Co-funded by the
Erasmus+ Programme
of the European Union

ERASMUS MUNDUS MASTER IN
IMAGE PROCESSING AND COMPUTER VISION

PÁZMÁNY PÉTER
CATHOLIC UNIVERSITY

université de BORDEAUX

UAM
Universidad Autónoma de Madrid

MSc THESIS

People detection in omnidirectional cameras: development of a deep learning architecture based on a spatial grid of classifiers

PRESENTED AT

UNIVERSIDAD AUTONOMA DE MADRID
ESCUELA POLITECNICA SUPERIOR

Escuela Politécnica Superior

Departamento de Ingeniería Electrónica y de las Comunicaciones

Author: Sepúlveda Jorcano, Enrique

Academic Supervisor: Carballeira López, Pablo

DATE: June, 2020

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politéctnica Superior
Universidad Autónoma de Madrid

Junio 2020

Trabajo parcialmente financiado por el Ministerio de Ciencia, Innovación y Universidades del Gobierno de España bajo el proyecto TEC2017-88169-R (MobiNetVideo)

Image Processing AND Computer Vision
PÁZMÁNY PÉTER CATHOLIC UNIVERSITY
UAM Universidad Autónoma de Madrid
université de BORDEAUX
Co-funded by the Erasmus+ Programme of the European Union

Declaration of Authenticity

(to be inserted as second page of the thesis)

I, the undersigned Enrique Sepúlveda Jorcano, student of the Image Processing and Computer Vision program, hereby certify that this thesis has been written without any unauthorized help, solely by me, using only the referenced sources. Every part quoted in whole or in part is indicated clearly with a reference.

I declare that I have not submitted this thesis in any other higher education institution, excluding the partner universities of the IPCV Consortium.

.....

Student

People detection in omnidirectional cameras: development of a deep learning architecture based on a spatial grid of classifiers

Abstract

Deep learning has arisen as one of the best tools to use in computer vision. With the potential of adapting to almost any problem, deep learning CNN (Convolutional Neural Networks) are highly effective in multiple tasks for computer vision. The application of deep learning in traditional cameras has improved the performance in detection tasks with respect to the state-of-the-art methods that don't use deep learning. Images from traditional cameras are perspective projections of the real world with almost no distortion. This allows us to train networks that can learn the general appearance of an object and assume that they are spatial-invariant between frames.

However, the use of omnidirectional cameras has increased in the last years thanks to the advantage they offer with respect of traditional cameras: a wider Field of View. While in conventional cameras it usually doesn't go further than 60º, in omnidirectional cameras it can reach values of 160º. With a single omnidirectional camera, therefore, we are able to cover a wider area than with a traditional camera, reducing costs of deployment. But this comes at a cost, and it's that they introduce a great distortion, making the objects change their appearance depending on their position in the image. This makes some of the existing deep learning techniques to highly reduce their performance, and it's necessary the use of new techniques.

The proposed method in this thesis uses CNNs to extract characteristics from omnidirectional images to detect objects using a spatial-aware grid of classifiers. The original idea, proposed by [1] was to use a HOG features and grid of SVM classifiers SVM where each of the classifier will learn from a region of the image. In this way we avoid the problem of using one single classifier that has learnt general spatial-invariant characteristics. In [1] the goal was to create a robust and real-time detector using (1) HOG to extract a feature vector from the image and (2) a grid of SVM (Support Vector Machine) classifiers that would predict the position of the objects. In our work we propose to substitute this two step architecture with a CNN that could be used end-to-end.

We have tested Alexnet, Resnet18 and Resnet50. Part of this project has been the study of data augmentation techniques to improve the performance of the system. An example of this is the creation of synthetic images where people are added from random frames to others. The best performance has been obtained with Resnet50, with values of F1-Score around XX%.

Keywords: Omnidirectional cameras, deep learning, people detection, convolutional neural networks, CNN, object detection, support vector machines, SVM, transfer learning, computer vision, grid of classifiers.

Acknowledgements

I would like to thank my tutor, Pablo, for helping me during this project and to Jesús for his help and patience during the master.

Thanks to my old friends for being there all this time and listening my complaints, and to my new IPCV friends, with whom I spent this last two years living a life-changing experience and a lot of adventures.

To my family, for being there all the time and specially my parents and brother, for the education they gave me and the support I receive from them everyday, supporting me in good and bad moments.

Index

List of Figures

Table index

Chapter 1

Introduction

Content

1.1 Motivation

Object detection has always been a challenge in the field of computer vision, given the difficulties it encloses. Object detection algorithms aim to identify the objects in a scene automatically, without the need of human interaction. These objects could be anything, from simple objects to animals or people, and this is specially important in computer vision as it has applications in fields such as video security, self-driving vehicles [10], sports[11] or robotics. Traditionally, the used cameras to capture the scene were the ones who introduced less distortion in the images, with the counterpart of having a reduced FoV (Field of View). This traditional cameras,

as we will call them for the rest of the work, are easier to build and problems such as illumination changes and occlusions are well-known, so they were preferred over other cameras for many years. However, there are other types of cameras that offer a wider FoV, reaching to 180º against the 60º of traditional cameras, but with the disadvantage of introducing several distortions in the shape of the objects and being more complex to build. This wider FoV offers the possibility of reducing the number of cameras needed to cover a big area, something specially important in surveillance tasks. The recent interest for this cameras is explained thanks to the development of faster and computationally efficient algorithms based on deep learning that are able to cope with the commented introduced distortion in the images and the inherent problems of traditional cameras. While in traditional cameras the are hundreds of algorithms based on deep learning for object and people detection, for other types of cameras the already existing algorithms are not robust enough and their efficiency when applied to these distorted images is lower. There is still a big space for research and improvements, and to solve the commented distortion introduced in the image a different approach to the problem and the use of new techniques are required.

In [1] the proposed idea is to extract a vector of characteristics from the image and train a grid of foveatic classifiers placed over the image. In this work, it is explained that the distortion suffered by an object (or in this case an person) in these images depends on its position in the image. The idea is to train a grid of classifiers in which each of them will learn to detect people in a region (or fovea) of the image. In this way, the classifier will learn to identify the characteristics of a distorted person in that specific region regardless of the distortion in other regions of the image. However, this work was limited to one point of view, i.e. that each camera needed to be trained independently.

1.2 Contribution

In this work, we've implemented a modification of the method proposed in [1] and which was continued in [7]. The goal is to robustly detect people in omnidirectional cameras making use of deep learning techniques. We will make use of CNNs (Convolutional Neural Networks) to detect people in a similar way: we will teach the network to activate a grid of classifiers in the regions were people appear, locating each person in the scene.

1.3 Structure

This document is organized in the following way: in Chapter 2 the internship report contains a description of the work done at the university. In Chapter 3 a review of the state of the art is done, explaining the previous methods and the first approach to this new method. A description of the existing types of cameras is also included, with its advantages and disadvantages. In Chapter 4, the proposed system is explained, with a description of each part of the process and the found problems and solutions proposed. Chapter 5 comments the obtained results and describes the tests and experiments made. Chapter 5.4.6 concludes the work, pointing out the flaws and advantages of the method and the possible improvements for the future.

Chapter 2

Internship report

Content

2.1 Introduction

In this Chapter we are going to explain the progress of the intership done in the Autonomous University of Madrid, and in particular in the VPULab (Video Processing and Understanding Lab). We will start talking about the institution, the general management and then we will explain more about the VPUlab group, the methodology used and the projects they have.

2.1.1 Autonomous University of Madrid

The UAM (Universidad Autónoma de Madrid, in Spanish) is one of the six public universities in Madrid, in the top three universities of Spain, in the range 201-300 in the world according to [12] and the 210th according to [13]. Founded in 1968, this university offers a huge range of studies. This university has two campus: the Faculty of Medicine, near La Paz Hospital, and the Cantoblanco Campus 15 km to the north of Madrid. It has eight faculties and several affiliated centers. The actual rector is Rafael Garesse Alarcón. This university has numerous agreements with different institutions, highlighting the Campus de Excelencia Internacional UAM+CSIC where the UAM collaborates with the CSIC (Spanish National Research Council, Consejo Superior de Investihación Científica in Spanish, Spanish National Research Council). The CSIC is the largest public institution in Spain dedicated to research and one of the most renowned of the European Research Area (ERA), top 5 by number of action in Eurpoe and the 7th public research globally [14].

2.1.2 Video Processing and Undestanding lab

The Video Processing and Understanding Lab (VPU) is a research group focused on image processing, video sequence analysis and content adaptation. It was originally created in Universidad Politécnica de Madrid, formed in 1981, and it was later extended to the EPS (Escuela Politécnica Superior in Spanish) in UAM. The main fields of this group are the video-surveillance systems and video content repositories, mainly oriented to real-time processing.

2.1.3 Project development and VPU

For this internship it was required basic knowledge in Python and/or C++, a master or equivalent study in image processing and computer vision and in machine learning. This was one of the proposed internships by the UAM in the IPCV program. The project has been done in the VPU lab as part of its group research, mostly focus on video processing as commented before. This project continues with an existing work done based in object detectors and deep learning. Most of the projects done in this lab are based on object detection and object tracking.

During the internship the project has been developed in the laboratories of VPU and from home. The laboratory has all the necessary equipment, with a workplace composed by a physical computer and the access to a virtual machine with more computational power (GPU) with the purpose of training neural networks. In the group, there where weekly meetings on Tuesdays where each of the PhD students in the group presented their work in around 10 minutes to the rest of student and the teachers in the group, commenting the challenges and solutions they found during their research. Master students were encouraged to participate showing their process to the rest of the group every two or three weeks. With these meeting the goal was not only to receive advices and help, but also to offer ideas to other people for future works. These meetings were done from home via Teams when the access to the laboratory was not possible.

Apart from this group meetings, I had a weekly meeting with my tutor to discuss the direction of the project. In these meetings I presented the current work and received some feedback, and we discussed the limitations we found in the project and possible solutions. If needed, we met more than once in a week to comment the progress and we communicate constantly through email.

Chapter 3

State of the art

Content

3.1 Introduction

Algorithms for people detection have a great importance in the field of computer vision. People detection is the first step in complex systems such as people identification or action recognition. While there are several people detection algorithms for traditional cameras, we will focus our study in the ones created for omnidirectional cameras. Both types of cameras have differences and they offer advantages and disadvantages with respect to the other, reason why using algorithms initially created for traditional cameras will not return good results in omnidirectional cameras if they are directly applied [15].

3.2 Traditional cameras

The usual way to understand how a camera works it to approximate a camera to the *pin-hole* model. This *pin-hole* approximation is a just simplification which allows us to understand how the 3D points in space are transformed in 2D points just the way the human eye works.

The light goes through a common 3D point called *projection center*. All the light rays then project on the other side in the so-called image plane, where the scene is represented in 2D. This 2D projected image is inverted, but equivalent to an upright version of it, as we can see in Figure 3.1. The projected image conserves the linearity but as every projective transformation, parallelism and angles are not preserved.

The biggest disadvantage of these traditional cameras is the FoV (Field of View).

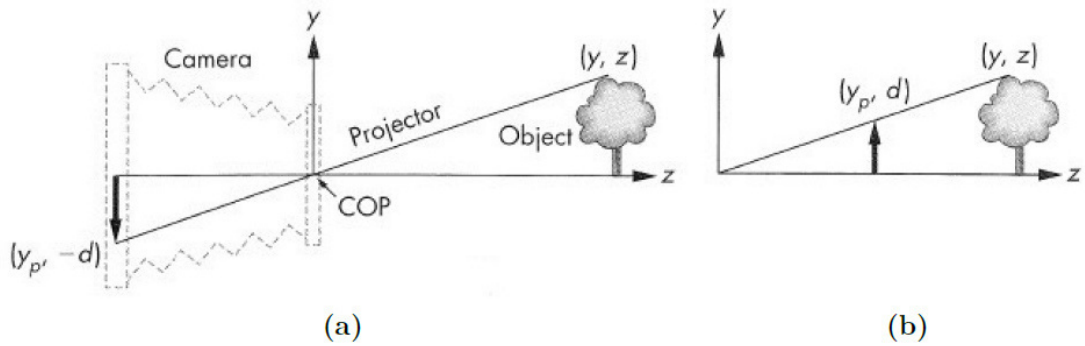
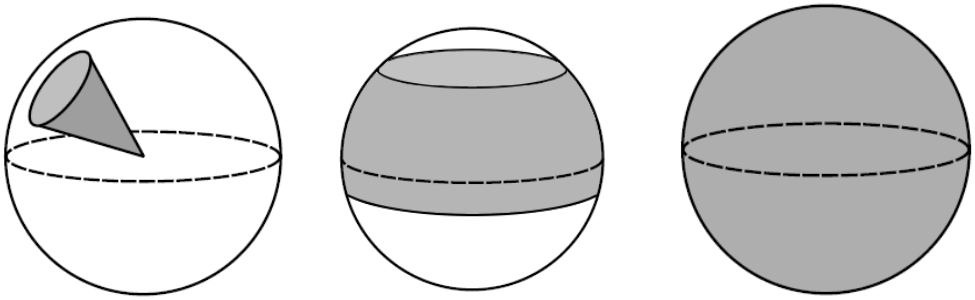


Figure 3.1: Pin-hole camera model where all the rays go through the COP (Center of Projection). (a) shows the inverted projection and (b) the equivalent upright version. Extracted from [1].

In humans, this FoV is around 150-160 degrees horizontally and 135 degrees vertically. In traditional cameras, the FoV is a around 40-60 degrees. This huge difference in the view of the scene justifies the interest for other cameras with a wider FoV where a greater view of the scene can be obtained. There are different types of cameras with a wider FoV than the traditional cameras (also known as directional cameras). If we imagine the area covered by the camera as a sphere (also called *view-sphere*), where the centre coincides with the center of projection, we can differentiate other two main groups apart from the directional cameras (Figure 3.2):

- Panoramic cameras, in which the covered area in the view-sphere is 360 degrees in one direction.
- Omnidirectional cameras, in which the FoV is the whole sphere.

Real omnidirectional cameras are in general more complex to build and therefore panoramic ones are preferred in most of the cases. Among the panoramic cameras, we can find several types with some advantages or disadvantages. Some cameras consist in a small camera system in which each camera takes charge of a region and



(a) Directional (b) Panoramic (c) Truly omnidirectional

Figure 3.2: Representation of the *view-sphere* in cameras. Extracted from [1]

whose FoV overlap with the cameras around it. Merging these views, we can create a wider view of the scene. Other cameras are made with a camera in a rotating axis that takes images while rotating, that are later merged in a panoramic image. The withdraw of this camera type is that given the delay caused by the rotation, the scene should be static to obtain a good result. Fish-eye cameras make use of lens that provide a greater FoV than directional cameras. Other type of cameras consists in adding a catadioptric lens to a traditional camera, focusing the light rays of a bigger region of the scene in the field of view of the traditional camera, with the problem of creating a blind spot in the centre of the image (where the catadioptric lens is centred). These are just some examples and there are more ways to obtain cameras with a wider FoV using lens or more complex camera systems. It's common to call fish-eye cameras and catadioptric cameras as omnidirectional cameras, and so we'll do for this work. However, as we will see next, in contraposition to the improved FoV, all these cameras introduce some distortion that we should consider.

The major problem with omnidirectional cameras is that we are trying to cover a wide 3D scene and represent it in just a 2D image. This lead to a radial distortion of the images that modifies the shape of the object according to its location in the

(a)

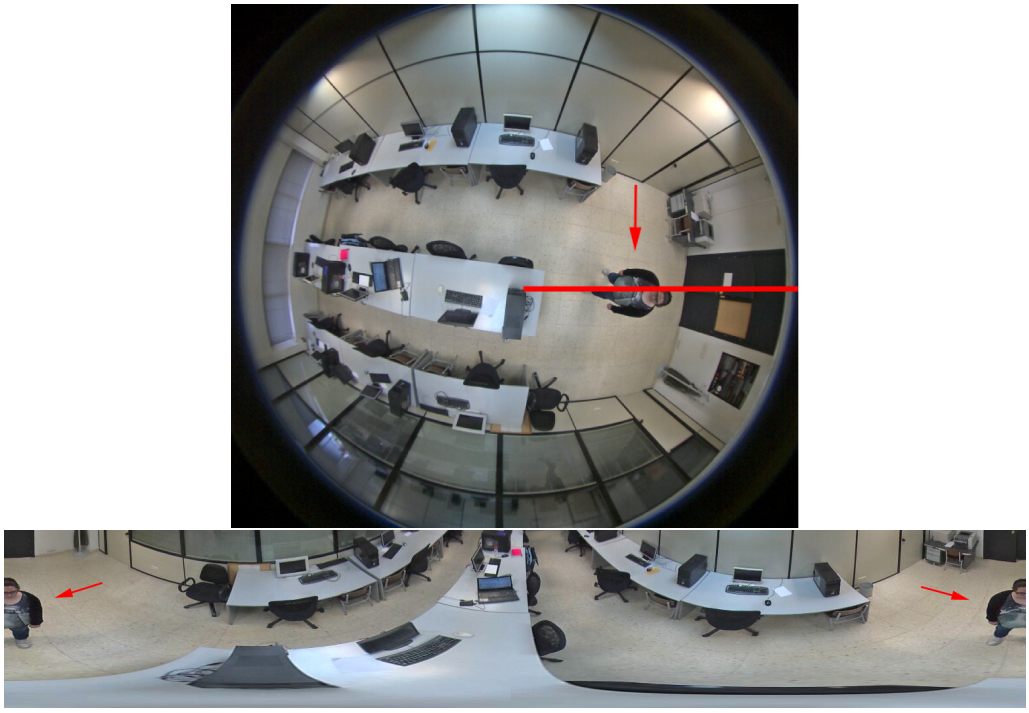
(b)

(c)

Figure 3.3: (a) covers the whole room. It's necessary the use of two directional cameras (b) and (c) for the same room. Extracted from [2].

image, contrary to the perspective image obtained from traditional cameras.

While a traditional camera isn't capable of cover a full room and it's usually necessary the use of a camera network system. Let's use the example of the security system used in an office. With a conventional camera network it's highly probable that we can't cover the entire room using one single directional camera. But an omnidirectional camera brings the possibility of reducing the number of cameras (Figure 3.3), therefore reducing costs, and the complexity of working with a camera



(a)

Figure 3.4: The limitations of unwrapping images. In (a) the red line indicates where the limits of the new perspective image. In (b) we can see that the creation of new borders cuts the person in two. Other example can be seen in the table, located in the centre in (a) but spread along the bottom in (b), suffering a high distortion. Extracted from [1].

network: calibrating several cameras and synchronizing them, checking locations to avoid blind spots... It makes a difference if instead of an office we are for example covering a mall or a large area where the number of cameras could be very high.

Therefore, there are different methods to deal with the commented distortions. One of the most used techniques consists in unwrapping the image. This method consists in transforming the omnidirectional image into a traditional perspective one. This is done by wrapping the image around a cylinder or a sphere and then sampling

it and projecting the points of the image to the image plane (Figure 3.4). This technique aims to convert the image in a way that existing perspective algorithms could be applied on it.

However, this technique doesn't return perfect results as it introduces more distortions in the image that didn't exist before. We can see an example in Figure 3.4 where the creation of a border cuts a person in half or some objects are spread along the bottom.

3.3 Object detection algorithms

We will then take a look at the most important detectors based on deep learning and the basics behind them. We will see the existing methods for people detection specifically designed for omnidirectional cameras and how this is applied in this work.

3.3.1 Pre-deep learning methods

Sliding windows algorithms were for several years the state-of-the-art method for people detection. The basic sliding window method first creates a window of fixed size that goes through the image, obtaining a vector of characteristics from each window. These descriptors are later classified into the possible objects we were detecting. This method was slow and had a high computational cost, so it was fast substituted with deep learning algorithms.

Other important algorithm widely used is the SVM (Support Vector Machine). Developed by V. Vapnik [16], it consists in finding the optimal separating hyperplane (Figure 3.5) between positive and negative samples in the feature space. For this

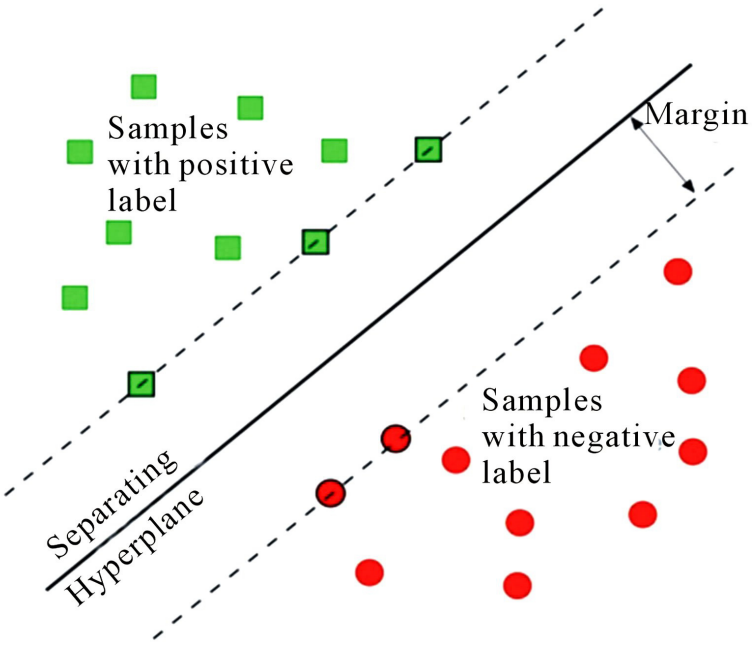


Figure 3.5: SVM hyperplane for 2D data. One side of the plane is considered positive while the other is considered negative. Samples are labeled according to the the side they lay on Extracted from [1]

the data and the labels are fed into the SVM, which with the use of a kernel are then transform into a multidimensional space.

3.3.2 Machine learning algorithms

Machine learning is nowadays one of the most powerful tools in computer vision. Based on the idea of teaching a computer to identify characteristics just like humans do, the number of possibilities it can offer is very huge and day after day more and more applications are emerging.

To define what a ANN (Artificial Neural Network) is, we should think in how the external layers of the brain works. It starts with the most basic unit, a neuron,

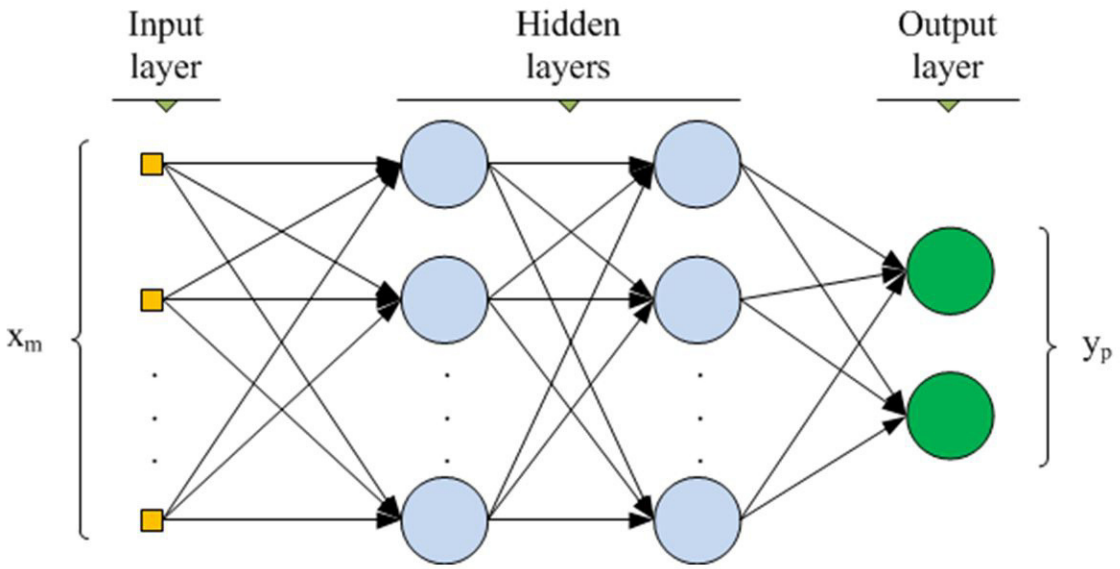


Figure 3.6: Example of neurons interconnected forming layers. In this example there are two outputs. These outputs are usually modify with an activation function, for example a *Lineal*, a *ReLu*, a *Sigmoid*... Extracted from [1]

which has an input and output, and that connects with other neurons forming layers. In machine learning, this neuron is represented with an input, a weight, a bias and an activation function to modify the output. In an ANN the structure is as follows:

1. An input layer, the first neurons to process the data fed to the network.
2. The hidden layers, that receives the processed data from one layer and outputs it to another.
3. The output layer, in which the data contains the final result of the process.

In an ANN, all the neurons are connected: the output of a neuron in a layer affects all the inputs in the next layer, and so on. We call a deep neural network a neural network with several hidden layers (Figure 3.6).

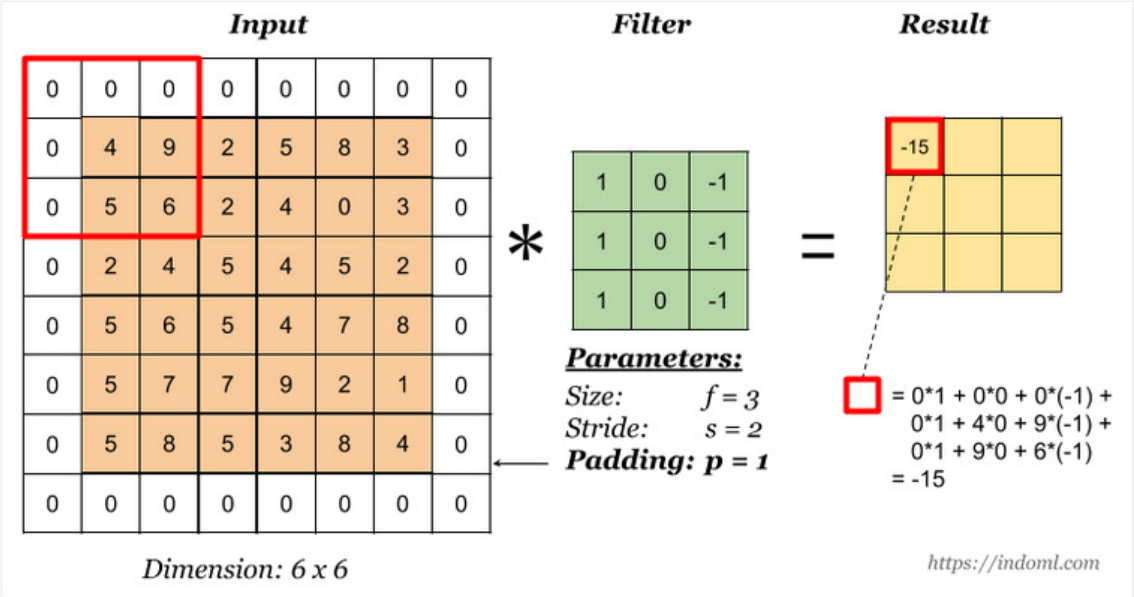


Figure 3.7: In this image we can see the effect of the parameters in the convolution. The size of the filter will affect how much we take into account neighboring pixels. The stride will affect how we move the filter. Finally, the padding is used to output a result of the same size as the input. Extracted from [3].

3.3.3 Convolutional Neural Networks

A CNN is just a deep neural network especially effective in image processing, as it's capable of deal with two-dimensional data more efficiently. It has less parameters to work with as not all the layers are fully connected, and in the case of images, it takes advantage of the relation between nearing pixels thanks to the use of 2D filters. The fact that it works especially well with two-dimensional data makes CNNs the first option in computer vision tasks such as object detection, although it's not limited to images and it's also used in other 2D data like audio.

A CNN starts with a multi-channel input image of a given size, usually with three channels (like an RGB image). This image is first convolved with a series of filters

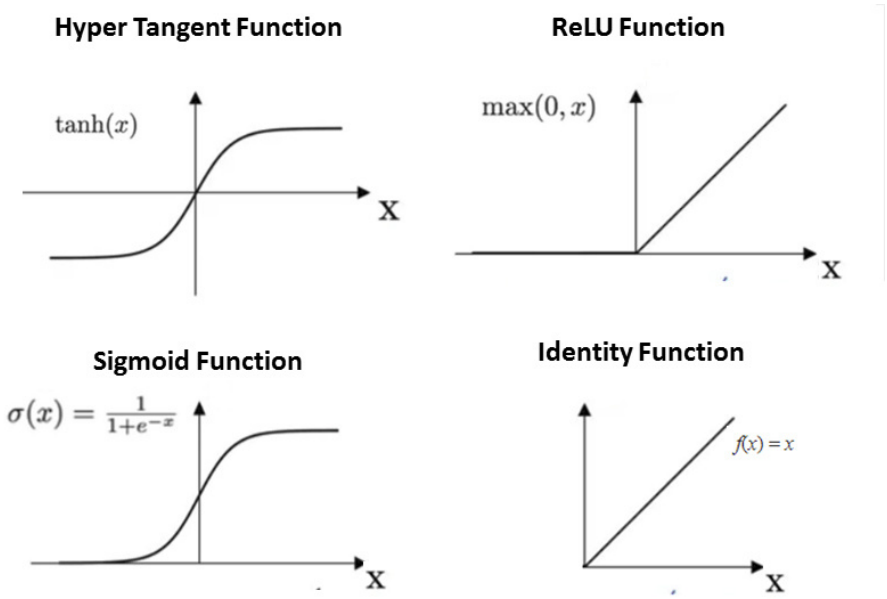


Figure 3.8: Some of the most important activations functions used in CNNs. The most used one is the *ReLu* (Rectified Linear Unit) for being the fastest [4], but there are other options like the *Sigmoid* or the *Tanh*. Note that the identity function is the same as not modifying the output. Extracted from [5].

that output what are called feature maps. A convolution (Figure 3.10 is a process that makes an element-wise multiplication between the image and the filter. This filter has a fixed size, which means that the result of the convolution will always have a fixed size too, but the values of the filter will change on each iteration. These are the weights or parameters that the CNN will learn. There are two main parameters worth to mention in a convolution (Figure 3.7: the padding and the stride. The first one makes sure that the output size is the same as the original image, while the second one tells how the filter should move in the image. The next step consists in adding a real value, called *bias*. This value helps to have some control over the final output. The last step in a convolutional layer is the activation function. In Figure 3.8 we can see some examples of these functions. These functions are non-

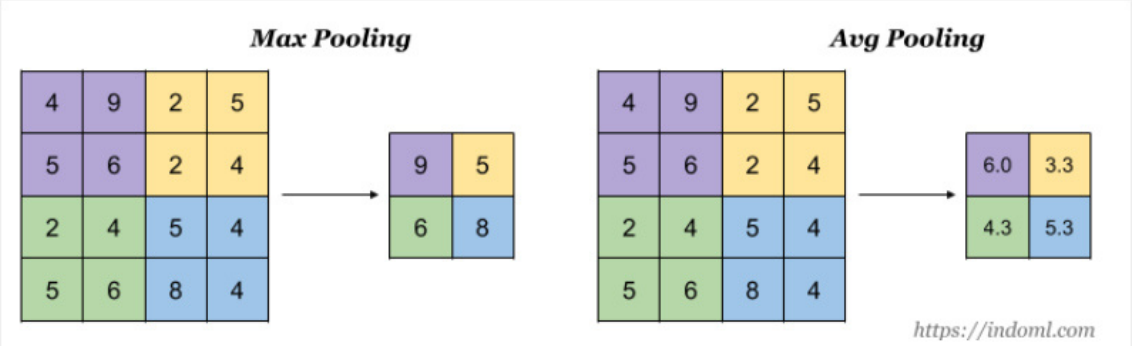


Figure 3.9: Pooling process to reduce the image size according to a kernel size of 2. On the left, *max pooling* takes the highest value in a $2x2$ region while on the right *avg pooling* makes the average among the values in that region. Extracted from [3].

lineal functions which goal is to allow the network to learn as a non-linear system. Adding to this process, it's normal to introduce some pooling (Figure 3.9 to reduce the size of the feature maps between layers. This is done to keep the features of the image that are more general. Like this, in Figure 3.10 we can see the full structure of a convolutional layer. The output consists in several features maps (as many as the number of filters), acting like the descriptor of the image. Usually, a CNN has several convolutional layers, with each of the layers having a set of independent filters, and at the end some fully connected layers. This layers are just like an ANN, layers where all the neurons are connected, and is where the classification is done using the extracted features from the convolutional part of the CNN. The output of these dense layers is usually modified with the use of some non-linear functions, like a *Softmax* or a *Sigmoid* function to normalize the output, for example between 0 and 1. We can use Figure 3.11 as example of a full CNN.

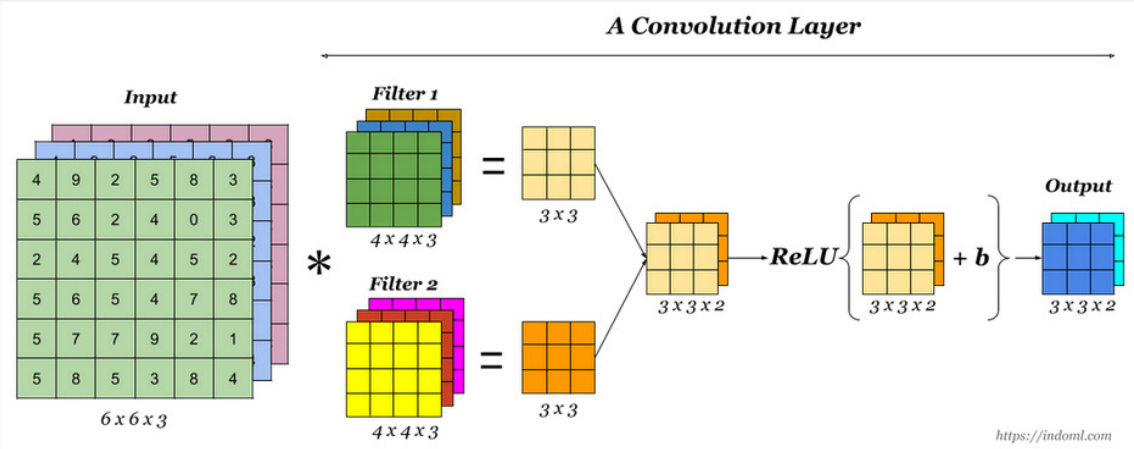


Figure 3.10: The whole process of a convolutional layer. The input image is convolved with 2 different filters. The result of these convolutions is modified with a non-linear activation function after adding the bias. The result is as many features maps as filters were used. Extracted from [3].

3.3.4 Well-known CNNs

It was in 2012 with the apparition of Alexnet that CNNs in computer vision started to gain fame for the obtained results. Since then different architectures have appeared, aiming to obtain the best results in object detection. Some of the most famous CNNs architectures for image processing are:

- Alexnet: The architecture proposed by Alex Krizhevsky (Figure 3.11) in [6] won the *Imagenet Large Scale Visual Recognition Challenge (LSVRC)*, a challenge for image classification that reached state of the art results at that time and made researchers think in the possibilities of deep learning applied to images.

- VGG-16: Proposed in [17], with a depth of 16 trainable layers, this architecture's strength resided in its simplicity (Figure 3.12: reducing the size in half

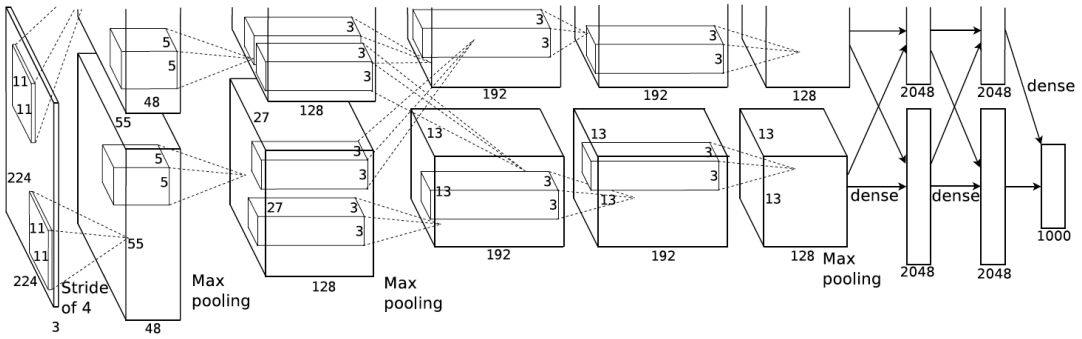


Figure 3.11: Alexnet architecture. The input image has a size of $224x224x3$. The first convolutional layer contains 48 filters. After that, a *max pooling* is applied and the result is forwarded into the next convolutional layer. Alexnet is made up of eight layers: five convolutional layers and three fully connected layers. All of these layers are followed by a *ReLu* activation function except for the last fully connected layer where a *Softmax* function is used to output a class probability between 0 and 1. Extracted from [6].

and increasing the depth in different steps in its 16 trainable layers

- Resnet: Increasing the number of layers more than a certain limit doesn't always return better results and increases heavily the computational cost. Resnet (Residual Network), proposed by He at all in [18], tries to solve this by skipping some layers and feeding the output of some convolutional layers into deeper ones.

- Inception: The idea of the Inception module is to use more than one filter size on the same input. As we saw before, in a convolutional layer a single fixed-size filter is convolved with the input. In Inception, more than one filter size is applied to the same input and then the results are concatenated and sent to the next layer. In Figure 3.13 we have an example of the first version

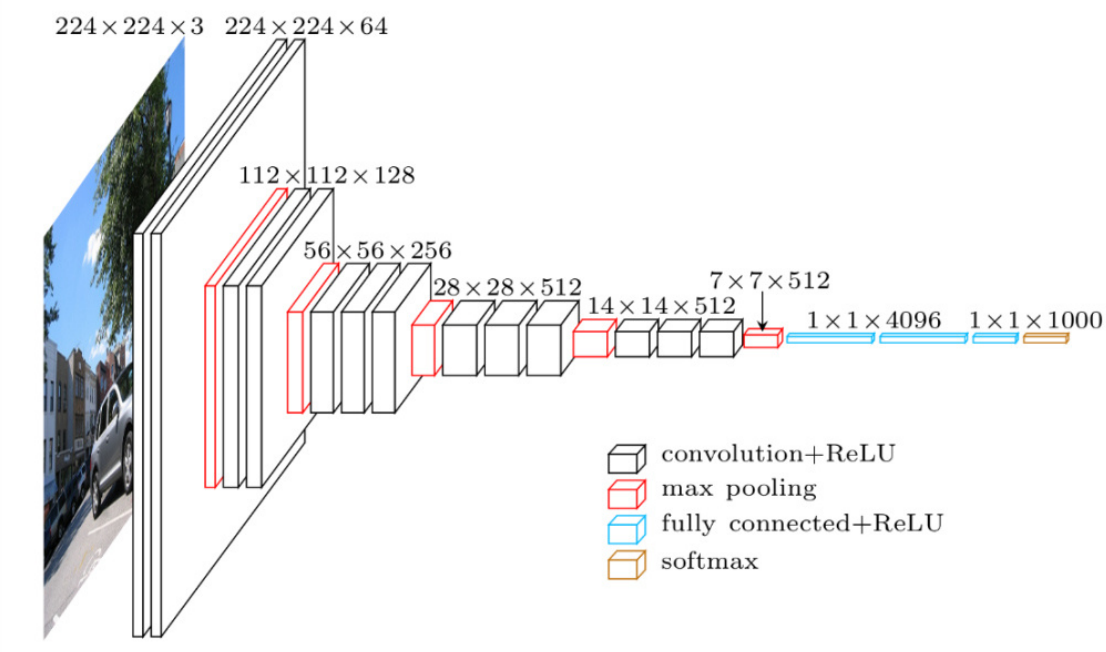


Figure 3.12: The VGG-16 architecture reduces the size of the input after a set of convolutions with the use of *max pooling*, and increases the depth of the filters used in the convolution. With a *softmax* function, the final value is restricted to a number between 0 and 1. Extracted from [3].

of an Inception network, GoogLeNet [19].

3.4 CNN-based detectors

We have seen what a CNN is and some of the most famous ones. The commented architectures are capable of classifying an image, but in most of the cases we are interested in the location of that object. This is generally done by surrounding the detected object by a bounding box, indicating the position of the object in the scene. To obtain this locations, there are different methods that make use of CNNs.

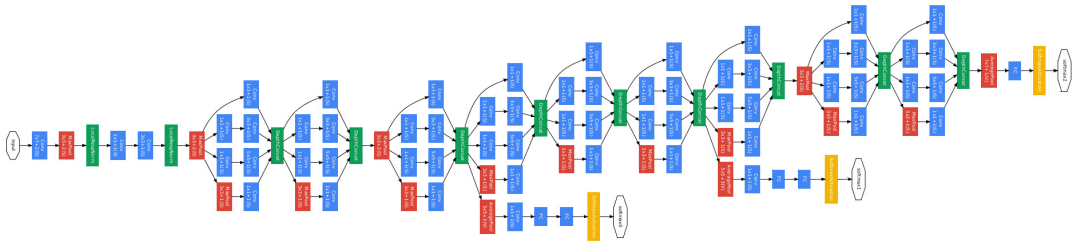


Figure 3.13: GoogLeNet has 9 inception modules stacked linearly. It is 22 layers deep with 5 pooling layers, and uses *avg pooling* at the end of the last inception module. Extracted from [3].

We can differentiate two main groups of detectors based on CNNs: those who first propose the regions and then make the classification in a CNN, and those who make both region proposal and classification in the same network.

- Region based detectors This group of object detectors divide the process in two steps. First, the input image is divided in different regions or *RoI* (Regions Of Interest) where an object could be located. Each region is separately fed into a CNN which makes the classification.

  – R-CNN: Regions with CNN features was proposed by Ross Girshick et al. in [20]. First, the *RoI* are calculated using the *selective search* method [21]. The main steps can be observed in Figure 3.14. First, the image is divided in small regions that are combined into larger ones according to their similarity. Finally, around 2000 regions are proposed. In a second step, these regions are fed into a CNN. The CNN outputs a feature vector to a last layer of SVMs, returning the final score for each class. The biggest disadvantage of R-CNN is the computational cost. Each region needs to be sent to the CNN, making the process very slow for real time

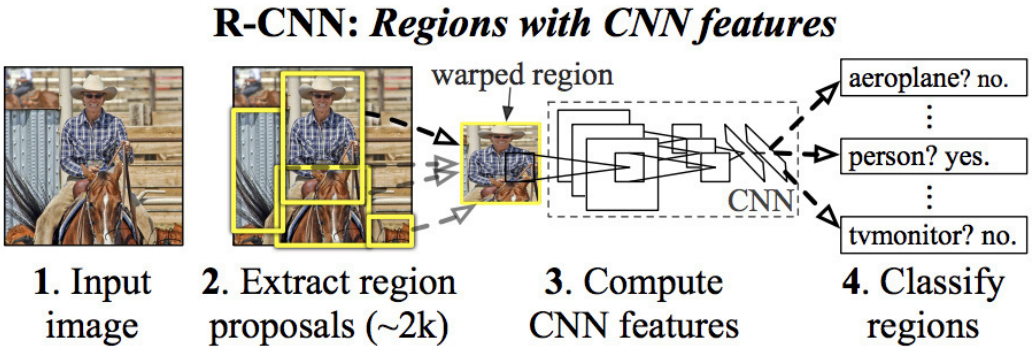


Figure 3.14: First, *ROI* are proposed externally to the CNN. Later, the CNN extract the vector of characteristics of each regions and a last layer of SVMs classifies them.

applications.

- Fast R-CNN: The Fast-CNN architecture [22], from the same authors of R-CNN, comes to solve the speed problems of R-CNN. The main difference can be found in Figure 3.15. This time, instead of proposing the regions externally, the image is directly fed into the CNN to generate the convolutional feature maps. And here it's where the regions are proposed and, by using a *RoI* pooling layer, they are reshaped and fed to the fully connected layer, which classifies them. Additionally, the SVM layer is changed with a *softmax* layer to output the class probabilities.
- Faster R-CNN: The Faster-CNN architecture was proposed by Shaoqing Ren et al. in [23] to improve the speed of the Fast R-CNN. The idea was the same but instead of using the selective search algorithm as Fast R-CNN to obtain the region proposals (making the process slower), a new network is added to learn to extract these regions.

- One-step detectors If the previous group consisted in extract regions and then classify them, this group aims to do both at the same time. These detectors,

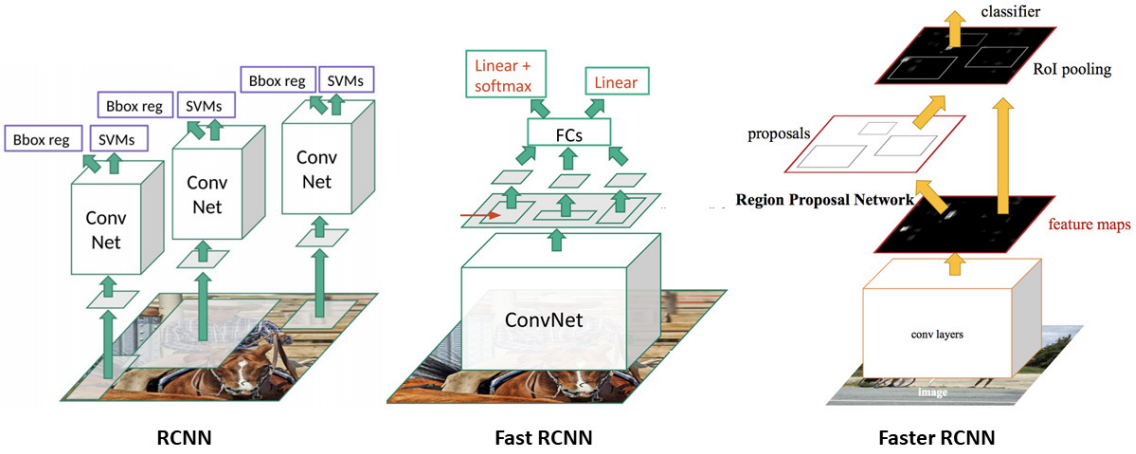


Figure 3.15: R-CNN has an external step where the regions are proposed, and then the CNN process each of these regions. Fast R-CNN changes this by including the region proposal inside the CNN, and extracting the regions from the feature maps instead of from the input image. Finally, Faster R-CNN proposes to use a second network in parallel to obtain the proposals instead of using selective search to find them. Extracted from [7].

also called *one-shot* detectors, prioritize the speed and real time applications.

- SSD: *Single Shot MultiBox Detector*, proposed by C. Szegedy et al. in [24], consists in using an CNN (originally VGG-16) to extract the image features, but substituting the fully connected layers with extra convolutional layers. Figure 3.16 shows the extra layers. As the size decreases with the introduction of more layers, making the small objects in the image hard to detect, SSD uses deep layers of the network for object detection and classification.
- YOLO: The YOLO (You Only Look Once) architecture divides the input image in a grid of SxS regions, and predicts N bounding boxes with a confidence score. This confidence reflects the accuracy of the bounding

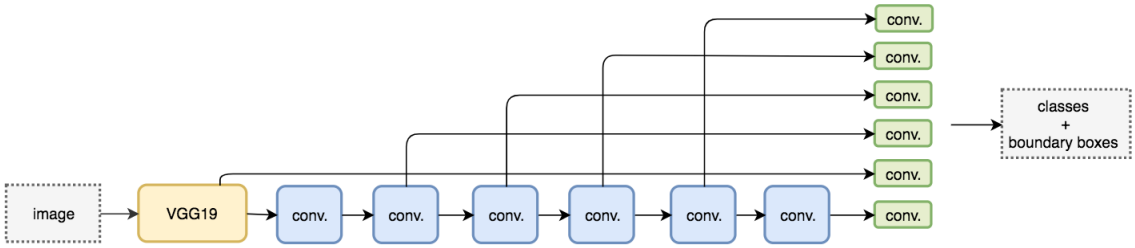


Figure 3.16: SSD uses a CNN in the first steps and then adds extra convolutional layers. To avoid losing the small objects in lower resolutions, SSD uses the output of different layers to predict bounding boxes and classify them

box and whether the bounding box contains an object (of any class) or not. In parallel, YOLO also predicts the classification score for each bounding box for every class in training. Combining both scores you can calculate the probability of each class being present in the predicted bounding box (figure 3.17).

3.4.1 Object detection methods for omnidirectional cameras

We have reviewed some of the most important state of the art algorithms for object detection. However, all this algorithms work under the assumption that the object's appearance remains very close to the original and they are spatially invariant. Although there are some works that tries to use this principle [25] [26], in omnidirectional cameras this is not true. Omnidirectional cameras, as commented before, introduce great distortions in the objects, causing that an object in the center of the image looks completely different from an objects in the border of the image. Other works apply a sphere-to-plane projection [27] in the input images to feed it to a CNN. However, there are different problems that deep learning applied to

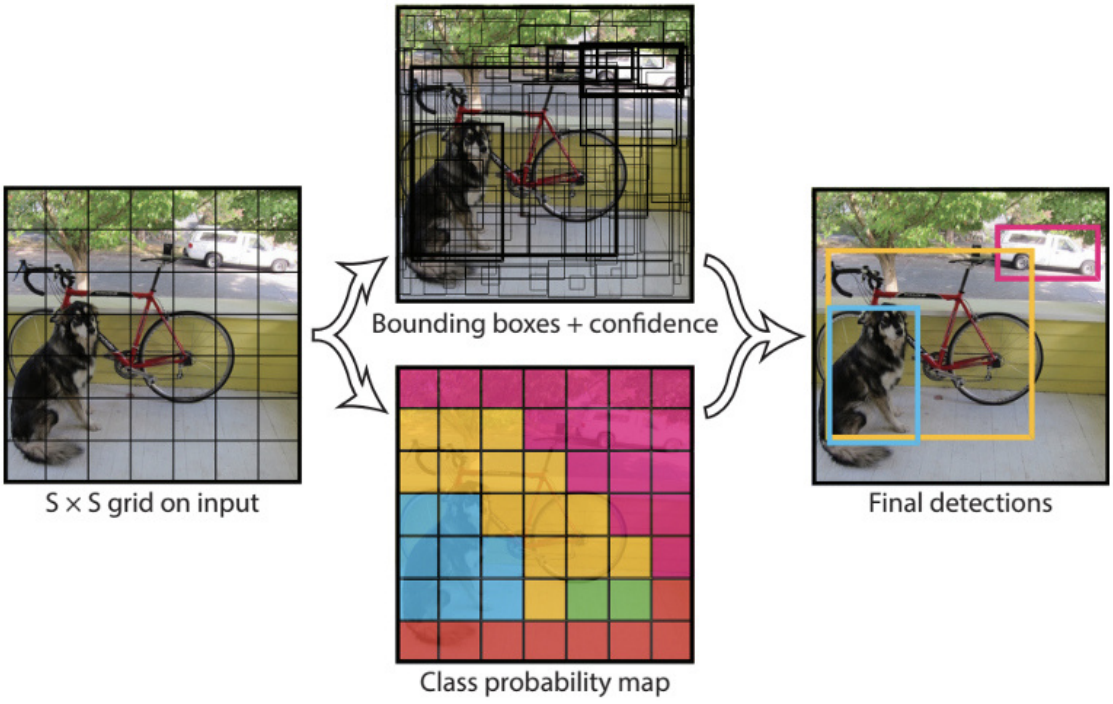


Figure 3.17: YOLO proposes a class probability for each proposed region. However, most of this confidences are very low and can be easily removed with an appropriate threshold. Extracted from [8]

omnidirectional cameras have to face.

One of the main problems of using CNNs in omnidirectional cameras is the lack of data. Data is hard to obtain and annotate, and most of the existing databases are for conventional cameras. Additionally, the data is usually annotated in perspective-like formats, like using bounding boxes that do not take into account the distortion of the objects. Some works like [28] propose the use of adapted bounding boxes to spherical images and extract spherical regions proposals as a previous step to an R-CNN.

Other main problems of CNNs is that the convolutional filters learn invariant

features assuming they won't change in space. This assumption works in conventional cameras but it's not applicable to omnidirectional cameras. In that line, other works like [29] and [30] propose a different approach: adapting the convolutional filters to the spherical image representation. In [31] the authors create a new convolutional kernel for CNNs using linear combinations of differential operators weighted by learnable parameters to substitute the filters in convolutional layers.

3.5 Grid of Spatial-Aware Classifiers

In [1] the idea was to use a grid of SVM classifiers to detect objects in omnidirectional cameras robustly and fast enough to work in real time applications, improving the previous methods from the state of the art based on sliding windows.

3.5.1 Architecture

The method starts obtaining a single descriptor for each image. This feature vector is fed to a grid of classifiers where each classifier is associated to a location in the image. This classifiers are SVMs that will learn the appearance of the objects from its attention area or *fovea*. The last step consists in a fusion of the classifiers outputs to obtain the final position of the object in the image.

3.5.2 Training

The used algorithm for feature extraction is HOG [32]. This is applied to the whole image to obtain one single descriptor that, together with the ground-truth of active classifiers, will be the information used for the SVMs of the grid to learn. One of the main differences in this architecture with respect to other algorithms is that the

ground-truth of positions is point-based. It means that the position of an object (in this case a person) is not given by the traditional bounding box, but by a point located in the head, shoulder or center of the person. This positions will activate just some classifiers of the grid: those in which the object's point location lays inside the fovea, and the rest will be marks as no active. This will create a ground-truth of active classifiers where just the SVMs of the grid where there is a person will be marked as active.

3.5.3 Testing

Once the SVMs of the grid are trained, an input image will generate a confidence score for each classifier. Each classifier will have an estimated confidence about how sure that classifier is active or, in other words, how probable is for a person to be in the region belonging to that classifier. It's assumed that the classifiers with the highest confidences are the ones that ideally are closer to the ground-truth position of the person.

First, a threshold (*hyperPlaneThreshold*) is set to decide if a classifier is active. The values above the threshold are consider active classifiers while the rest is marked as not active. This step by itself is not enough to remove false detections, so a second filtering step is applied as shown in Figure 3.18. A group confidence is calculated in neighbourhoods of active classifiers in the grid. These neighbourhoods can contain active and no active classifiers, and the same classifier can belong to different neighbourhoods. The group confidence is then calculated summing the confidences of the classifiers in it, and then filtered out with a minimum group threshold (*MinGroupThreshold*). Finally, a process of *NMS* is applied to keep the group of classifiers (the neighbourhood) with the highest group confidence. The classifiers in this neighbourhood will be the ones used to calculate the location of

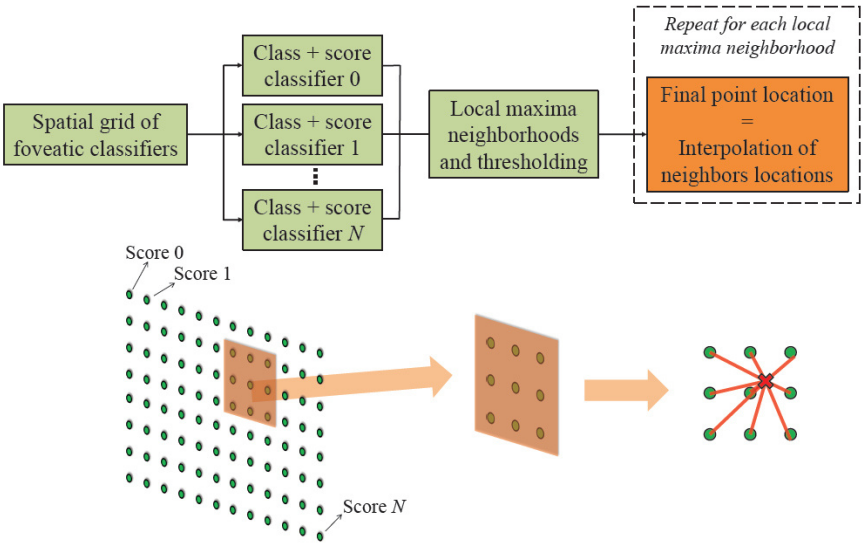


Figure 3.18: A group score is calculated adding the scores inside a neighbourhood and then a threshold is applied to remove activations with a low score. Then, a NMS (Non-Maximum Suppression) is applied to each neighbourhood's score to get rid of multiple detections for the same person. The final location is calculated by the average position (interpolation) of the classifier's scores in the neighborhood. Extracted from [1].

the person by an average sum of its confidences. Assuming that the classifiers with the highest confidences are the ones that ideally are closer to the ground-truth position of the person, the confidence of each classifier could be understood as a weight and therefore the position can be calculated as the average position between them:

$$x, y = \sum_{i=0}^{N_n} \alpha_i x_i, \sum_{i=0}^{N_n} \alpha_i y_i \tag{3.1}$$

Where N_n is the total number of active classifiers in the neighborhood, α_i is the confidence value of classifier i and x_i, y_i are the coordinates of that classifier.

3.5.4 Adaptation to deep learning

In [7] the possible use of features vectors extracted from the last layers of different CNNs is studied. The goal is to analyze if the extracted characteristics in the last layers of some of the most famous CNNs could be used as an input to the grid of SVM classifiers, instead of using HOG. For the different CNNs used refer to [7].

The main disadvantage in [1] is that the training can only be done for one camera. It means that when deploying a camera system, each camera needs to be trained independently.

We will see more details about it in section 4 of this thesis. This work tries to push that idea of introducing deep learning a little further, substituting the whole training and detection process with a CNN, teaching a neural network to activate classifiers according to the characteristics extracted by the network itself from the images.

Chapter 4

Proposed System

4.1 Introduction

The developed system focuses on the detection of objects on static, omnidirectional cameras. We are going one step further in the work done in [1] and [7] trying to implement an architecture based on CNNs that can work end-to-end, extracting the features from the image and classifying in the same network.

4.2 System overview

The main idea is to train a neural network that will learn to predict the position of an object (in our case a person) based on a grid of points placed over the image. The proposed system applies for any CNN, although in this work we have used Alexnet and Resnet architectures.

Figure 4.1 shows the main steps of the process and the differences with the system proposed in [1].

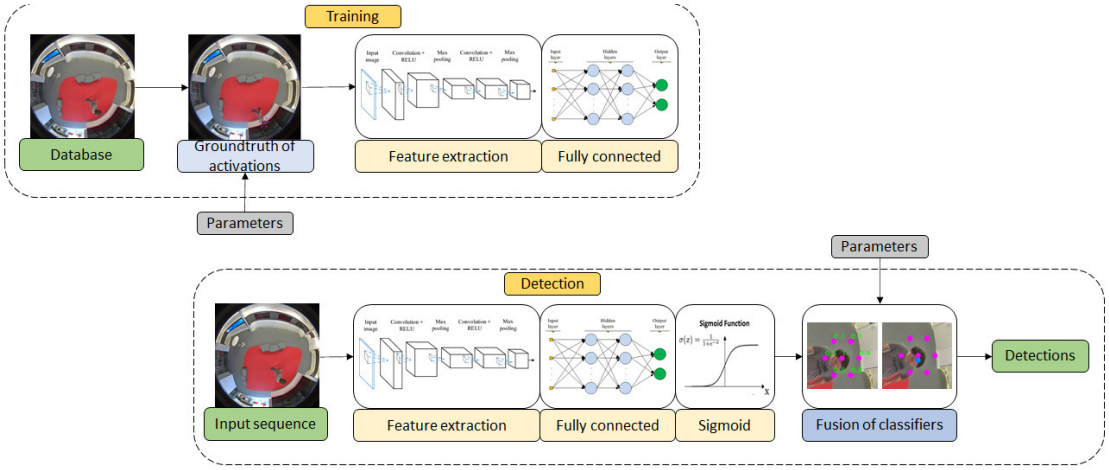


Figure 4.1: Workflow of the proposed system. In the training module the database is first adapted to the CNN: the images are resized and the ground-truth of classifiers is calculated from the ground-truth locations. This part is inherited from [1]. Then, the images are fed to the feature extraction module, and the vector of features is sent to the fully connected layers to obtain a vector of N classifiers. In the detection step, the sequences are fed to the trained network (feature extraction and fully connected modules), and the obtained vector of N classifiers is sent to a *sigmoid* function to obtain a prediction score in between 0 and 1. These scores are finally used in the *fusion of classifiers* module, inherited from [1], where the final position is obtained as described in Section 3.5. Image for the feature extraction module adapted from [9].

First the ground-truth of activations is calculated from the ground-truth of positions using the module from [1]. Here the shape and number of classifiers in the grid (Figure 4.2) are eligible parameters and the effect in the performance was studied in the commented work. A grid with just a few classifiers will not learn to locate the position of a person with much accuracy. On the other side using a large number of classifiers will locate a person more accurately but will make the classifiers very

(a) Rectangular (b) Hexagonal (c) Circular

Figure 4.2: The shape of the grid and the number of SVMs can be chosen. These are important parameters to achieve good results. Extracted from [1].

specific, learning features from a very small area and not general characteristics. It's then important to find a balance between the accuracy and the number of classifiers.

Later, the CNN extracts the characteristics of the input image and outputs a vector with the predictions for the grid of classifiers. These predictions consist in a vector of confidence scores indicating the probability of a person being in that region on the image, and are calculated applying a *sigmoid* to the output of the last fully connected layer of the CNN. The output is a vector of size N, being N the number of classifiers in the grid. A single image will always activate several classifiers, with an independent probability for each of them. As this is not a single classification problem with just one class as a solution but a multilabel problem, we need a function which outputs a probability for each classifier of the grid. We use the non-linear function *sigmoid*. This function also makes sure that the probability is a number between 0 and 1.

Finally, the same module used to obtain the final position in [1] is used here to calculate the position of the person from the predicted classifiers.

4.3 Training

For this task, images of omnidirectional cameras from different scenarios (different cameras) are used to avoid over-fitting to one camera (an analysis of the data is done in Section 5.2). We make use of *fine-tuning* to train the network with images from PIROPO dataset. The network uses the already pre-trained weights (*imageNet* for both) and update the parameters (weights) of the network to train it. There are two reasons to do this: first, using *fine-tuning* with a different set of images generalize better the network, specially when the amount of training data is limited. Second, training a network from scratch has a high computational cost and using an already pre-trained network can reduce it considerably.

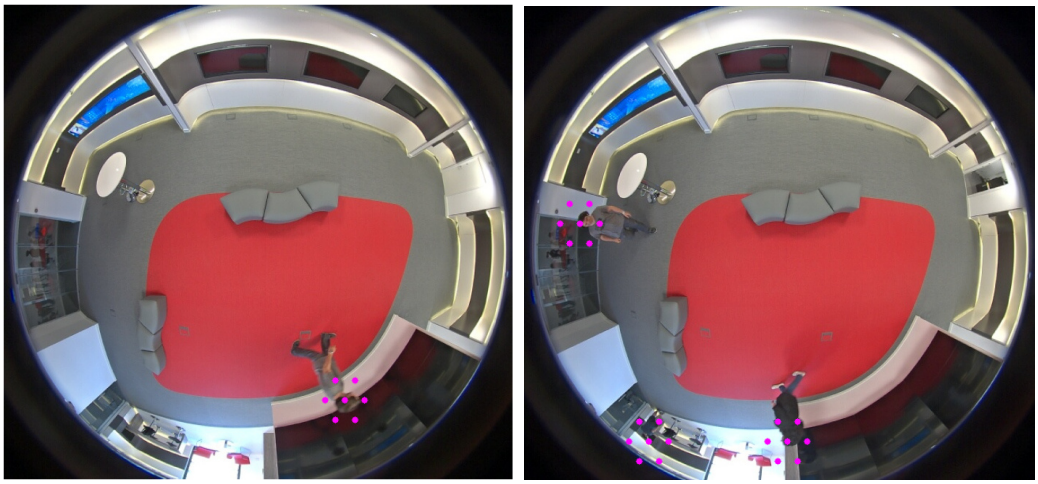
The number of classifiers activated by a person in one image is low if we compare it the total number of classifiers. This inevitably causes a problem of class imbalance, in which we have less positive samples than negative, biasing the network in favour of negative samples. To avoid the network to learn to predict mostly 0s (non-active classifiers), it is necessary to weight the positive samples in a proportion inversely proportional to the number of negative samples in the database. In this case, we opted for a simple weighting in which the weight for a positive sample is calculated according to the ratio of positive and negative samples of each classifier (the number of times the classifier of the grid has been active and not active in each image of the database):

$$W_i^{pos} = \frac{N_i^{no-active}}{N_i^{active}} \tag{4.1}$$

Where W_i^{pos} is the weight applied to positive samples of classifier i and N_i^{active} and $N_i^{no-active}$ are total number of times that classifier i is active and not active, respectively.

Finally, data augmentation techniques can be applied as long as they respect the spatial relation between the grid of classifiers and the image. Some examples of possible transformations could be modifying the intensity levels or converting the images to grayscale. This will be discussed deeper in Chapter 5.

4.4 Testing



(a) (b)

Figure 4.3: Active classifiers over (a) one person and (b) three people. Extracted from [1].

Once the training of the CNN is complete, the last step is to calculate the predicted position of a person in the image from the predicted scores of the grid of classifiers. The scheme is identical to the one presented in Section 3.5 except for the origin of the obtained scores: in [1] the confidences were the output of the SVM but in this work they are the result of a *sigmoid* applied at the end of the CNN. Therefore, to decide whether a classifier is active or not, a different threshold called

MinThreshold is set. These predictions are expected to consist in a small group of active classifiers in a region around the person as shown in Figure 4.3.

Chapter 5

Results

Content

5.1 Introduction

In this chapter we will discuss the experiments and results we had during the development of the project. We will start explaining the database used and the parameters selection. Then we will describe the different experiments done with CNNs and the proposed improvements.

5.2 Datasets

The used dataset has been the public PIROPO (*People inIndoor Rooms with Perspective and Omnidirectional cameras*) dataset [2]. This dataset contains sequences from both omnidirectional cameras and traditional cameras of indoors scenarios. Only the omnidirectional images has been used for this work. There are four different cameras from different scenarios that can be seen in Figure 5.1: The dataset consist in two different rooms. Room A is an open area covered by 8 cameras: three omnidirectional (*omni_1A*, *omni_2A* and *omni_3A*) and five conventional ones. Room B is a smaller area but with more objects in the scene: tables, chairs and computers. There are two cameras in this room: one omnidirectional camera (*omni_1B*) and one conventional camera. The field of view of the omnidirectional cameras is 187º and the size of these images is $800x600$ with a frame rate of around 10 fps. All the cameras are synchronized and recording at the same time, sharing the content of the scenes among them.

This database was created as part of the work [1]. In this work the ground-truth for the sequences was created point-based, i.e. the position of a person is given by a point located in the head. In that work the idea was to generate a database with a reduced number of people to check if the training could generalize enough to be used

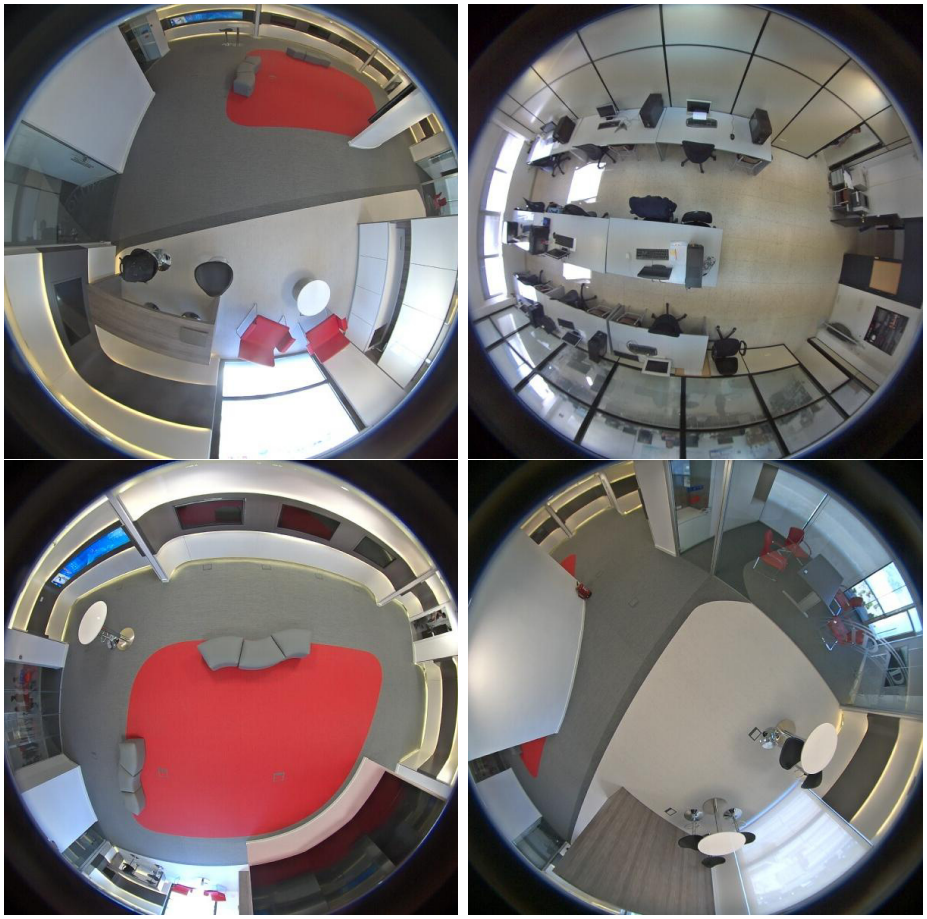


Figure 5.1: Views for some of the cameras contained in the Dataset. In order, *omni_1A*, *omni_1B*, *omni_2A* and *omni_3A*.

in an extended group of people. With this idea four test sequences were created, simulating different scenarios and possible situations. The information about the sequences for training and testing can be found in Table 5.1.

5.2.1 Adaptation to current system

As the used networks need square images as input, it's necessary to adapt the image shapes before feeding the CNN. Given the spatial relationship of the grid of

Sequence	Description
training	1 person walking around the room
test1	1 person present in the training walking around the room
test2	Up to three people walking simultaneously
test3	1 person not present in the training walking around the room
test4	1 person walking around the room and standing still

Table 5.1: Information for PIROPO sequences. The sequences without ground-truth were not used and are not included in the table.

classifiers with the image, cropping the image requires to recalculate the ground-truth of classifiers from the ground-truth positions. This has been done cropping the image horizontally to give the images a size of $600x600$, respecting the circle of vision of the camera, and adapting the ground-truth of classifiers accordingly as explained in Section 3.5.

5.2.2 Other datasets

Part of this work consisted in studying the available dataset for omnidirectional cameras, and focusing on people detection. Table 5.2 show characteristics of these datasets. Although they haven't been used in this project, it is expected to be useful for future works.

5.3 Metrics

The performance of the method has been measured in two different domains: classifier level and detection level. In both domains the performance is measured compar-

	Ground-truth	N of images	Format	N of people in scene	Actions	Camera position	Content type	N of cameras
PIROPO	Yes, point-based	+20.000	*images + .csv*	One or various	Walking	Ceiling	Real	3
***Theodore* dataset** [33]		100.000	*Tfrecord*	Various	Sitting and walking	Ceiling, moving	Artificial	?[3]
***Car and Human* dataset** [34]	Yes, bounding boxes	30	images + .mat	Various	Walking	Ceiling	Real	4[1]
Fisheye* Dataset** [35] - ***LabRoomB	No[2]	443	images	Various	Sitting and walking	Wall	Real	1
Fisheye* Dataset** [35] - ***LivingroomB	No[2]	82	images	One	Walking	Ceiling	Artificial	1
Fisheye* Dataset** [35] - ***HallwayC	No[2]	120	images	One	Walking	Ceiling	Artificial	1
Bomni [36] ***Scene1***	Yes, bounding boxes aprox 1:40 min	17+17 videos,	Videos + .dat 1 view ceiling	One and various	Sitting and walking	1 view wall +	Real	2
Bomni [36] ***Scene2***	Yes, bounding boxes aprox 1:40 min	4+4 videos,	Videos + .dat 1 view ceiling	One and various	Sitting and walking	1 view wall +	Real	2

Table 5.2: Information for different datasets. [1]In the same dataset, 6-8 images from four different scenarios. [2]Not available yet. [3] In the same dataset, each approximately 20 images the scenario changes.

ing the predicted data with the ground-truth. In the classifier level we compare the predicted classifiers with the ground-truth of classifiers. For the detection level, we calculate the metrics using the predicted position and the ground-truth positions. In both cases, the metric used are the *Precision* 5.1, *Recall* 5.2 and *F1-Score* 5.3. The *precision* indicates the fraction of relevant instances among the retrieved instances, while the *recall* is the fraction of the total amount of relevant instances that were actually retrieved. The F1-Score is a useful metric which combines both measures and helps to compare the classification results faster.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.3}$$

Where TP is the number of True Positives, TN of True Negatives, FP of False Positives and FN of False Negatives.

5.3.1 Detection metrics

In the detection step, the metrics are obtained from the ground-truth of positions and the predicted position. To decide whether a point is a TP or not, a distance d between these two points is used as show in Figure 5.2. If the predicted point P_{CNN} lays inside the circumference of radius d centered in the ground-truth point P_{GT}, the sample is considered a TP. Else, it is considered a FP. In case that more than one P_{CNN} lay in the circumference, a greedy algorithm chooses the closest one and marks it as TP and the other as FP.

(a) Defined distance from ground truth point (red).

(b) True Positive (blue).

(c) False Positive (blue).

(d) Greedy matching algorithm - TP (blue) and FP (purple).

Figure 5.2: Possible cases for a predicted position and the ground-truth position. Extracted from [1].

5.4 Experiments

5.4.1 Parameter selection

In this section we are going to explain the fixed parameters used during the experiments. In this work we have tested two main networks: Alexnet [6] and Resnet18 and Resnet50 [18] as CNN in the CNN module in Figure 4.1. We have used Python with the Pytorch [37] library for the classifier prediction module. To obtain the classifier ground-truth and final positions the C++ code provided in [1] has been used.

After some experiments to see the best combination of parameters, we ended up with using this configuration:

- For the grid parameters, we use an hexagonal grid of $25x33$ (i.e. 825 classifiers).
- *Adam* (derived from Adaptive Moment Estimation) [38] as an optimization algorithm given its fast convergence when reducing the loss, and a weighted *BCE* (Binary Cross Entropy) for the loss function for being the most suitable one for binary classification. Other future plans include the use of *Focal Loss* to deal with the imbalance data.
- The *minThreshold* commented in Section 4.2 is set to 0.5, for being the middle point of possible values for the classifiers. The *minGroupThershold* is set to 1.
- The radius d is set to 15 pixels. This is a strict value which assures that the positive detections are as close as possible to the ground-truth.

5.4.2 Comparison with existing works and analysis of proposed system

For the first experiments we compared the existing method proposed in [1] with Alexnet and Resnet18. The experiments consisted in training the grid of SVMs and the CNNs with the training sequences of the four cameras: *omni_1A*, *omni_2A*, *omni_3A* and *omni_1B* to prove that the network was able to generalize better in the CNNs than in the proposed system in [1], where the training was originally individual for each camera. The SVMs was trained with the parameter's values indicated in [1], which are $minGroupThreshold = 1.9$ and $hyperplaneThreshold = 1$. For the CNN architectures we've trained for 200 epochs. The results have been evaluated with the test sequences of all the cameras.

As shown in Table 5.3, the results with the CNNs show a better generalization for all the cameras, proving that a CNN can offer better results. Trying other architectures from the state of the art such as YOLO are part of the future work.

Between the two CNNs, experiments with Resnet18 are in average better than Alexnet. However, looking at the results for each test sequence independently (Table 5.4) we can see that the precision and recall in different test and cameras suffer from high variation. While for the *test1* sequence the results are above 94% for precision, recall and F1-score in three of the cameras and above 88% for *omni_1A*, other test sequences show low performance results. This is coherent with the fact that *test1* contains people from the training set.

The first limitation found is that the recall for *test2* in all the cameras is very low, which means that we are not detecting people most of the time. The reason behind this is the number of people that appear on the scene. As commented in Section 5.2, while in other test sequences the number of people who appear simultaneously is

		HOG+SVM [1]	**Alexnet**	**Resnet18**
omni_1A	Precision	0.2234	0.7814	0.8726
	Recall	0.6188	0.5951	0.6403
	F1-Score	0.3069	0.6578	0.7160
omni_1B	Precision	0.2847	0.6985	0.8117
	Recall	0.5986	0.6238	0.6702
	F1-Score	0.3796	0.6464	0.7016
omni_2A	Precision	0.1987	0.6283	0.7853
	Recall	0.6429	0.6411	0.7044
	F1-Score	0.2920	0.6063	0.7004
omni_3A	Precision	0.1886	0.7201	0.8904
	Recall	0.5727	0.5364	0.4557
	F1-Score	0.2634	0.6049	0.5335

Table 5.3: Comparison with SVM from [1], Alexnet and Resnet. The table show the aggregate results of the four test sequences (three for *omni_1B*) for each camera. In average, the results from Resnet18 show a better performance than Alexnet.

always one, in *test2* this number is three. The results are coherent with the fact that in all the training sequences the maximum number of people at the same time in the room is just one person so the network is biased to towards single detections.

Another limitation found is in camera *omni_3A*. In this case the recall for both *test3* and *test4* is very low. The problem here is that we are not detecting the person in the sequence in most of the frames. Our first hypothesis was that this could be caused for a low number of images with people (i.e. a low number of images with positive samples) in the training sequence for *omni_3A*. This was making the network learn that images with that specific background don't usually

Resnet18		*omni_1A*	*omni_1B*	*omni_2A*	*omni_3A*
test1	Precision	0.8821	0.9572	0.9408	0.9450
	Recall	0.8831	0.9702	0.9408	0.9497
	F1-Score	0.8826	0.9637	0.9408	0.9474
test2	Precision	0.8008	0.6241	0.8093	0.8961
	Recall	**0.2600**	**0.1865**	**0.2507**	**0.5568**
	F1-Score	0.3926	0.2872	0.3828	0.6868
test3	Precision	0.8962	0.8528	0.6999	0.8235
	Recall	0.7532	0.8538	0.8972	**0.0415**
	F1-Score	0.8185	0.8538	0.7863	0.0791
test4	Precision	0.9115		**0.6912**	0.8971
	Recall	**0.6671**		0.7291	**0.2748**
	F1-Score	0.7704		0.7096	0.4207

Table 5.4: Resnet18 metrics for the test sequences of each camera. In black, those values specially low in comparison to others. The *test2* sequences in all cameras and the cameras *omni_3A* have specially low recall.

contain people. We can see that the number of images with an actual person is less than half the number of training images for that camera in Table 5.5.

Finally, in *test2* and *test3* of camera *omni_2A* the precision reduction comes from some punctual false detections in the lower part of the scene (Figure 5.3, probably caused by the illumination of the scene. Aditionally, in sequences *test3* and *test4* of camera *omni_1A* the recall is also reduced because there are some people in the borders of the scene that are not being detected.

With these three challenges in mind, we thought in different options to solve them.

	Number of images	Number of empty images
omni_1A	10969	3032
omni_1B	4584	398
omni_2A	10401	3148
omni_3A	10975	5997

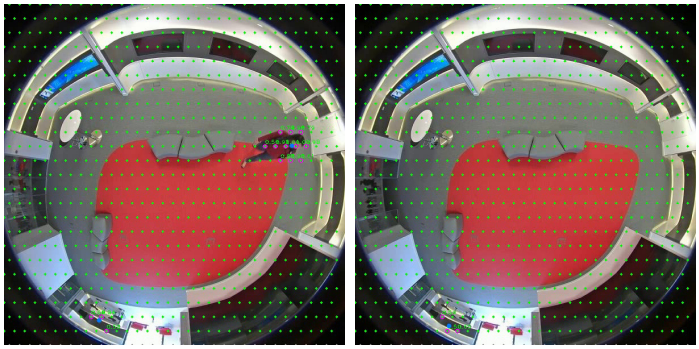
Table 5.5: Number of images in each training sequence and number of empty images. An empty image is defined as those images where there are no people and the image represents the background.

5.4.3 Use of data augmentation

For the first and third limitation encountered we proposed two different techniques of data augmentations:

- As a solution to the first problem with the sequences of more than one person, our hypothesis was that adding more images with multiple people would make the network to learn that more than one person can be in the scene at the same time. The idea was to generate artificial images adding people from one image to another. To do this, first a pair of random images was selected from the training database. Using the Mixture Gaussian model for background subtraction (Figure 5.4) the person was cut from one image and pasted into the other. Through this process, we created an extra 33% of random synthetic images from each training sequence, that was later used (together with the original database) to train the network.

- To solve the third limitation related with the color and intensity changes in the scene, a different data augmentation was proposed. As commented in Chapter 4, data augmentation is always possible as long as the shape of the image is



(a) (b)

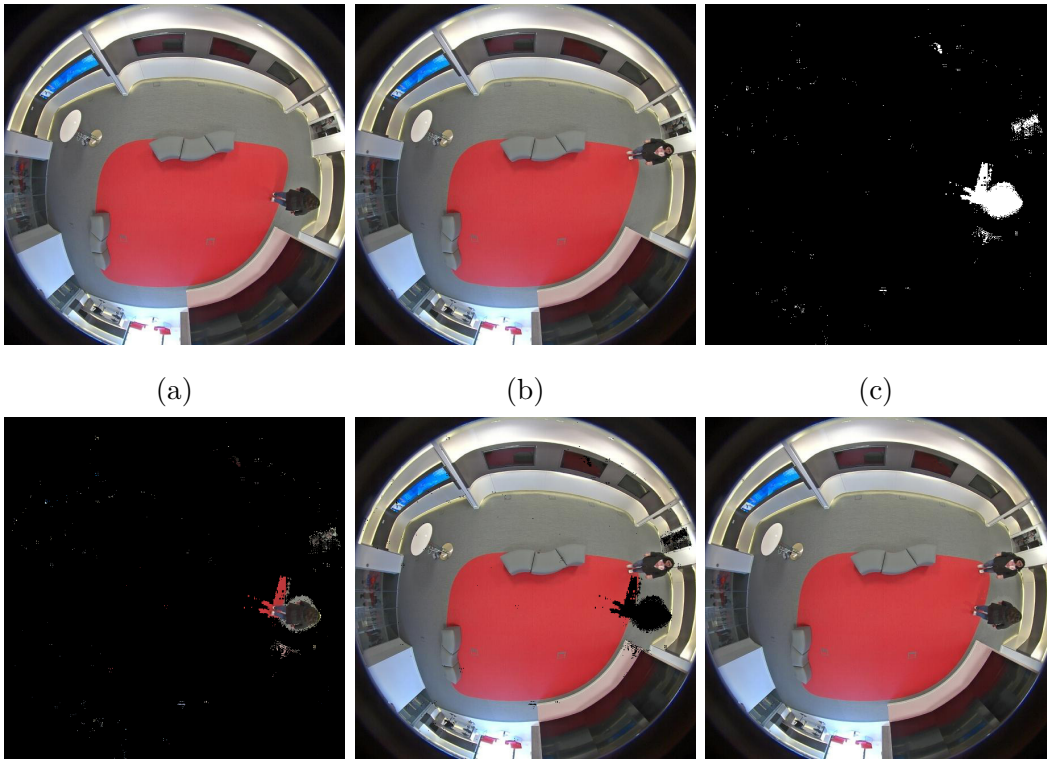
Figure 5.3: (a) shows in person not being detected in camera *omni_1A*, sequence *test3*. (b) shows an example of the false detection that appears in camera *omni_2A*, sequence *test3*.

respected. In this case, both changes were applied to the original database with a probability of 33% :

- Convert the image to grayscale.
- Randomly change the brightness, contrast and saturation of the image.

5.4.4 Reducing the number of negative samples

In the second case the our hypothesis was that the problem didn't come from appearance, but for the number unbalance in the number of positive and negative samples in the camera *omni_3A*. The proposed solution was to remove all the frames that were empty, i.e. all the frames were none of the classifiers are active, as those frames are almost identical and consist in the background of the scene as can be seen in Table 5.5.



(a) (b) (c)

(d) (e) (f)

Figure 5.4: (a) and (b) show the original images. Using a background subtraction algorithm, a mask of the person is created (c). The mask is used to obtain the person (d) and the inverse of the mask is used on the other image (e) to avoid overflow when adding them. Finally the person is added to generate a synthetic image (f).

5.4.5 Results for the proposed solutions

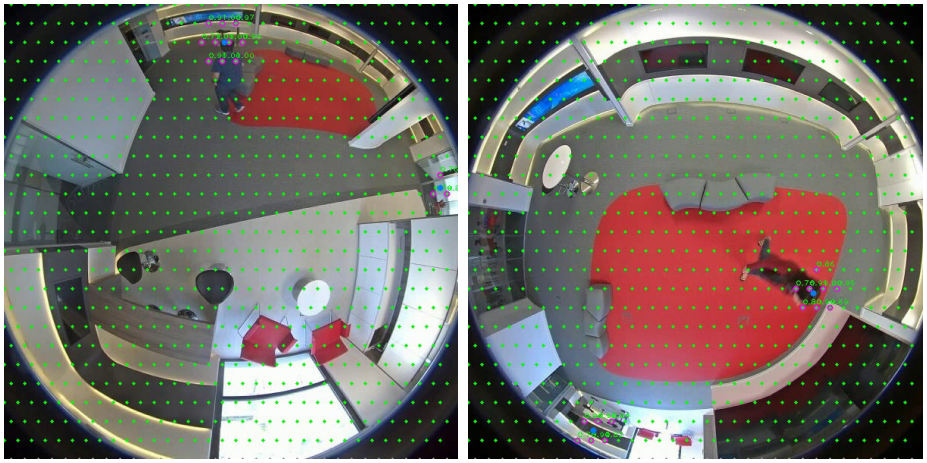
After analysing the previous problems and implementing the proposed solutions, three different experiments have been done. The aggregate results for each camera can be observed in Table 5.6.

omni_1A					
		No data augmentation	**Data augmentation: Grayscale and intensity changes**	**Data augmentation: Empty frames removed**	**Data augmentation: Synthetic people added**
*test*1	Precision	0.9313	0.9331	0.6531	0.9195
	Recall	0.9359	0.9301	0.7620	0.9254
	F1-Score	0.9336	0.9316	0.7034	0.9224
*test*2	Precision	0.7826	0.8199	0.6921	0.8092
	Recall	0.3135	0.2893	0.3200	0.7456
	F1-Score	0.4477	0.4277	0.4376	0.7761
*test*3	Precision	0.8183	0.6069*	0.6792	0.6113
	Recall	0.6364	0.4855	0.3674	0.8112
	F1-Score	0.7160	0.5395*	0.4769	0.6972
*test*4	Precision	0.8333	0.5746*	0.7973	0.5570
	Recall	0.5570	0.2752	0.2890	0.8497
	F1-Score	0.6677	0.3723*	0.4242	0.6729
Improvement of more than 0.1 Loss of more than 0.1					

Table 5.6: Results for Resnet18. The results are the average of the test sequences for each camera. The experiment with the higher performance is the one using synthetic images (fourth column). The values marked with * had an indeterminate value for one of the sequences. In this case the value is considered 0.

Results for the *data augmentation* solution

The experiment of data augmentation where images were converted to grayscale and intensity values changed randomly didn't return the expected results. With this solution we were trying to make the system robust to illumination changes, forcing the network to learn more general characteristics from human shapes and to trust less in the colour characteristics of the appearance. As commented before, this solution was motivated by test sequences where people not present in the training sequences appear on the scene with different clothes colour. More details can be seen in the extended Table 5.8



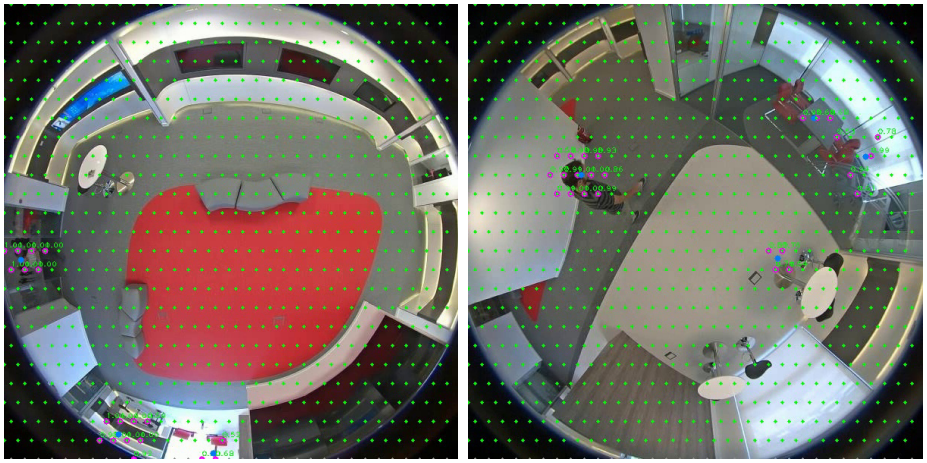
(a) (b)

Figure 5.5: False detections in (a) camera *omni_1A* and (b) camera *omni_2A*. These false activations in the grid are detected as a person, therefore reducing the precision. The effect gets worst after training with the artificial people dataset.

The experiment with the use of synthetic images to include some samples during the training step with multiple people can be seen in the fourth column of Table 5.8. The increase in the recall for *test2* in all the cameras is noticeable. Additionally, other sequences have improved the recall where other techniques have failed, like sequences *test3* and *test4* from the camera *omni_3A*. However, the precision in some test sequences has decreased. The cause of this diminution in the precision comes from the increase in the number of false positives (FP), as shown in Figure 5.5. The false detection suffered in camera *omni_1A* at the left and in *omni_2A* at the bottom becomes almost constant.

Results for the *negative samples reduction* solution

For the experiment where all the empty frames are removed, once more the performance of the network was lower than the base version were no modifications in



(a) (b)

Figure 5.6: False detections in (a) camera *omni_2A* and (b) camera *omni_3A*. The use of synthetic images in Resnet50 increases the number of false positives more than in Resnet18.

the dataset were done. The extended results can be seen in the third column of Table 5.8 This was not the expected results as the idea was to remove those frames that were only negative samples. The reason behind these results is not clear and more research needs to be done in this aspect.

5.4.6 Experiments with Resnet50

After seeing the previous results, we decided to try a deeper version of Resnet, Resnet50. Two experiments have been done: Resnet50 and Resnet50 with the data augmentation using synthetic images. The results can be seen in Table 5.7. These results show that using a deeper version of Resnet doesn't improve the performance, and although the use of data augmentation seems to work better with respect to the base version Resnet50, its use highly increases the false positives as can be seen in Figure 5.6. This happens in all cameras except *omni_1B*. The problem could

be in the way the classifiers react to different regions of the scene, and its study is part of the future work.

		Resnet18	**Resnet50**	**Resnet50 with synthetic images**
omni_1A	Precision	0.8726	0.7679	0.6548
	Recall	0.6403	0.7185	0.8480
	F1-Score	0.7160	0.7070	0.7115
omni_1B	Precision	0.8117	0.7849	0.8806
	Recall	0.6702	0.6850	0.8787
	F1-Score	0.7016	0.7181	0.8797
omni_2A	Precision	0.7853	0.6897	0.5977
	Recall	0.7044	0.6655	0.8402
	F1-Score	0.7004	0.6382	0.6578
omni_3A	Precision	0.8904	0.8266	0.5459
	Recall	0.4557	0.4994	0.8593
	F1-Score	0.5335	0.5931	0.5936

Table 5.7: The table show the aggregate results for each camera for the detections using Resnet18, Resnet50 and Resnet50 with data augmentation. The results show that using Resnet50 doesn't improve the performance of the network. Surprisingly, the use of data augmentation show some better results with respect to the base Resnet50, but it's use increases the apparition of false positives.

omni_1A					
		No data augmentation	**Data augmentation: Grayscale and intensity changes**	**Data augmentation: Empty frames removed**	**Data augmentation: Artificial people added**
test1	Precision	0.8821	0.8873	0.6383	0.8760
	Recall	0.8831	0.8808	0.8226	0.8780
	F1-Score	0.8826	0.8840	0.7188	0.8770
test2	Precision	0.8008	0.8304	0.6724	0.8453
	Recall	0.2600	0.2739	0.2510	0.6892
	F1-Score	0.3926	0.4120	0.3655	0.7593
test3	Precision	0.8962	0.8179	0.6871	0.4455
	Recall	0.7532	0.1752	0.0716	0.8837
	F1-Score	0.8185	0.8538	0.7863	0.0791
test4	Precision	0.9115	0.8500	0.8298	0.4413
	Recall	0.6671	0.0661	0.1010	0.8523
	F1-Score	0.7704	0.1226	0.1801	0.5815
omni_1B					
test1	Precision	0.9572	0.9807	0.6962	0.9543
	Recall	0.9702	0.9648	0.8509	0.9621
	F1-Score	0.9637	0.9727	0.7659	0.9582
test2	Precision	0.6241	0.7719	0.7213	0.6440
	Recall	0.1865	0.0989	0.1978	0.7236
	F1-Score	0.2872	0.1753	0.3104	0.6815
test3	Precision	0.8538	0.7826	0.5119	0.7907
	Recall	0.8538	0.7774	0.6412	0.7907
	F1-Score	0.8538	0.7800	0.5693	0.7907
omni_2A					
test1	Precision	0.9408	0.9258	0.8184	0.9273
	Recall	0.9408	0.9278	0.8716	0.9293
	F1-Score	0.9408	0.9268	0.8442	0.9283
test2	Precision	0.8093	0.7701	0.6746	0.8435
	Recall	0.2507	0.2382	0.2756	0.7563
	F1-Score	0.3828	0.3639	0.3913	0.7975
test3	Precision	0.6999	0.8271	0.8034	0.3978
	Recall	0.8972	0.8250	0.7421	0.8242
	F1-Score	0.7863	0.8261	0.7716	0.5366
test4	Precision	0.6912	0.8739	0.7159	0.3557
	Recall	0.7291	0.7595	0.5677	0.7915
	F1-Score	0.7096	0.8127	0.6333	0.4908
omni_3A					
test1	Precision	0.9450	0.9387	0.4595	0.9203
	Recall	0.9497	0.9472	0.8615	0.9324
	F1-Score	0.9474	0.9430	0.5994	0.9263
test2	Precision	0.8961	0.9074	0.7001	0.9039
	Recall	0.5568	0.5464	0.5557	0.8130
	F1-Score	0.6868	0.6821	0.6196	0.8560
test3	Precision	0.8235	0	0.7143	0.8113
	Recall	0.0415	0.0000	0.0148	0.7463
	F1-Score	0.0791	0	0.0291	0.7774
test4	Precision	0.8971	0	0.8462	0.8739
	Recall	0.2748	0.0000	0.1982	0.9054
	F1-Score	0.4207	0	0.3212	0.8894
Improvement of more than 0.1 Loss of more than 0.1					

Table 5.8: Results for all the test sequences of every camera. The best performance is obtained with the synthetic images. Other experiments have shown a lower performance than the basic experiment without any kind of data augmentation.

Conclusions and future work

5.5 Conclusions

In this thesis we developed a object detector based on CNNs that can be use end-to-end. We have improved the obtained results in [1] and solving some of its limitations, like the use of different cameras to generalize better during the feature extraction. This work is presented as robust and system based on the idea of a simple detector: extracting characteristic directly from omnidirectional images to detect objects without any previous pre-processing.

In a first experiment we tested different networks, Alexnet and Resnet18 and compared them with [1]. The results show a better performance when using CNN characteristics in average when trained with different cameras, but showed some limitations of the system that we tried to solve proposing different ideas. Some of the limitations found were the lack of detections in scenarios with multiple people, limitations we partially solved using data augmentation techniques. For this problem the proposed idea was to generate synthetic images from selecting two images from the training database, and adding people from one frames into the other artificially. As part of the future work, more synthetic images can be created using images with more people or using more images to create one. Other limitations were related to the number of negative training samples and how it affected the performance of

specific scenarios where the number of frames with no positive samples is very large. We proposed the eliminations of those frames as possible solution but the results showed that the performance decreased and more research needs to be done in this. We tested other data augmentations techniques like intensity changes aiming to reduce false detections probably caused by illumination ant to improve the number of detections in different regions of the image, and we discovered that the effect in the training process doesn't improve the performance as expected.

5.6 Future work

In this work we have tested two well known networks: Alexnet and Resnet. However, the use of other networks with better performance in object detection tasks can highly improve the results and open new possibilities of study. The comparison with other State of the Art detectors trained with omnidirectional images could help in giving a different perspective to the results. Additionally, the used CNNs required an images size of $224x224$. Using architectures that allows bigger resolutions could help in the detection of objects in the external parts of the image.

In this thesis we have use simple techniques of data augmentation but as we have seen the overall results have improved. Given the lack of annotated data for omnidirectional cameras, the use of other existing datasets as commented in 5.2 or thinking in different ways to augment the existing data as commented in 5.5 could be also an important part of the future work. And other important possibility is the use of the focal loss to deal with data imbalance efficiently.

During the development we have found unexpected results in some experiments that are still not clear. CNNs are a powerful tool but sometimes finding the causes of errors could be a non-trivial task given the large amount of parameters and

layers they have. We have seen that some data augmentation techniques can help, like the synthetic images commented before. But sometimes other techniques like intensity changes results in a lower performance of the system. Other ideas of data augmentation could be to rotate the images (taking into account the spatial-relation commented in 5.2).

Bibliography

[1] Lorena García and Pablo Carballeira. Development of an algorithm for people detection using omnidi-rectional cameras. 2016. master thesis.

[2] Pablo Carballeira C. R. del Blanco. The piropo database (people in indoor rooms with perspective and omnidirectional cameras), 2016. https://sites.google.com/site/piropodatabase/.

[3] Student notes: Convolutional neural networks (cnn) introduction. https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/.

[4] Relu, sigmoid and tanh: today's most used activation functions. https://www.machinecurve.com/index.php/2019/09/04/relu-sigmoid-and-tanh-todays-most-used-activation-functions/#rectified-linear-unit-relu.

[5] Neural network models in r. https://www.datacamp.com/community/tutorials/neural-network-models-r.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[7] Nicolás García and Pablo Carballeira. Adaptación de un sitema de detección de personas en cámaras omnidireccionales a descriptores deep learning. 2019. bachelor thesis.

[8] Review of deep learning algorithms for object detection. https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852.

[9] Patrik Kamencay, Miroslav Benco, Tomas Mizdos, and Roman Radil. A new method for face recognition using convolutional neural network. *Advances in Electrical and Electronic Engineering*, 15, 11 2017.

[10] Y. Chen, J. Wang, J. Li, C. Lu, Z. Luo, H. Xue, and C. Wang. Lidar-video driving dataset: Learning driving policies effectively. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5870–5878, 2018.

[11] H. Hu, Y. Lin, M. Liu, H. Cheng, Y. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360 sports videos. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1396–1405, 2017.

[12] Academic ranking of world universities 2016. http://www.shanghairanking.com/ARWU2016.html.

[13] Qs ranking of world universities. https://www.topuniversities.com/universities/universidad-autonoma-de-madrid.

[14] Spanish national research council. https://www.csic.es/en.

[15] Ugur Kart, Joni-Kristian Kämäräinen, Lixin Fan, and Moncef Gabbouj. Evaluation of visual object trackers on equirectangular panorama. pages 25–32, 01 2018.

[16] K-R. Müller, AJ. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. *Artificial Neural Networks: ICANN'97*, pages 999–1004, October 1997.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[20] Jasper Uijlings, K. Sande, T. Gevers, and Arnold Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[21] Jasper Uijlings, K. Sande, T. Gevers, and Arnold Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[22] R. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. Ssd: Single shot multibox detector. 9905:21–37, 10 2016.

[25] Fucheng Deng, Xiaorui Zhu, and Jiamin Ren. Object detection on panoramic images based on deep learning. *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 375–380, 2017.

[26] Roman Seidel, André Apitzsch, and Gangolf Hirtz. Improved person detection on omnidirectional images with non-maxima supression. pages 474–481, 01 2019.

[27] W. Yang, Y. Qian, J. Kämäräinen, F. Cricri, and L. Fan. Object detection in equirectangular panorama. *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2190–2195, 2018.

[28] Pengyu Zhao, Ansheng You, Yuanxing Zhang, Jiaying Liu, Kaigui Bian, and Yunhai Tong. Reprojection r-cnn: A fast and accurate object detector for 360 images. 07 2019.

[29] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. *Computer Vision – ECCV 2018*, pages 525–541, 2018.

[30] Yu-Chuan Su and Kristen Grauman. Learning spherical convolution for fast features from 360 imagery. *Advances in Neural Information Processing Systems 30*, pages 529–539, 2017.

[31] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. Spherical CNNs on unstructured grids. *International Conference on Learning Representations*, 2019.

[32] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:886–893 vol. 1, 2005.

[33] T. Scheck, R. Seidel, and G. Hirtz. Learning from theodore: A synthetic omnidirectional top-view indoor dataset for deep transfer learning. pages 932–941, 2020.

[34] I. Cinaroglu and Y. Bastanlar. A direct approach for human detection with catadioptric omnidirectional cameras. *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, pages 2275–2279, 2014.

[35] A. Eichenseer and A. Kaup. A data set providing synthetic and real-world fisheye video sequences. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1541–1545, 2016.

[36] B. E. Demiröz, İ. Arı, O. Eroğlu, A. a. Salah, and L. Akarun. Feature-based tracking on a multi-omnidirectional camera dataset. *International Symposium On Communications, Control, And Signal Processing (ISCCSP12), Rome, Italy*, 2012.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035, 2019.

[38] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.