial Emotion Recognition in Real Time" 2018, 2nd International Conference on Inventive Systems and Control (ICISC)
- [39] David G. Lowe, "Distinctive image features form scale-invariant key-points" University of British Columbia, 2004.
- [40] Drew Schmitt, Nicholas McCoy, "Object Classification and Localization Using SURF Descriptors" December 13, 2011.
- [41] European Robotics SPARC, "Robotics 2020 Multi-Annual Roadmap (MAR). For Robotics in Europe. Horizon 2020 Call ICT-2017 (ICT-25, ICT-27 & ICT-28)," 2015. [Online]. Available: https://www.eu-robotics.net/cms/upload/topic_groups/H2020_Robotics_Multi-Annual_Roadmap_ICT-2017B.pdf. [Accessed: 04-August-2018].
- [42] European Robotics. SPARC, "Strategic Research Agenda. For Robotics in Europe 2014-2020," 2014. [Online]. Available: https://www.eu-robotics.net/cms/upload/topic_groups/H2020_Robotics_Multi-Annual_Roadmap_ICT-2017B.pdf

robotics.net/sparc/upload/topic_groups/SRA2020_SPARC.pdf. [Accessed: 09-Apr- 2018].

[43] Florent Perronnin, Jorge Sánchez, Thomas Mensink, “Improving the Fisher Kernel for Large-Scale Image Classification” ECCV 2010: Computer Vision – ECCV 2010 pp 143-156

[44] Gee-Sern Hsu, Truong Tan Loc, and Sheng-Lung Chung. A comparison study on appearance-base object recognition. In “International Conference on Pattern Recognition” 2012.

[45] Girshick, Ross; Donahue, Jeff; Darrell, Trevor; Malik, Jitendra, “Rich feature hierarchies for accurate object detection and semantic segmentation” CVPR '14 Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Pages 580-587, 11 Nov 2013

[46] Grace Tsai, Changhai Xu, Jingen Liu, “Real Time Indoor Scene understanding using Bayesian Filtering with Motion Cues” 2011 International Conference on Computer Vision

[47] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. “Speeded-up robust features (surf)” 2008.

[48] Hervé Goëau, Pierre Bonnet, Alexis Joly. “Plant Identification in an Open-world” CLEF 2016 - Conference and Labs of the Evaluation forum, Sep 2016, Évora, Portugal. Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, pp.428--439, 2016

[49] <https://www.tensorflow.org/tutorials/layers>. [Accessed: 15-08-2018]

[50] Hung Nguyen; Sarah J. Maclagan; Tu Dinh Nguyen; Thin Nguyen; Paul Flemons. “Animal Recognition and Identification with Deep Convolutional Neural Networks for Automated Wildlife Monitoring” IEEE International Conference on Data Science and Advanced Analytics (DSAA). 19-21 Oct. 2017

[51] Ian Goodfellow, Yoshua Bengio, Aaron Courville. “Deep Learning” 18 Nov 2016.

[52] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel. “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition” Neural Networks, Volume 32, August 2012, Pages 323-332.

[53] James Carrol, Askarshan Dhakal. “Distracted Driving Recognition: Classifying Safe and Unsafe Driving with Deep Learning” 2017, Department of Computer Science. Stanford University.

[54] Jerome Paul N. Cruz, Ma. Lourdes Dimaala, Laurene Gaile L. Francisco, Erica Joanna S. Franco, “Object Recognition and Detection by Shape and Color Patten Recognition Utilizing Artificial Neural Networks” Department of Computer Science, Polytechnic University of the Philippines.

[55] John Wiley & Sons, “Introduction to MPEG-7: Multimedia Content Description Interface” 2002.

- [56] Jörg Conradt. "State of the Art of Object Recognition Techniques" 2016.
- [57] José Carlos Rangel Ortiz, "Scene Understanding for Mobile Robots exploiting Deep Learning Techniques" Doctoral Thesis, Universitat d'Alacant, 2017.
- [58] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, Chu-Song Chen, "Deep Learning of Binary Hash Codes for Fast Image Retrieval" Academia Sinica, Taiwan & Yahoo! Taiwan
- [59] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks" (PDF). Advances in Neural Information Processing Systems. 1: 1097–1105. "Very Deep Convolutional Networks for Large-Scale Image Recognition". Karem Simonyan & Andrew Zisserman. University of Oxford. 2015
- [60] Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises".
- [61] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, Gwo-Jong Yu "A new LDA-based face recognition system which can solve the small sample size problem" 2000 in Pattern Recognition [IF: 4.58].
- [62] Lucas Pinheiro Cinelli, "Anomaly Detection in Surveillance Videos using Deep Residual Networks" Universidade Federal do Rio de Janeiro. February 2017.
- [63] M.S. Barlett, G. Littlewort, C.Lainscsek "Machine learning methods for fully automatic recognition of facial expressions and facial actions" IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583). 10-13 Oct. 2004.
- [64] Marco Alexander Treiber, "An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications".
- [65] Marios Anthimopoulos, Stergios Christodoulidis. "Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network" IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 35, NO. 5, MAY 2016
- [66] Matija Radovic, Offei Adarkwa and Qiaosong Wang, "Object Recognition in Aerial Images Using Convolutional Neural Networks" Journal of Imaging 2017, 3(2), 21 14 June 2017
- [67] Michal Busta, Lukáš Neumann and Jiří Matas, "Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework" Centre for Machine Perception, Department of Cybernetics Czech Technical University, Prague, Czech Republic.
- [68] Ming Li, Baozong Yuan, "2D-LDA: A statistical linear discriminant analysis for image matrix". Pattern Recognition Letters, Volume 26 Issue 5, April, 2005, Pages 527-532
- [69] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, Wenyu

Liu, "TextBoxes: A Fast Text Detector with a Single Deep Neural Network" School of Electronic Information and Communications, Huazhong University of Science and Technology. 21 Nov 2016.

[70] Morgan Quigley, Biran Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler y Andrew Ng. ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software, 2009

[71] Nasser M. Nasrabadi "Pattern Recognition and Machine Learning" Journal of Electronic Imaging 16(4), 049901 (1 October 2007)

[72] Osvaldo Simeone (2017) "A brief introduction to Machine Learning for Engineers", Vol.3 , pp 1-201, 8 Sep 2017.

[73] P. Espinance, T.Kollar, A.Soto, N.Roy, "Indoor Scene Recognition Through Object Detection Using Adaptative Objects Search" Department of Computer Science and computer Science and Artificial Intelligence Lab.

[74] P.Viola, M.Jones. "Rapid object detection using a boosted cascade of simple features" Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. 8-14 Dec. 2001

[75] Pablo Arbaláez & Charles Fowlkes, "Contour Detection and Hierarchical Image Segmentation" IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 33, Issue: 5, May 2011)

[76] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, "Fast Single Shot Detection and Pose Estimation" Fourth International Conference on 3D Vision (3DV). 25-28 Oct. 2016

[77] Peiyun Hu, Deva Ramanan, "Finding Tiny Faces" Robotics Institute, Carnegie Mellon University.

[78] R.Begg, J. Kamruzzaman. "A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data" Journal of Biometrics, Volume 38, Issue 3, March 2005, Pages 401-408.

[79] R.C. Veltkamp, M. Tanase, "Content-Based Image Retrieval Systems: A survey" Technical Report UU-CS-2000-34, 2002

[80] Redmon, Joseph; Farhadi, Ali, "YOLO9000: Better, Faster, Stronger" 25 Dec 2016

[81] Redondo-Cabrera, C., López-Sastre, R.J. "Recognizing in the depth: Selective 3D spatial pyramid matching kernel for object and scene categorization" Image and Vision Computing 32(12) · September 2014

[82] Rosten E., Drummond T. (2006) Machine Learning for High-Speed Corner Detection. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

- [83] Sang-Geol Lee; Yunsick Sung; Yeon-Gyu Kim; Eui-Young Cha. "Variations of AlexNet and GoogLeNet to Improve Korean Character Recognition Performance" Journal of Information Processing Systems. Feb2018, Vol. 14 Issue 1, p205-217. 13p
- [84] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, Jitendra Malik, "Indoor Scene Understanding with RGB-D images", IJCV manuscript No 3, 1 Nov 2015
- [85] Sayed-Mahdi Khaligh-Razavi. "What you need to know about the state-of-the-art computational models of object-vision: A tour through the models" Univeristy of Cambridge, 2014.
- [86] Simon Achatz. "State of the Art of Object Recognition Techniques", Technische Universität München, 2016.
- [87] Simon Haykin, "Neural Networks and Learning Machines"
- [88] Simon Thorpe, Denis Fize & Catherine Marlot, "Speed of processing in the human visual system" Centre de Recherche Cerveau & Cognition, UMR 5549, 31062 Toulouse, France Nature volume 381, pages 520–522 (06 June 1996).
- [89] Steinberg, D.M., Pizarro, O., and Williams, S.B. "Hierarchical Bayesian models for unsupervised scene understanding" Computer Vision and Image Understanding. Volume 131 Issue C, February 2015, Pages 128-144
- [90] Sushma Nagaraj, Bhushan Muthiyan, Swetha Ravi, "Edge-based street object detection" IEEE SmartWorld, Ubiquitous Intelligence & Computing. 4-8 Aug. 2017
- [91] Sushree Dhyanimudra Nayak, Siddharth Swarup Rautaray, "A review on object recognition techniques for practical application" International Journal of Science, Technology & Management. Vol. 4. Issue.1 January 2015.
- [92] Tung, F. and Little, J.J "Improving scene attribute recognition using web-scale object detectors" Computer Vision and Image Understanding, Volume 138 Issue C, September 2015, Pages 86-91
- [93] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, Cordelia Schmid, "Action Tubelet Detector for Spatio-Temporal Action Localization" Computer Vision and Pattern Recognition, 4 May 2017 (v1)
- [94] Wenyi Zhao, Arvinth Krishnaswamy, Rama Chellappa, Daniel L. Swets, John Weng, "Discriminant Analysis of Principal Components for Face Recognition" Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, April 1998