

# Universidad de Alcalá

## Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

### Trabajo Fin de Grado

Labeling and evaluation of a new dataset for human action  
recognition in large vessels

**Author:** Rodrigo Antonio Vidal Pinto

**Advisors:** Cristina Losada Gutiérrez and Antonio Carlos Cob  
Parro

2022



UNIVERSIDAD DE ALCALÁ  
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Labeling and evaluation of a new dataset for human action  
recognition in large vessels

Author: Rodrigo Antonio Vidal Pinto

Advisors: Cristina Losada Gutiérrez and Antonio Carlos Cob Parro

**Tribunal:**

**President:** Luciano Boquete Vázquez

**1<sup>st</sup> Vocal:** Sonia Martín López

**2<sup>nd</sup> Vocal:** Cristina Losada Gutiérrez

Calification: .....

Date: .....



*“Be grateful with everything you have and you will be successful in everything you do.”*

Conor McGregor



# Acknowledgements

Gracias a mi familia, sobre todo a mis padres, César y Jimena, por el apoyo incondicional que me han dado siempre.

Gracias a Ángeles Puebla por haberme dado la ilusión y el interés que tengo actualmente por la ingeniería.

Gracias a Cristina, mi tutora, por su paciencia y las incontables horas que ha dedicado conmigo a este proyecto.

Gracias a Carlos, mi cotutor, por haberme ayudado y enseñado a dar mis primeros avances en este trabajo.

Gracias a Marta por haberme dado la oportunidad de participar en este proyecto.





# Resumen

El objetivo de este Trabajo de fin de Grado (TFG) es la generación, etiqueta y evaluación de un nuevo dataset denominado *Human Action Recognition on Ships* (HARS) para el posterior entrenamiento y evaluación de un sistema para la evacuación de personas en grandes embarcaciones, en el marco del proyecto *PALAEMON: A holistic passenger ship evacuation and rescue ecosystem* (H2020-PALAEMON-814962). Las secuencias a etiquetar incluyen diferentes personas realizando distintas actividades y han sido grabadas en un barco disponible en Astilleros de Santander S.A.U. (ASTANDER). Para el etiquetado se ha partido de una herramienta proporcionada por el grupo de investigación GEINTRA, que ha sido modificada para su adaptación a las necesidades de etiquetado del dataset, incluyendo no solo acciones individuales, sino también grupales. Además, se han definido criterios para realizar el etiquetado de las personas y acciones. La evaluación del dataset se ha llevado a cabo utilizando la red neuronal YOLOv3 y realizando una evaluación de los resultados obtenidos en la detección de personas con dicha red a partir de la información etiquetada. La implementación y ejecución de YOLOv3 se ha realizado en Google Colab y los resultados se han comparado con los etiquetados empleando MABLAB. El trabajo desarrollado y los resultados obtenidos han permitido validar el etiquetado del dataset y el cumplimiento de los objetivos del TFG.



# Abstract

The aim of this Final Degree Thesis (TFG) is the generation, labeling and evaluation of a new dataset named *Human Action Recognition on Ships* (HARS) for the later training and evaluation of a system in charge of person evacuation in large cruise ships within the framework of *PALAEMON: A holistic passenger ship evacuation and rescue ecosystem* project (H2020-PALAEMON-814962). The different sequences to be labeled inside the dataset include different persons performing distinct activities and have been recorded in a ship available at Astilleros de Santander S.A.U. (ASTANDER). The labeling has been based on a tool provided by the GEINTRA research group and has been modified and adapted to the labeling needs of the dataset including just not individual actions but also group actions. In addition, criteria to perform the labeling process of persons and actions has been defined. The evaluation of the dataset has been carried out using the neural network YOLOv3 and performing an evaluation of the results obtained in person detection with this network from the labeled information. The implementation and execution of YOLOv3 has been carried out in Google Colab and the results have been compared with the labeled ones using MATLAB. The developed work and the obtained results have allowed to validate the labeling of the dataset and the compliance of the objectives of the TFG.



# Contents

<b>Contents</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xxi</b>
<b>List of Acronyms</b>	<b>xxiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Objectives . . . . .	1
1.3 Memory Structure . . . . .	2
<b>2 Theoretical foundations</b>	<b>5</b>
2.1 Introducción . . . . .	5
2.2 Datasets for people detection . . . . .	5
2.2.1 ImageNet . . . . .	5
2.2.2 Open Image . . . . .	7
2.2.3 COCO . . . . .	8
2.2.4 Summary . . . . .	9
2.3 Datasets for action recognition . . . . .	10
2.3.1 UCF-101 . . . . .	10
2.3.2 NTU RGB+D . . . . .	11
2.3.3 Kinetics (Kinetics Human Action Video Dataset) . . . . .	12
2.3.4 Summary . . . . .	12
2.4 Group action datasets . . . . .	13
2.4.1 Collective activity dataset . . . . .	13
2.4.2 Atomic Group actions . . . . .	13
2.4.3 UMN . . . . .	14
2.4.4 BEHAVE . . . . .	14
2.5 Artificial Neural Networks . . . . .	15

---

2.5.1	Activation functions	16
2.5.2	ANN structure	20
2.6	Convolutional Neural Networks	21
2.7	People and object detection with YOLO	22
2.7.1	YOLO features	23
2.7.2	Network Design	24
2.7.3	Limitations	25
2.8	YOLOv3	25
2.9	Conclusions	27
<b>3</b>	<b>Development</b>	<b>29</b>
3.1	Introduction	29
3.2	HARS dataset	30
3.2.1	Recording setup	30
3.2.2	HARS dataset characteristics	33
3.3	Dataset labeling	35
3.3.1	Labeling tool	35
3.3.2	Labeling criteria	38
3.3.2.1	Person Labeling criteria	39
3.3.2.2	Individual actions labeling criteria	41
3.3.2.3	Group actions criteria	46
3.3.2.4	Important considerations about the labeling process	47
3.4	Labeling result	48
3.5	Dataset evaluation	48
3.5.1	YOLOv3 implementation in Google Colab	48
3.5.2	Data processing for evaluation	52
3.5.2.1	Preparing Ground Truth file for evaluation	53
3.5.2.2	Preparing YOLO coordinates file for evaluation	54
<b>4</b>	<b>Results</b>	<b>57</b>
4.1	Introduction	57
4.2	Experimental setup	57
4.2.1	Metrics used for evaluation	57
4.2.1.1	Classification metrics	57
4.2.1.2	Intersection over Union (IoU)	58
4.2.1.3	AUC and ROC curves	58
4.2.2	Preparing the data for evaluation	60
4.2.2.1	IoU matrix	60

---

4.2.2.2	YOLO accuracy matrix . . . . .	61
4.3	Experimental results . . . . .	62
4.3.1	RGB vs IR . . . . .	63
4.3.2	Lighting conditions . . . . .	65
4.3.3	Depth and occlusions . . . . .	66
<b>5</b>	<b>Conclusions and future issues of research</b>	<b>69</b>
5.1	Conclusions . . . . .	69
5.2	Future issues . . . . .	69
<b>6</b>	<b>Tools and resources</b>	<b>71</b>
6.1	Hardware requirements . . . . .	71
6.2	Software requirements . . . . .	71
<b>7</b>	<b>Budget</b>	<b>73</b>
7.1	Hardware resources . . . . .	73
7.2	Software resources . . . . .	73
7.3	Human resources . . . . .	74
7.4	Total cost budget . . . . .	74
	<b>Bibliography</b>	<b>75</b>





# List of Figures

1.1	General flowchart of the different stages developed in this TFG . . . . .	2
2.1	ImageNet dataset organization according to WordNet sample structure (extracted from [1])	6
2.2	ImageNet dataset sample including bounding boxes (extracted from [1]) . . . . .	7
2.3	Open Image dataset sample that includes bounding boxes (extracted from [2]) . . . . .	8
2.4	COCO segmentation process sample (extracted from [3]) . . . . .	8
2.5	COCO keypoint detection process sample (extracted from [3]) . . . . .	8
2.6	COCO panopting segmentation process sample (extracted from [3]) . . . . .	9
2.7	COCO dense pose process sample (extracted from [3]) . . . . .	9
2.8	Sample images from UCF-101 dataset (extracted from [4]) . . . . .	10
2.9	Distribution of video in actions along the UCF-101 dataset (extracted from [4]) . . . . .	11
2.10	NTU RGB+D dataset sample images.(extracted from [5]) . . . . .	11
2.11	Kinetics dataset sample images (extracted from [6]). . . . .	12
2.12	Collective activity dataset sample images (extracted from [7]) . . . . .	13
2.13	Collective activity dataset labeling distribution . . . . .	13
2.14	Atomic group actions dataset sample (extracted from [8]) . . . . .	14
2.15	UMN group action dataset sample images (extracted from [9]) . . . . .	14
2.16	BEHAVE group action dataset sample (extracted from [9]) . . . . .	15
2.17	Neuron linear and non-linear parts (extracted from [10]) . . . . .	16
2.18	Binary step function representation . . . . .	17
2.19	Some linear step functions representations . . . . .	17
2.20	Some linear step functions representations . . . . .	18
2.21	Sigmoid activation function representation . . . . .	18
2.22	ReLU activation function representation . . . . .	19
2.23	Leaky ReLU activation function representation . . . . .	20
2.24	Input-output relation of an ANN (extracted from [11]) . . . . .	20
2.25	Diagram of the steps followed by a CNN classifier (extracted from [12]) . . . . .	21
2.26	Convolution Operation (extracted from [12]) . . . . .	21
2.27	Pooling Types (extracted from [12]) . . . . .	22

2.28	Basic diagram that contains the steps followed by YOLO to make a detection and assign a bounding box (extracted from [13]) . . . . .	23
2.29	Graphical representation of the YOLO methodology used to perform detections (extracted from [13]) . . . . .	24
2.30	YOLO Network Design (extracted from [13]) . . . . .	24
2.31	Anchor box visual example (extracted from [14]) . . . . .	25
2.32	YOLOv3 general Architecture (extracted from [15]) . . . . .	27
2.33	Comparison of different detectors using COCO dataset (extracted from [13]) . . . . .	27
3.1	Flowchart of this TFG . . . . .	30
3.2	Relevant information about the camera used for the recording process . . . . .	31
3.3	Common area scenario shown from RGB (left) and IR (right) camera perspective . . . . .	31
3.4	Cabins corridor scenario shown from RGB (left) and IR (right) camera perspective . . . . .	32
3.5	Exit stairs scenario shown from RGB (left) and IR (right) camera perspective . . . . .	32
3.6	Exterior scenario shown from RGB (left) and IR (right) camera perspective . . . . .	32
3.7	Lifeboat scenario shown from RGB (left) and IR (right) camera perspective . . . . .	33
3.8	Stair corridors scenario shown from RGB (left) and IR (right) camera perspective . . . . .	33
3.9	Exterior scenario shown from RGB (left) and IR (right) camera perspective . . . . .	34
3.10	Main windows of the annotation interface window before and after modifications made to adapt the tool to the HARS dataset . . . . .	35
3.11	Annotation Interface information . . . . .	36
3.12	Use of the Annotation Interface . . . . .	37
3.13	Example of the labeled Bounding Boxes. . . . .	37
3.14	Generating GT file using the Annotation Interface . . . . .	38
3.15	GT File distribution . . . . .	38
3.16	GT File distribution including more than one person per frame . . . . .	38
3.17	Bounding Boxes delimitations defined for partially covered persons in the scenario . . . . .	39
3.18	Choosing when to start labeling a user . . . . .	40
3.19	Choosing when to end the labeling process of a user . . . . .	40
3.20	Special case 1 example (A person can be seen in the frame but not identified) . . . . .	41
3.21	Stationary action example where a person moves but does not perform any displacement . . . . .	42
3.22	Stationary action considerations coming from performing other of the actions included in the dataset . . . . .	42
3.23	Walking action being labeled due to a displacement between frames from user 0 . . . . .	43
3.24	Walking action considerations . . . . .	43
3.25	Running and Walking differentiation . . . . .	44
3.26	Falling Down action . . . . .	44
3.27	Beginning and end of stairs action . . . . .	45

3.28	Falling Down in stairs action	45
3.29	Stampede action sample	46
3.30	Blockade action sample	47
3.31	Example of poor characteristics of the IR videos	47
3.32	General flowchart of this TFG	49
3.33	Flowchart of the implementation of You Only Look Once (YOLO)v3 in Google Colab	49
3.34	Repository Cloning and Installing	50
3.35	Compile	50
3.36	Downloading of pretrained YOLOv3 weights with COCO dataset	50
3.37	Flowchart representing how to obtain the different outputs	51
3.38	Importing the videos directly from the computer	51
3.39	YOLOv3 video output example	52
3.40	YOLO GT file information format	52
3.41	Ground Truth coordinates distribution	53
3.42	Example of the generated array with Ground truth file information	53
3.43	Reading and storing corresponding frame number	54
3.44	Content of the character array obtained	54
3.45	Example of array in which non-useful detection coordinates have been removed.	55
3.46	YOLO coordinates assigned to each 'person' detection	55
3.47	YoloGTtxt distribution	56
4.1	IoU mathematical definition	58
4.2	ROC curve example	59
4.3	ROC curve visual examples for specific cases.	59
4.4	IoU matrix example	60
4.5	IoU results matrix plot	61
4.6	YOLO accuracy matrix	61
4.7	ROC curves representation for IoU and YOLO accuracy	62
4.8	Histogram representations of IoU and YOLO accuracy	63
4.9	Difference between YOLO detection bounding boxes depending on the video recording setup	64
4.10	Difference between YOLO IoU and accuracy ROC curves	64
4.11	Difference between YOLO detection bounding boxes depending on the video recording setup	65
4.12	RGB ROC curve display	65
4.13	IR ROC curve display	66
4.14	Wrong people detections made by YOLO due to the mirror	66
4.15	ROC curve corresponding to this scenario	67
4.16	Corridor cabins scenario showcase	68
4.17	Corridor cabins ROC curve display	68



# List of Tables

2.1	Main characteristics of the analyzed datasets for person detection . . . . .	9
2.2	Main characteristics of the analyzed datasets for action recognition . . . . .	12
2.3	Backbone comparison (extracted from [13]) . . . . .	26
2.4	Average precision comparison between one stage detectors for different sized objects (extracted from [16]) . . . . .	26
3.1	Specifications of the modules included in the Intel D435 . . . . .	31
3.2	Video, persons and frames distribution of the HARS dataset . . . . .	34
3.3	Video, persons and frames distribution of the HARS dataset . . . . .	48
7.1	Price of the different hardware used in the Final Degree Thesis (TFG) . . . . .	73
7.2	Price of the different software used in the TFG . . . . .	73
7.3	Price of human resources needed in the TFG . . . . .	74
7.4	Total cost budget of the TFG . . . . .	74



# List of Acronyms

ANN	Artificial Neural Network.
ASTANDER	Astilleros de Santander, S.A.U..
AUC	Area Under ROC Curve.
CNN	Convolutional Neural Network.
COCO	Common Objects in Context.
DNN	Deep Neural Network.
DSSD	Deconvolutional Single Shot Detector.
FN	False Negative.
FOV	Field of View.
FP	False Positive.
FPR	False Positive Rate.
FPS	Frames per second.
GEINTRA	Group of Electronic Engineering applied to Intelligent Systems and Transport.
GPU	Graphical Processing Unit.
GUI	Graphical User Interface.
HARS	<i>Human Action Recognition on Ships.</i>
IoU	Intersection over Union.
IR	Infrared.
MIMO	Multiple Input Multiple Output.
MNN	Modular Neural Network.
R-FCN	Region-based Fully Convolutional Network.
ReLU	Rectified Linear Unit.
RGB	Red, green and blue.
RNN	Recurrent Neural Network.
ROC	Receiver Operating Characteristic.

TFG Final Degree Thesis.

TN True Negative.

TP True Positive.

TPR True Positive Rate.

YOLO You Only Look Once.



# Chapter 1

## Introduction

### 1.1 Introduction

During the last decades, continuous advances in processing technologies and communications allows automating more complex tasks and in a more accurate way. Thus, in the last years, several efforts have been done to improve computer vision systems for several applications, such as object [17, 18] or people [19–21] detection, action recognition [22–24], etc. using color or depth data.

Besides, in recent years, the improvements in the processing capabilities and the availability of large image and video datasets (such as Imagenet [25]) have led to a rise in the number of works that solve computer vision problems using Deep Neural Network (DNN), outperforming classical approaches. However, these available datasets are limited to specific types of environments, type of available information, and considered situations or behaviors.

These limitations can generate problems if there are used to train deep learning algorithms that are going to be used in a different context. Due to this, in some works it can be necessary to carry out recording, labeling and evaluation of a new dataset called *Human Action Recognition on Ships* (HARS) dataset, designed specifically to train such deep learning algorithms.

In this context, this TFG arises with the main goal of creating, labeling and evaluating a new dataset for people detection and action recognition in large cruise ships. All of this, within the framework of the *PALAEMON: A holistic passenger ship evacuation and rescue ecosystem* project (H2020-PALAEMON-814962), in which members of the Group of Electronic Engineering applied to Intelligent Systems and Transport (GEINTRA) research group are participating, and whose main objective is to engage innovative technologies in a new intelligent, sophisticated ecosystem of mass evacuation of vessels.

### 1.2 Objectives

As it can be stated in section 1.1, this TFG arises with two main objectives: the creation and labeling of a new video sequences dataset oriented to people detection and action recognition in large cruise ships, and the evaluation of that dataset using a state-of-the-art deep learning method for people detection or action recognition. This global objective can be divided in several specific goals detailed below.

1. Search and study of theoretical and practical aspects related to computer vision and such as artificial neural networks, deep neural networks, datasets used for human and action recognition and alternatives used to speed-up the labeling process.

2. Modification of the available labeling tool to include group actions in labeling and adapt it to the necessities of the [HARS](#) dataset.
3. Labeling the person locations, individual and group actions of the new dataset acquired inside a ferry in Astilleros de Santander, S.A.U. (ASTANDER).
4. Evaluation of the labeled dataset by training and testing a people or action detection approach from the state-of-art, comparing the obtained results with the manual labels.

Figure 1.1 shows a general block diagram of the proposed system.

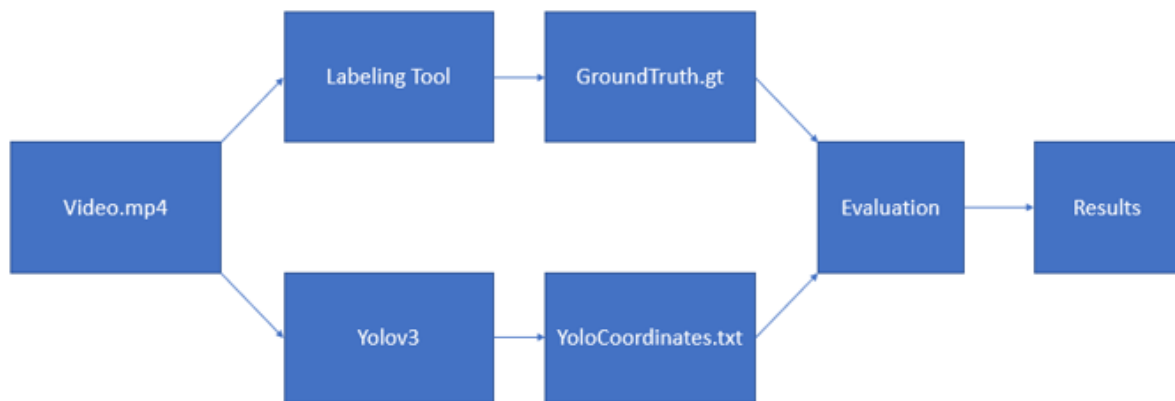


Figure 1.1: General flowchart of the different stages developed in this TFG

Although the main objective of this [TFG](#) is the labeling of actions and persons in this new dataset in order to reach the evaluation stage it is needed to obtain two different files containing the Ground Truth coordinates and the [YOLO](#) [26] coordinates, to obtain those files the videos that form the dataset are fed into our labeling tool for manual labeling and into [YOLOv3](#) to obtain its coordinates information, once done that, the processing of those files and the evaluation methods used lead to results and through in-depth analysis of those the conclusions of this [TFG](#) are obtained.

### 1.3 Memory Structure

This work is structured in 5 chapters, which are set below with a brief explanation of them:

1. **Chapter 1, *Introduction*** in this chapter the context of the need of having labeled datasets for specific contexts is provided as well as the objectives of the [TFG](#) and the structure of it.
2. **Chapter 2, *Theoretical foundations*** contains a study of the most used datasets in the state of art for people detection and individual and group action recognition, this is followed by a summary of the theoretical concepts needed to understand [YOLO](#), the solution selected for the evaluation of the dataset, and an in depth explanation of the characteristics and architecture of it.
3. **Chapter 3, *Development***, describes the work performed in this [TFG](#), begins with a study of the characteristics of the [HARS](#) dataset. Continuing, the tool used for the labeling process is presented as well as the criteria used and the modifications made to it in order to adapt it to the characteristics of interest of the [TFG](#). Following this explanation an implementation of [YOLO](#) in Google Colab

is made to obtain the results needed. Ending this chapter the data processing of the results of the labeling process and the **YOLO** implementation are processed and adapted for the evaluation process later presented in *Results chapter*.

4. **Chapter 4, *Results***, contains an explanation of the metrics used for the evaluation of the **HARS** dataset, then the data obtained and adapted from the *Development chapter* is processed again and the results obtained through this process are displayed, then the global results and the relation between those results and the characteristics of **HARS** dataset set up to see how the different physical aspects of it affect **YOLO** performance is also presented.
5. **Chapter 5, *Conclusions***, includes the conclusions obtained from the results seen in the *Results chapter* as well as the future lines in which how this study can be followed up.
6. **Chapter 6, *Tools and resources***, contains listed the needed hardware and software requirements to carry out the project.
7. **Chapter 7, *Budget***, in which the costs needed for the development of this project are summarized.



# Chapter 2

## Theoretical foundations

### 2.1 Introducción

As the main objective of this thesis is the labeling and evaluation of the [HARS](#) dataset including person position, as well as individual and group actions, in this chapter there are presented different topics that helps the understanding of the methods being used later in this [TFG](#).

At first, there has been carried out an analysis of different available datasets, which are widely used for people detection and for action recognition. There has been analyzed their main characteristics, highlighting the most remarkable aspects of any of them.

Another aspect evaluated in this [TFG](#) is if the [HARS](#) dataset allows working with a neural network such as [YOLO](#) and obtain reasonable results.

In order to understand this, an explanation of how Artificial Neural Network (ANN) and Convolutional Neural Network (CNN) works, is presented followed by an in-depth study of [YOLO](#), the chosen alternative, which later explains the particularities of [YOLOv3](#).

### 2.2 Datasets for people detection

The objective of people a detector systems is to determine if in an image exists people, if so, it should also provide its position in the image. This function is generally performed in two steps, extraction of image characteristics and classification [\[20, 27\]](#), although more recent works, based on [DNNs](#) perform people detection in one step [\[28, 29\]](#).

In order to train or validate those classifiers for people detection, datasets are used. There are some datasets that only includes people, such as [\[30, 31\]](#), however, most of the recent large-scale datasets, that allow training [DNNs](#), include a wide variety of objects, apart from people. Below, there are described three widely used large-scale datasets.

#### 2.2.1 ImageNet

In this section there are explained the main characteristics of the ImageNet dataset [\[32\]](#).

ImageNet is an image dataset that is organized accordingly to the WordNet hierarchy. WordNet [\[33\]](#) [\[34\]](#) is a large lexical database of English vocabulary that includes nouns, verbs, adjectives,

and adverbs which are grouped in sets of synsets (cognitive synonyms) that each express a distinct concept, WordNet contains around 80000 noun synsets and ImageNet aim is to provide on average 500-1000 images to illustrate each synset.

Figure 2.1 represents the organization of the Imagenet dataset according to WordNet sample structure.



Figure 2.1: ImageNet dataset organization according to WordNet sample structure (extracted from [1])

Some characteristics of ImageNet dataset are the following:

- The density of ImageNet is unmatched by other similar datasets due to the synsets of images being interlinked by several types of relations, for example ImageNet contains images of 157 dog categories, this leads up to denser classification trees.
- Accuracy: ImageNet images are quality-controlled and human-annotated, the precision achieved in the whole dataset on average is 99.7%, although achieving high precision in all the depths of the datasets is challenging due to the lower in the hierarchy the synset is the harder to classify.
- Diversity: one of ImageNet goals is to provide different appearances, positions, viewpoints, poses and background clutter to objects included in the images.

ImageNet dataset consists of 14197122 images of 256x256 pixels organized in 27 high-level categories such as “animals”, “appliance”, etc. that divide into 21841 subcategories representing this synsets. Of all the dataset images, only 1034908 of them have been annotated with bounding boxes including the classified object coordinates. This means that the annotations of ImageNet fall into two categories:

- Image-level annotation that indicates if there is or is not a specific object class contained in the image.
- Object-level annotation of a bounding box and class label surrounding an object included in the image.

Figure 2.2 displays a sample of the ImageNet images that are annotated with bounding boxes, including the coordinates.



Figure 2.2: ImageNet dataset sample including bounding boxes (extracted from [1])

### 2.2.2 Open Image

In this section there are explained some of the most important characteristics of the Open Image dataset [35]. Open Image dataset contains around 9 million images divided into a set of classes derived from JFT [36], an internal dataset at Google. For Open Image, 19764 classes from JFT are selected which serve as image-level classes in the Open Images Dataset:

- Coarse-grained object classes (e.g. animal)
- Fine-grained object classes (e.g. Pembroke welsh corgi)
- Scene classes (e.g. sunset)
- Events (e.g. birthday)
- Materials and attributes (e.g. leather and red)

Out of the 19764 classes acquired from JFT, 600 object classes were chosen to need extra annotation information, bounding boxes, this means that around two million of the images in the Open Image dataset contain bounding boxes, this gives as a result a total of 15.4 million bounding boxes. Compared to the next larger datasets such as COCO and ImageNet, Open Image dataset provides 15 times more bounding boxes which demonstrates the complexity of the contained images as well as the quality of the performed annotation.

Figure 2.3 shows an example of the labeling process provided by OpenImage dataset, in which it can be seen that multiple bounding boxes belonging to multiple classes are included in the same image.

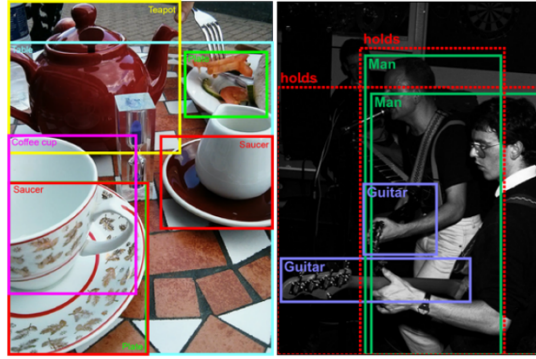


Figure 2.3: Open Image dataset sample that includes bounding boxes (extracted from [2])

### 2.2.3 COCO

Common Objects in Context (COCO) dataset [37] consists of 328 thousand images, so its image quantity size is small compared to ImageNet and Open Image, but it also contains additional features in the annotations:

1. **Object segmentation information** instead of just bonding boxes, performs a mask with 80 object categories. An example can be shown in figure 2.4.



Figure 2.4: COCO segmentation process sample (extracted from [3])

2. **Textual captions** per image describing the images for example “a plane flying through a cloudy blue sky”.
3. **Keypoint detection** task detecting people and localizing their keypoints (such as left eye, nose, right ankle). It is noteworthy that COCO dataset contains more than 20000 images and 250000 person instances labeled with its corresponding keypoints. Two examples of people images, including labeled keypoints are shown in figure 2.5.



Figure 2.5: COCO keypoint detection process sample (extracted from [3])



4. **Panoptic segmentation** consisting of background scene and object segmentation performing a segmentation on the entire image rather than just the dominant objects, action performed with the help of 80 thing categories (such as person, dog, bicycle) and 91 stuff categories (such as grass, sky road). Figure 2.6 displays a visual representation of this segmentation.



Figure 2.6: COCO panoptic segmentation process sample (extracted from [3])

5. **Dense Pose** that simultaneously detects people and localize their dense keypoints (each labeled person annotation includes an instance id and a mapping between all the pixels that belong to that person body). Figure 2.7 displays this mapping.



Figure 2.7: COCO dense pose process sample (extracted from [3])

### 2.2.4 Summary

Below, table 2.1 summarizes the main characteristics of the three datasets for person detection described previously in this section.

Dataset	Number of images	Number of classes	Data type
ImageNet	14.1M	21841	RGB
Open Image	9.2M	19754	RGB
COCO	328k	80	RGB

Table 2.1: Main characteristics of the analyzed datasets for person detection

## 2.3 Datasets for action recognition

In computer vision works that include person and action recognition, it's usual to evaluate those techniques employing datasets of images or videos to compare the performance of different approaches. There exist multiple datasets widely used by the scientific community for action recognition. The most significant ones are described in this section: UCF-101, NTU-RGB+D and Kinetics.

### 2.3.1 UCF-101

The UCF-101 dataset [38] consists of 101 action pre-defined classes, including a total of 13320 videos of 25 fps and a mean length of 7.21 seconds, forming a total of 1600 minutes of video data. These actions can be divided in five types:

1. **Human object interaction**, such as applying eye makeup, cutting in kitchen, shaving beard, knitting among others.
2. **Body-Motion Only**, some examples are blowing candles, pushups, trampoline jumping, swing, among others.
3. **Human-Human Interaction**, as military parade, haircut, head massage, band marching among others.
4. **Playing Musical Instruments**, some examples are playing flute, playing guitar, playing violin, playing piano among others.
5. **Sports**, some examples are surfing, horse riding, diving, bowling among others.

Figure 2.8 shows some sample images from UCF-101 dataset.



Figure 2.8: Sample images from UCF-101 dataset (extracted from [4])

It is worth highlighting that the labeling information only includes the performed action, not the location of the person performing that action.

The number of clips per action class and the distribution of the clips according to duration is illustrated in the figure 2.9.

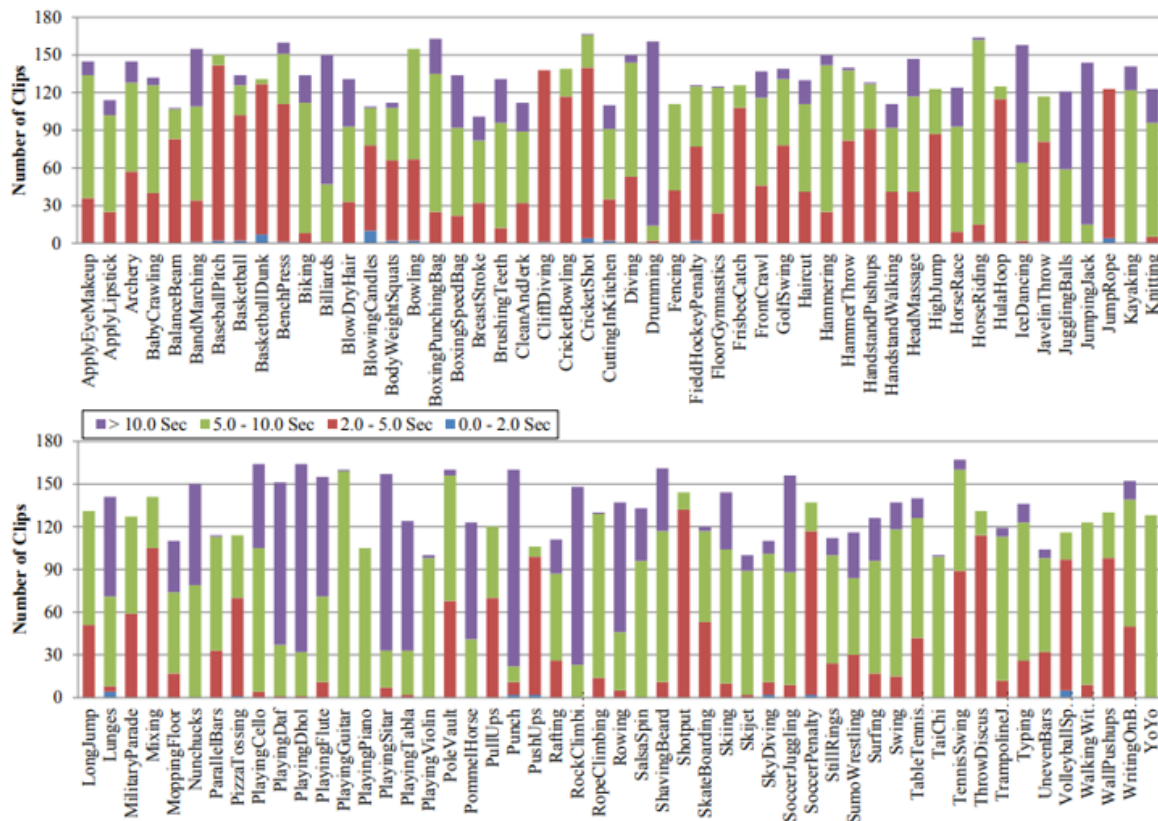


Figure 2.9: Distribution of video in actions along the UCF-101 dataset (extracted from [4])

### 2.3.2 NTU RGB+D

In this section, there are explained the main characteristics of the NTU RGB+D dataset [5].

This dataset contains 60 different action classes, performed by 40 subjects, with a total of 56880 video samples in highly variant camera settings using multiple Kinetic cameras at the same time. The characteristic of this dataset is that it uses the physical structure of the human body to improve the performance of the learning framework, since it includes Red, green and blue (RGB), Depth and skeleton data for each action.

The labeling information of this dataset includes both: the action performed and the location of the person performing it. Figure 2.10 shows in the first row the different camera views used to collect the data, and second row displays the different types of data that the cameras provide.



Figure 2.10: NTU RGB+D dataset sample images.(extracted from [5])

### 2.3.3 Kinetics (Kinetics Human Action Video Dataset)

The Kinetics dataset [39] contains a total of 400 human action classes with at least 400-1150 videos of each action with a mean duration of 10 seconds taken from different youtube videos. The labeling information only includes the action performed, not the location of the person performing the action. The actions are distributed in different action classes such as:

1. Person Actions, some examples are drawing, drinking, laughing among others.
2. Person-Person Actions, some examples are hugging, kissing, shaking hands among others.
3. Person-Object actions, some examples are opening presents, mowing lawn, washing dishes among others.

The whole dataset is composed by a total of 306245 videos, that are divided into three splits inside each one of the 400 actions, 250-1000 for training, 50 for validation and 100 for testing per class. Some Visual examples of the dataset are shown in figure 2.11.

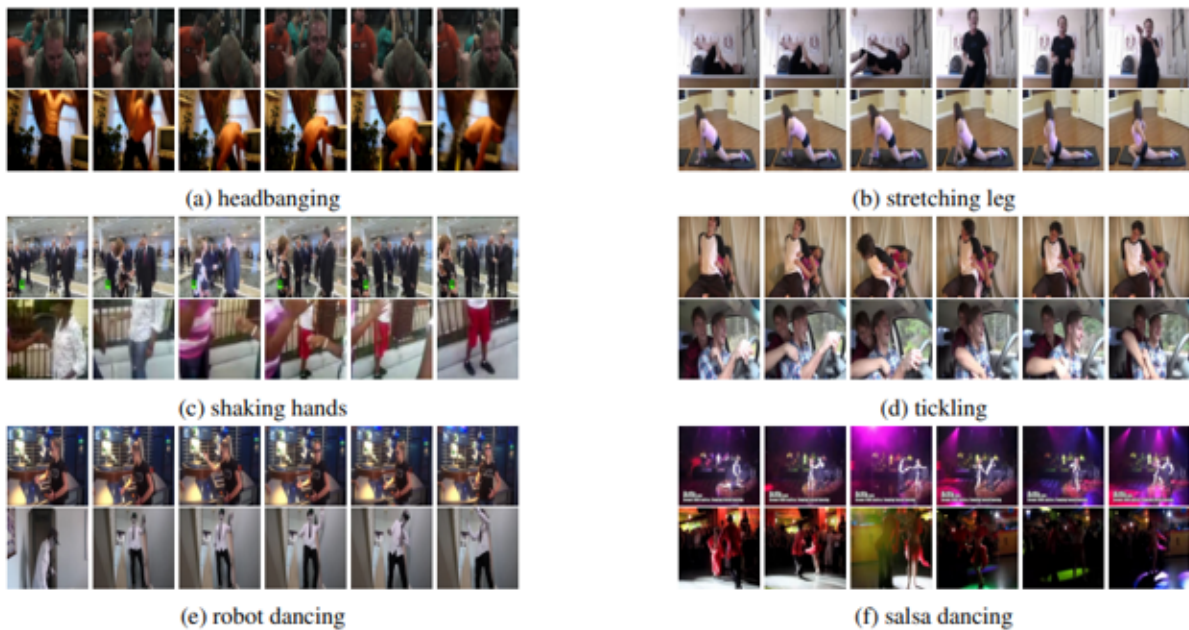


Figure 2.11: Kinetics dataset sample images (extracted from [6]).

### 2.3.4 Summary

Below, table 2.2 summarizes the main characteristics of the three datasets for action recognition described previously in this section.

Dataset	Number of actions	Number of sequences	Data type
UCF-101	101	13320	RGB
NTU RGB+D	60	56880	RGB+D
Kinetics	400	306245	RGB

Table 2.2: Main characteristics of the analyzed datasets for action recognition

## 2.4 Group action datasets

Group actions are one of the most important actions to label in the PALAEMON project as one of its main goals is to detect when a stampede or a blockade is happening. Due to that, below, they are described some widely used datasets containing group actions labeling.

### 2.4.1 Collective activity dataset

The Collective activity dataset [40] [41] contains 5 different group actions: *crossing*, *walking*, *waiting*, *talking* and *queuing* included in a total of 44 short videos. Figure 2.12 shows a sample of each one of the group actions contained in the dataset.



Figure 2.12: Collective activity dataset sample images (extracted from [7])

This dataset annotation includes the frame number, the coordinates of the person bounding box and its class and pose IDs as shown in figure 2.13. The Class ID is a number between 1 and 5 that gives information about the action performed and the Pose ID is a number between 1 and 8 related to the way the person being labeled is looking at (such as right, front right, front, back, back-right etc). However, this dataset does not include any of the actions of interest for the PALAEMON project.

Frame number	X	Y	Width	Height	Class ID	Pose ID
--------------	---	---	-------	--------	----------	---------

Figure 2.13: Collective activity dataset labeling distribution

### 2.4.2 Atomic Group actions

Atomic group actions dataset [42] [43] is a set of 200 videos focused on the group-group actions of formation, dispersal, and movement, and group-person actions of person joining and person leaving a group. It is equally divided in sets of 40 videos in each of these 5 actions included. All the dataset videos are cropped to  $640 \times 480$  pixels, and have a frame rate of 30 fps of a mean length of 5 seconds, being the

shortest at 1 second and the longest at 15. The annotation of this dataset includes individual and group position with bounding boxes and their trajectory. Figure 2.14 shows some samples of the Atomic group actions dataset, this samples include the trayjectory performed by the persons included (not included in the dataset).



Figure 2.14: Atomic group actions dataset sample (extracted from [8])

### 2.4.3 UMN

The UMN dataset [9] is collected from the University of Minnesota. This dataset contains videos located in 11 different scenarios in which a stampede event is taking place. The scenarios include feature indoor and outdoor locations and the average number of persons shown per video are around 20 which lead up to crowded scenarios in which the videos start with normal behaviors and end with abnormal ones.

Figure 2.15 shows three sample images from the UMN dataset.



Figure 2.15: UMN group action dataset sample images (extracted from [9])

### 2.4.4 BEHAVE

The Behave dataset [9] contains 321 clips of different events including crowded scenarios and group activities such as, meeting, splitting up, standing, walking together, ignoring each other, fighting, escaping and running, this actions can be divided into normal and abnormal events.

Figure 2.16 shows some examples of the BEHAVE dataset, it displays normal behaviors in the upper row and abnormal behaviors in the lower one.



Figure 2.16: BEHAVE group action dataset sample (extracted from [9])

## 2.5 Artificial Neural Networks

An **ANN** can be defined as a Multiple Input Multiple Output (MIMO) system that operates using a very large number of parallel and series connected arithmetic units called neurons. These neurons that form the neural network are usually arranged in layers, and their functionality depends directly on the organization structure and functionality that is specified by the type of layer they are included in. This means that an **ANN** can change its functionality depending on the type of layers included as the operations the layers perform globally depend on the type of it. As it can be seen **ANNs** can be implemented in many different scenarios due to their flexibility. One important thing about them is that their performance, in the case of image processing, is highly improved when they are dealing with non-linear dependence between inputs and outputs.

These artificial neurons are supposed to mimic the action of a biological neuron that accepts many different input signals from other neurons and processes them in a pre-defined way. They are composed of two different parts, “net input” (linear part) and “transfer function” (non-linear part, also called activation function).

The first part is a linear combination of the input variables which is multiplied by the "weights". These, "weights" are coefficients that may the neuron vary by itself, with this, the results (outputs) of an **ANN** are constantly feed-back as new inputs and these weights are modified in order to acquire the desired result, normally represented as a threshold, also known as back-propagation, this means that the "weights" are constantly being reajusted by the neurons in order to achieve the desired result, this modification of the neuron weights is only applied during the training process of the **ANN**.

The second part serves as a transfer function, since it transfers the signal through the neuron to another neuron input and also, as said before, it can introduce non-linearities which helps the Neural Network to learn better in each epoch (number of iterations training the **ANN**), eventhough a linear

activation function can be useful in certain cases. During this section of the chapter there is an in depth study of the most used activation functions nowadays.

Figure 2.17 displays the internal architecture of an artificial neuron explained before. The inputs corresponds to the input signals received directly from the input or from other neurons. This input signals are then multiplied by the weights, added and a threshold is applied to them, finally it passes through its transfer function to the neuron output.

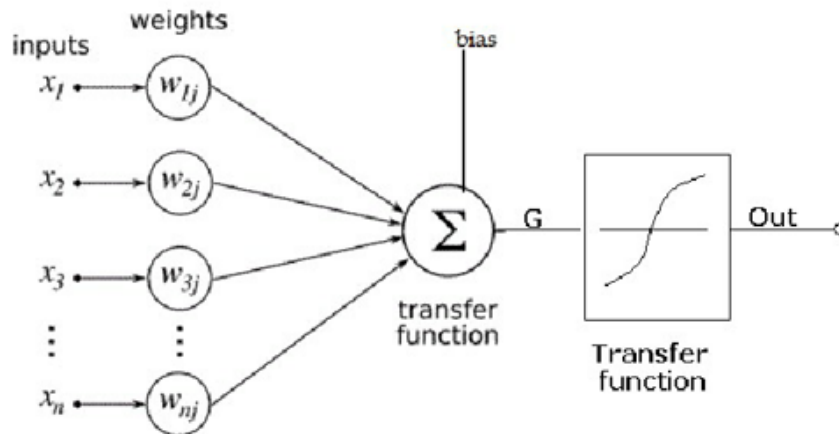


Figure 2.17: Neuron linear and non-linear parts (extracted from [10])

### 2.5.1 Activation functions

As it can be seen in figure 2.17, the internal structure of an artificial neuron includes a transfer function or activation function. Below, there are described are displayed the most common ones, including its characteristics [44] [45]:

- **Binary step function:** this function thresholds the input values to 1 and 0 depending on if the value is above or below 0 respectively. This activation function is used in binary classification scenarios, but it can not perform on multi-class detection problems. Equation 2.1 and figure 2.18 displays the equation and the representation of the binary step activation function respectively.

$$a_{ij} - f(z_{ij}) = \begin{cases} 0 & \text{if } z_{ij} < 0 \\ 1 & \text{if } z_{ij} > 0 \end{cases} \quad (2.1)$$



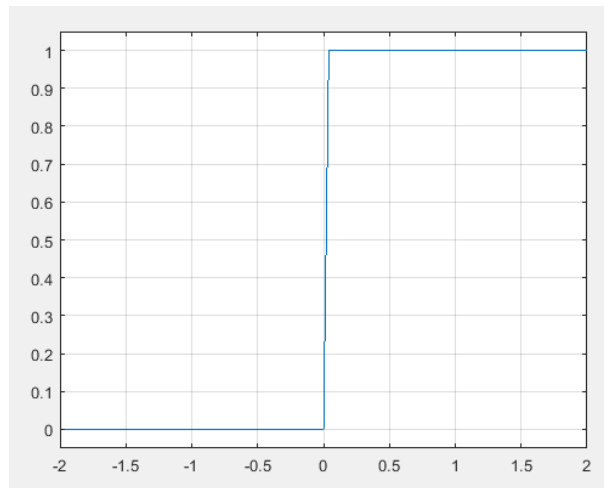


Figure 2.18: Binary step function representation

- **Linear activation function:** a linear activation function provides the sum of the weighted inputs of the node. As no previous operations are performed, the linear activation function can give at its output any numeric value, this makes this activation function unable to predict probabilities. Equations 2.2 and 2.3 displays some of the equations used in linear activation functions, with their respective representation shown in the figures 2.19 and 2.20 respectively.

$$R(z, m) = z * m \quad (2.2)$$

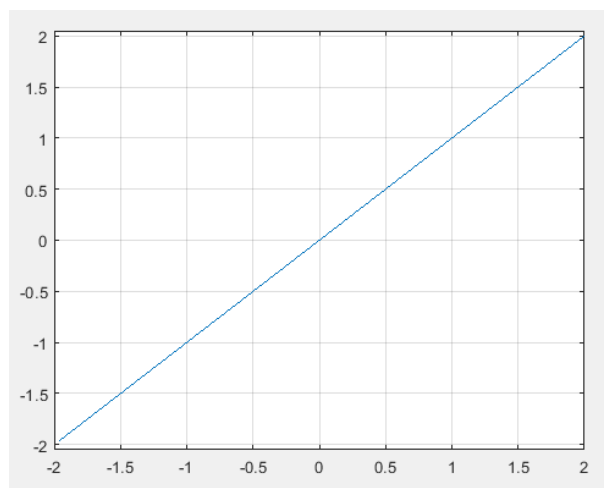


Figure 2.19: Some linear step functions representations

$$R'(z, m) = m \quad (2.3)$$

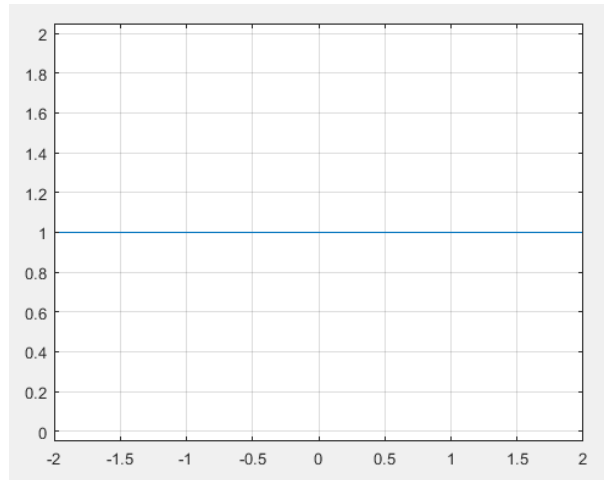


Figure 2.20: Some linear step functions representations

- **Sigmoid activation function:** this function maps the entire number line to a small range normally between 0 and 1 or -1 and 1. This means that the sigmoid function can convert any real value into one that can be understood as a probability. In machine learning sigmoid functions can be placed in the last layer of the model to convert its output into a probability score, since this eases the understanding of the model output. They are also used in two-class classification as they are an important part of a logistic regression model that, again, turns any real value into a probability. However, sigmoid activation function is only ideal in the case of a binary classification problem where the output can be understood as a binomial probability distribution. Taking this into account, the sigmoid activation function is not appropriate for multi-class classification problems. Equation 2.4 displays the corresponding equation for the sigmoid function represented in figure 2.21.

$$\phi(s_k) = \frac{1}{1 + e^{-s_k}} = \frac{e^{s_k}}{1 + e^{s_k}} \quad (2.4)$$

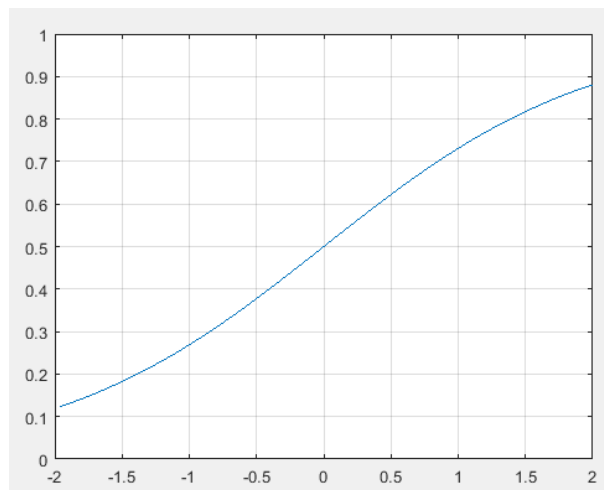


Figure 2.21: Sigmoid activation function representation

- **Softmax activation function:** it is a mathematical function that converts a vector of real numbers into a vector of probabilities in a proportional way. This activation function is used in machine learning when having a multi-class classification problem, as the outputs of the model are converted

from weighted sum values into probabilities that sum to one, this means that each value in the output of the softmax function is interpreted as the probability that the input belongs to that class. Equation 2.5 displays the equation of the Softmax activation function.

$$\sigma(\vec{z}_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.5)$$

- **Rectified Linear Unit (ReLU) activation function:** this is the most used activation function in neural networks as no heavy computation is required due to the function returning 0 if it receives a negative input and returning the input for any positive input value, which gives an output range from 0 to infinity. As ReLU does not require heavy computation a model that includes this function takes less time to train or run. The fact that any negative input value returns 0 results in better predictive power and less overfitting noise. This means that neuron will be processing meaningful aspects of the classification problem. However, this leads to certain neurons of the network to not activate at all and will be stuck on the negative side of the function permanently, what leads to a problem called dying ReLU because once a neuron gets negative its unlikely for it to recover, this neurons are not playing any role and are essentially useless which eventually leads up to a large part of the network doing nothing. This problem is likely to occur when the learning rate is too high. Equation 2.6 and figure 2.22 displays the equation and the representation of the ReLU activation function.

$$y = \max(0, x) \quad (2.6)$$

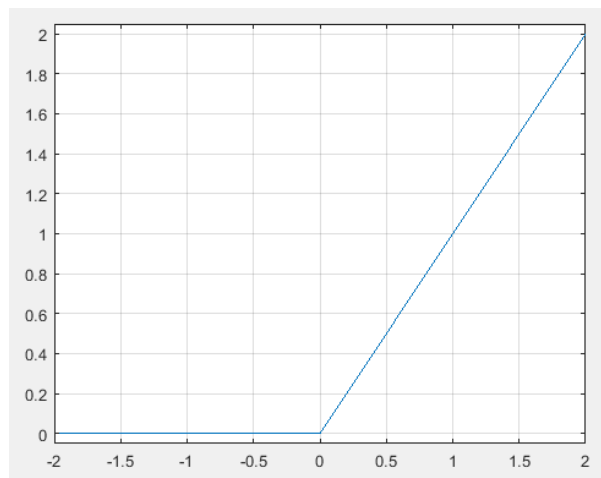


Figure 2.22: ReLU activation function representation

- **Leaky ReLU activation function:** this function is an improved version of ReLU activation function as instead of 0 for any negative input values a small linear component is defined. This linear component returns a very small value of its input and in conclusion in that region there will no longer be dead neurons, this means that the dying neuron problem of the ReLU function is solved using Leaky ReLU instead. Equation 2.7 and figure 2.23 displays the equation and the representation of the leaky ReLU activation function.

$$f(x) = \begin{cases} 0.01x, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \quad (2.7)$$

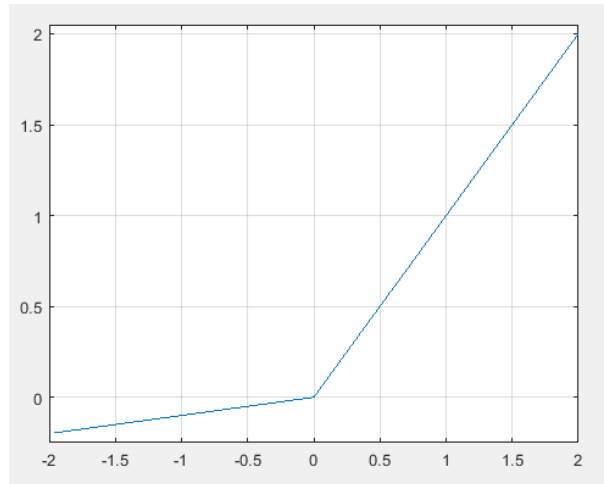


Figure 2.23: Leaky ReLU activation function representation

### 2.5.2 ANN structure

To understand ANN better there are presented the different parts of which a neural network is composed of, as seen below it is composed of an input layer, hidden layers and the output layer.

The only “fixed” sizes when using ANNs for image classification and object recognition problems are the sizes of the input and output layer, the size of the input layer often corresponds to the size of the image you are using as input (i.e.  $320 \times 240$  neurons in the case of using an  $320 \times 240$  pixels image), and the output layer has the same number of neurons as the number of classes of interest in classifying the image with. This means that if there is presented a binary classification problem in which there is interest in classifying an image between two different classes (i.e. person and not a person) the output layer only needs two neurons.

The main difference between the neural networks are the hidden layers as, by changing the type of them, you can obtain different neural networks such as CNN in which the hidden layers perform the convolution operation, Recurrent Neural Network (RNN) which uses time series data, Modular Neural Network (MNN) whose neural networks composed of are linked by an intermediate, between others. Figure 2.24 displays a visual representation of the input-output relation of an ANN.

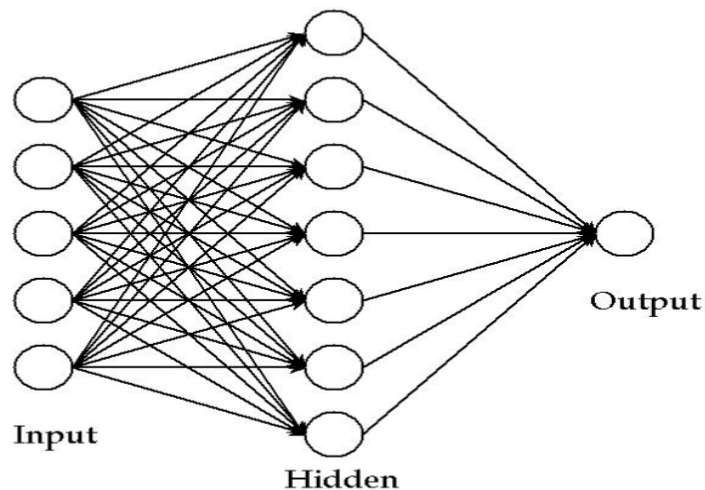


Figure 2.24: Input-output relation of an ANN (extracted from [11])

DNNs are ANNs with a large number of layers of neurons in the hidden layer section of its structure. This is, between the input and output layers of the DNN, this DNNs are mainly feedforward, this means that the data flows from input to output without going back, CNNs are one form of implementation of this process described. Currently CNNs are the most used method for processing visual data. As this TFG arises a visual data problem a more in depth explanation of CNNs is presented below.

## 2.6 Convolutional Neural Networks

Once the theoretical basis of neural networks have been presented, as mentioned in the previous section, an in depth explanation of CNNs takes place.

A CNN [12] is a neural network that contain layers that perform the convolution operation, for processing visual data problems it typically contains several types of layers which include convolutional layers, pooling layers, and activation layers in order to maximize its performance with this type of input data as this architecture allows a better image characteristic extraction and image preprocessing as it allows to reduce its size before feeding the information to the fully connected layers.

Figure 2.25 shows a general diagram of a CNN classifier including the different layers mentioned before as it can be seen that an image is used as input of the CNN.

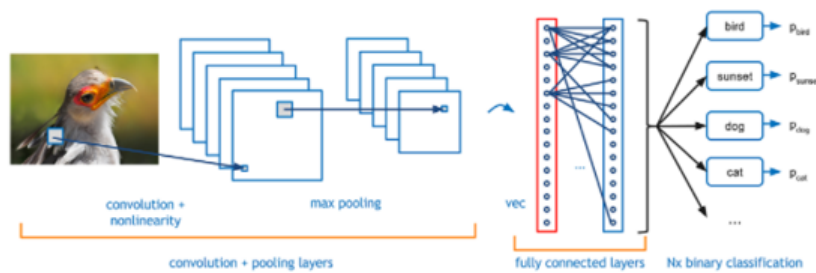


Figure 2.25: Diagram of the steps followed by a CNN classifier (extracted from [12])

The convolutional layer is the core of all this process and does most of the computational heavy lifting, it is in charge of performing a convolution operation which is explained up next. If figure 2.26, is represented as a  $10 \times 10$  matrix of values and a  $3 \times 3$  matrix is taken and slide it all around the image at each position making it multiply every element of the image by the values of our  $3 \times 3$  matrix this results in a single number that represents all the values in the window. This process allows characteristic extraction of the input image through a reduced sized version of it, product of this convolution operation. Figure 2.26 displays a graphical representation of the sliding window convolution operation.

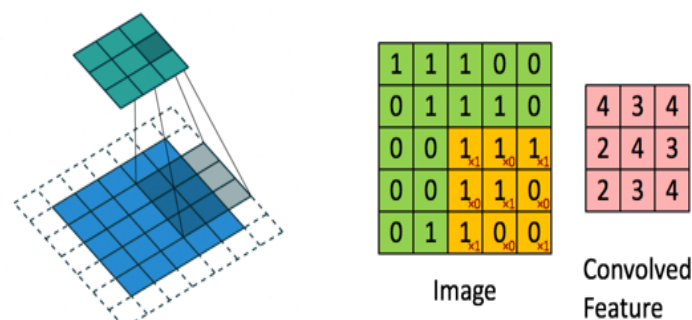


Figure 2.26: Convolution Operation (extracted from [12])

This  $3 \times 3$  matrix that moves at each point of the image is called kernel and it has random values in it, that is because its initial values are not important at all due to the model changing those values as the training process unfolds, similar to the backpropagation concept explained in previous sections, as explained before, once the training process has been finished the kernel weights are fixed.

It is important to highlighting that pooling layers can be preceded by one or more convolutional layers or not be pooling layers included in the CNN. In the presented example, after the convolutional layer there is a pooling layer which is in charge of combining the outputs of the convolutional layer. This pooling operation reduces the dimension of the information that has to be handled.

There are two main different types of pooling, max value pooling which uses the maximum value of each array and average pooling which uses the average value. Figure 2.27 shows a visual example of the two different pooling methods mentioned before and the outputs obtained from the two different processes.

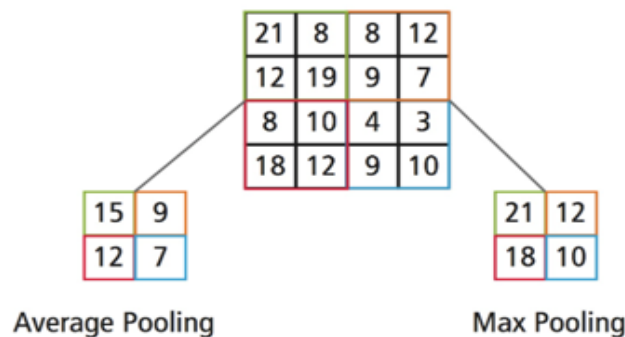


Figure 2.27: Pooling Types (extracted from [12])

Finally, in this example, an activation layer is used, which is very important because in the majority of the scenarios applying CNNs to visual data problems there is a need to introduce non-linear properties into the network, as helps the model to understand the complexity of the problem given and provide accurate results. It is similar to the different the transfer functions of the artificial neurons explained along this chapter, as different functions adapt better to certain visual data problems.

This activation layer calculates the weighted sum and decides whether to discard a particular neuron or not depending on if it is needed in order for the network to perform accordingly or not, there are several types of non-linear activation functions like: Sigmoid, ReLU, Leaky ReLU, etc, as seen before.

## 2.7 People and object detection with YOLO

There are numerous approaches in the literature for people and object detection based on CNNs, such as RetinaNet [46] which is composed of a backbone and two task-specific subnetworks that are in charge of convolutional object classification and convolutional bounding box regression respectively, Deconvolutional Single Shot Detector (DSSD) [47] which consists on feature extraction and detection, deconvolutional layers are used in the detection process in order to increase the resolution of the feature maps product of the feature extractor or Region-based Fully Convolutional Network (R-FCN) [48] which is a fully-convolutional region-based detector that applies a subnetwork per region hundreds of times this means that all the computation is shared on the entire image.

In this TFG, there is used YOLO [26] [49] [16] [14] for people detection, due to its good balance between speed and accuracy, what makes it widely used in different applications. This sections describes the main YOLO characteristics that justify choosing this network.

### 2.7.1 YOLO features

Figure 2.28 displays a visual representation about the process YOLO follows to make a detection and assign a bounding box to it. The system resizes the input image to  $448 \times 448$  (this may vary with the version of YOLO being used). Then, it runs a single convolutional network on the image and finally its thresholds the results based on the model confidence.

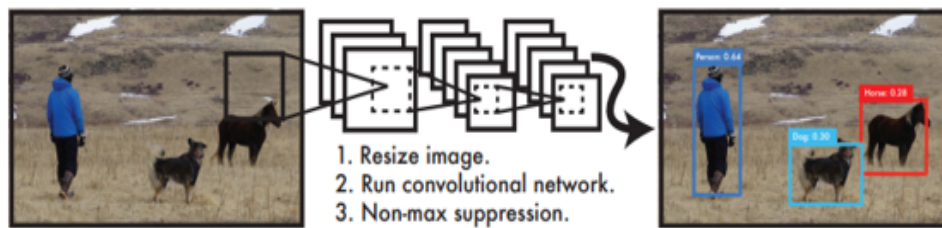


Figure 2.28: Basic diagram that contains the steps followed by YOLO to make a detection and assign a bounding box (extracted from [13])

This single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for each of those boxes. For being capable of this, YOLO trains on full images and directly optimizes detection performance.

YOLO system divides the input image into an  $S \times S$  grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object that falls into an specific category and how accurate it thinks the bounding box assigned to the prediction is. If there is no object in the cell this confidence score should be zero. Each bounding box consists of 5 values corresponding to:  $x$ ,  $y$ ,  $w$ ,  $h$ , and confidence. The  $(x, y)$  coordinates represent the center of the box relative to the bounds of the grid cell. The width and height ( $w, h$ ) are predicted relative to the whole image. Figure 2.29 displays a diagram that shows how the previously explained process is performed.

- YOLO is capable to run up to 45 frames per second which means that it can be used to process video in real-time with less than 25 milliseconds of delay.
- YOLO reasons globally about the image when making predictions. This means that it sees the entirety of the image and considers its features during training and testing to acquire contextual information about the classification classes included and how do they look like. This allows YOLO to predict all bounding boxes across all of the included classes simultaneously.
- YOLO learns to generalize representations of objects which provides YOLO with versatility, so it is less likely to perform incorrectly when applied to different scenarios or receiving unexpected inputs different of the ones being trained with.

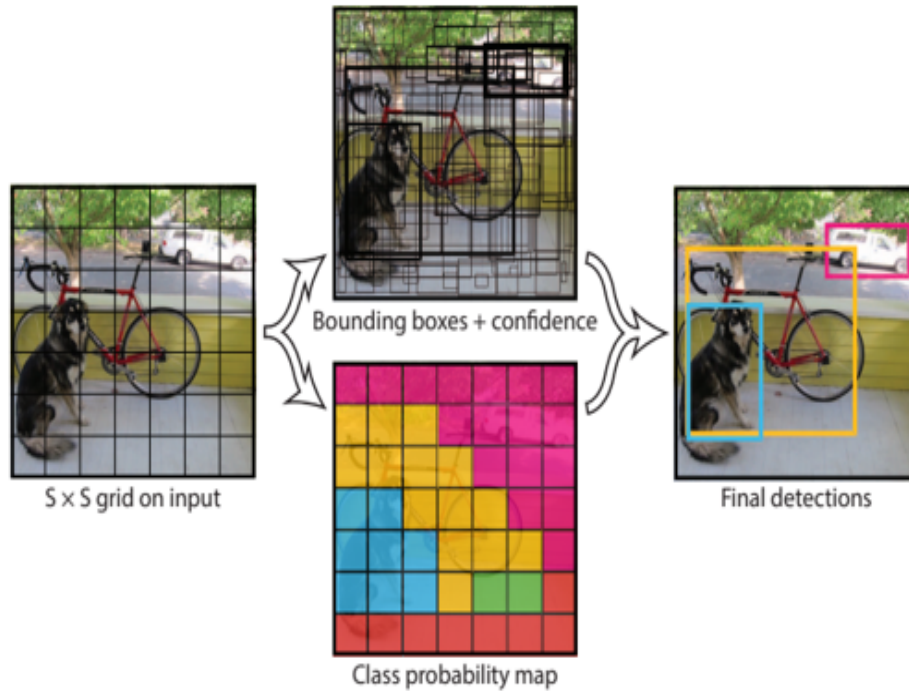


Figure 2.29: Graphical representation of the YOLO methodology used to perform detections (extracted from [13])

### 2.7.2 Network Design

YOLO is implemented as a CNN, the initial convolutional layers of the network extract feature from the image while the fully connected layers predict the output probabilities and coordinates. YOLO network has 24 convolutional layers followed by 2 fully connected layers. Figure 2.30 displays the architecture of YOLO network.

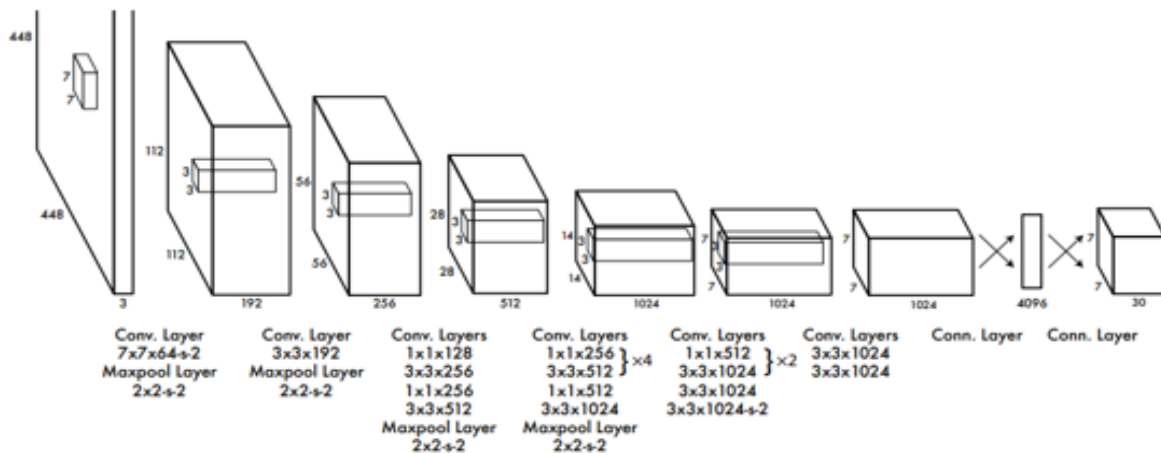


Figure 2.30: YOLO Network Design (extracted from [13])



### 2.7.3 Limitations

YOLO imposes strong spatial constraints on bounding box predictions as each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of objects that YOLO can predict per grid cell. YOLO struggles with small objects that appear in groups, such as small animals or very crowded spaces filled with people. YOLO makes a significant amount of localization errors and has low recall.

## 2.8 YOLOv3

After the development of YOLO there have been emerging improvements included in different versions of it. The main features and improvements included in YOLOv3 are shown in this section of the chapter.

As seen before the main problems of YOLO were localization errors and low recall, therefore some mechanisms were included in later versions of it in order to improve them while maintaining accuracy, the mechanisms included are listed below.

1. **Batch Normalization**, adding *Batch Normalization* to all the convolutional layers, it is a method used to make ANNs faster and more stable by normalization of the layers inputs by recentering and re-scaling, this ensures that in deep convolutional layers inputs are still normalized, applied to this case it resulted in a 2% improvement of the mean average precision compared to previous version that did not include *Batch Normalization*.
2. **Anchor boxes**, as previously seen YOLO was limited as it was only capable of one prediction per grid cell, as a solution *anchor boxes* were implemented, which are a variation in width and height of the corresponding grid cell that ease object detection as it can keep the best fitting anchor box and it is not limited by the original shape of the grid cell, this lead to multi-object prediction. Figure 2.31 displays the grid cell (red) and 5 anchor boxes (yellow) with different shapes.

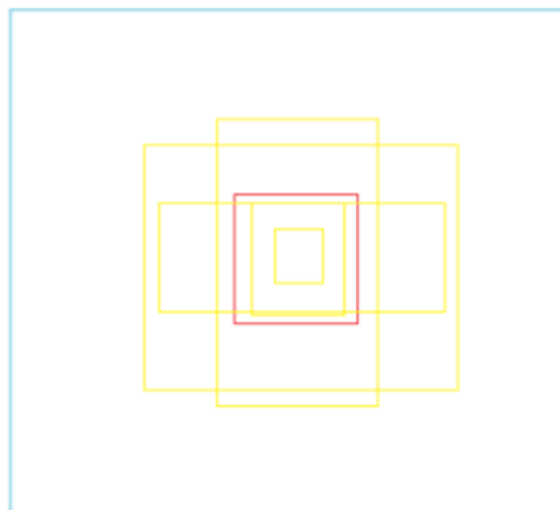


Figure 2.31: Anchor box visual example (extracted from [14])

3. **Speed improvements**, the speed improvements came from modifying the architecture of **YOLO** itself, this has been addressed by modifying the backbone, which is in charge of feature extraction. Initially **YOLO** contained 24 convolutional layer which were reduced to 19 in **YOLOv2** as it uses Darknet-19, for the version used in this **TFG**, **YOLOv3**, the number of convolutional layers were changed again to 53 as the backbone used is Darknet-53. Table 2.3 shows a comparison between this network and other backbone examples.

Backbone	Top-1	Top-5	BnOps	BFLOP/s	Frames per second (FPS)
Darknet-19	74.1	91.8	7.29	1246	171
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

Table 2.3: Backbone comparison (extracted from [13])

The first two columns refer to accuracy and the next three to speed, as it can be seen compared to Res-Net-152 it gives the same performance while giving less billion operations and more billion floating operations per second and **FPS**, this means that has an improvement in speed while maintaining accuracy compared to other backbones used.

4. **Precision of small objects**, table 2.4 shows different one-stage detectors with their respective backbone and the average precision score for small (APs), medium (APm) and large (APl) objects respectively.

One-stage method	Backbone	APs	APm	APl
YOLOv2	Darknet-19	5.0	22.4	35.5
SSD513	ResNet-101-SSD	10.2	34.5	49.8
DSSD513	ResNet-101-DSSD	13.0	35.4	51.1
RetinaNet	ResNet-101-FPN	21.8	42.7	50.2
RetinaNet	ResNeXt-101-FPN	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	18.3	35.4	41.9

Table 2.4: Average precision comparison between one stage detectors for different sized objects (extracted from [16])

As it can be seen, **YOLOv3** presents significant improvements regarding its previous version, **YOLOv2**, as it can be observed, the average precision for small objects has been highly increased from the previous version by 13.3, however, the average precision scores are less than RetinaNet but, as previously seen, it is still faster due to the backbone used. The improvement in detecting small objects is due to *short-cut connections* that concatenate the intermediate layers of **YOLOv3** backbone to the layer after the upsampling layer (in charge of increasing the sampling rate).

5. **Specifity of classes**, **YOLOv2** uses softmax activation function, previously seen in section 2.5.1, what leads to each bounding box only being assigned to one class, which in the case of complex dataset is sometimes not the case, on the other hand "**YOLOv3** uses a multi-label approach which uses independent logistic classifiers and binary cross-entropy loss for the class predictions during training" [16], this allows individual bounding boxes to be multiple and more specific which allows, as previously stated, to work with complex datasets that may contain overlapping labels.

Figure 2.32 shows the general architecture used by YOLOv3 layer by layer and can be compared to the one of YOLO shown in figure 2.30.

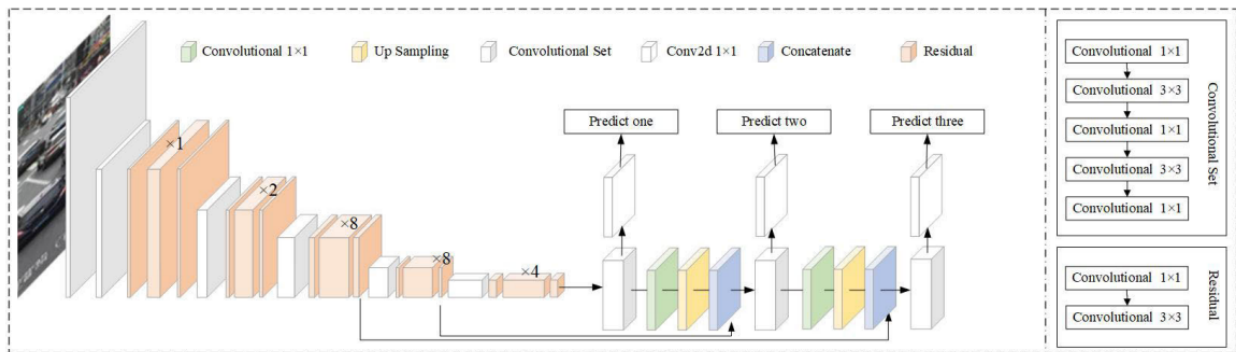


Figure 2.32: YOLOv3 general Architecture (extracted from [15])

## 2.9 Conclusions

YOLOv3 is fast and accurate, and it runs faster than other detection methods while maintaining comparable performance and it allows to trade-off between speed and accuracy by changing the model size and not retraining it. Figure 2.33 shows a graph that displays the performance difference between different detectors for COCO dataset.

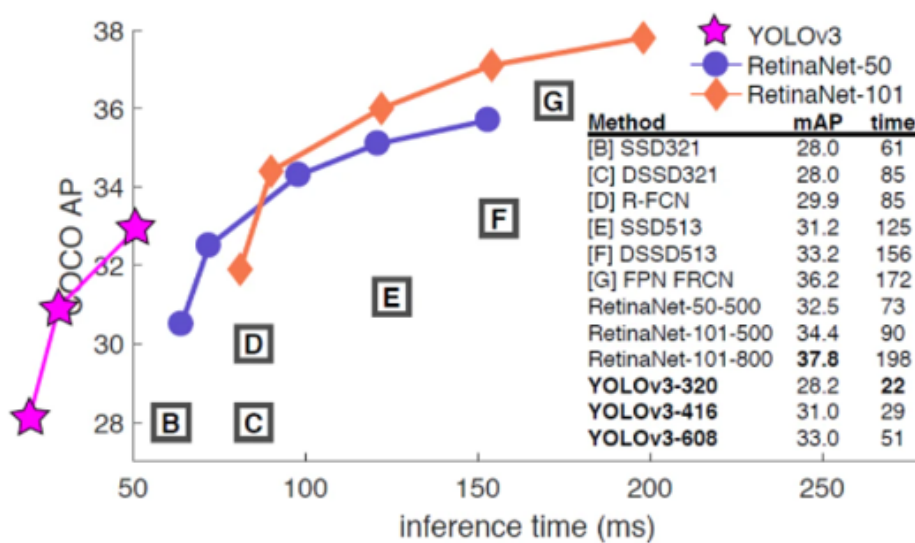


Figure 2.33: Comparison of different detectors using COCO dataset (extracted from [13])

As previously stated YOLOv3 increased its speed by changing the backbone to Darknet-53, this backbone is faster than other state-of-art one-stage detectors backbones while maintaining similar performance to them, there are also improvements that allows YOLOv3 to perform decently in complex datasets like multi-label approach and small object improvements with previous versions of it.



# Chapter 3

## Development

### 3.1 Introduction

The main objective of this [TFG](#) consists in labeling the [HARS](#) dataset which, as said before, consists of a set of videos that take place in certain parts of a cruise and includes a set people performing different actions.

As it has been stated in the introduction, this work is framed the *PALAEMON: A holistic passenger ship evacuation and rescue ecosystem* project (H2020-PALAEMON-814962) whose aim is to engage innovative technologies in a new intelligent ecosystem for mass evacuation of vessels. The labeling process focuses on the position of the persons in the different scenarios included in the dataset, individual actions (including walking, running, stationary, fall down, sit down and stairs), and group actions (stampede and blockade).

The main objective of labeling this dataset is to generate a set of ground truth files corresponding to each one of the videos for different applications that include the coordinates of the bounding boxes per frame and the actions performed by each one of the persons included as well as the group action performed by the people on the scenario. In the case of this [TFG](#) the generated ground truth is going to be used by [YOLO](#) to validate that, with the labeling process done, it is possible to evaluate the results as well as to study how [YOLO](#) performs in the dataset videos as they are very complex due to the scenarios conditions such as lighting, person distribution and the high level of occlusions between them.

It is important to highlight that the only part of the ground truth generated that is going to be used with [YOLO](#) are the bounding boxes, as [YOLO](#) is only capable of detecting objects but not actions even though the labeling process also includes individual and group actions.

Figure [3.1](#) shows a general flowchart of this [TFG](#), highlighting the sections that are detailed in this chapter.

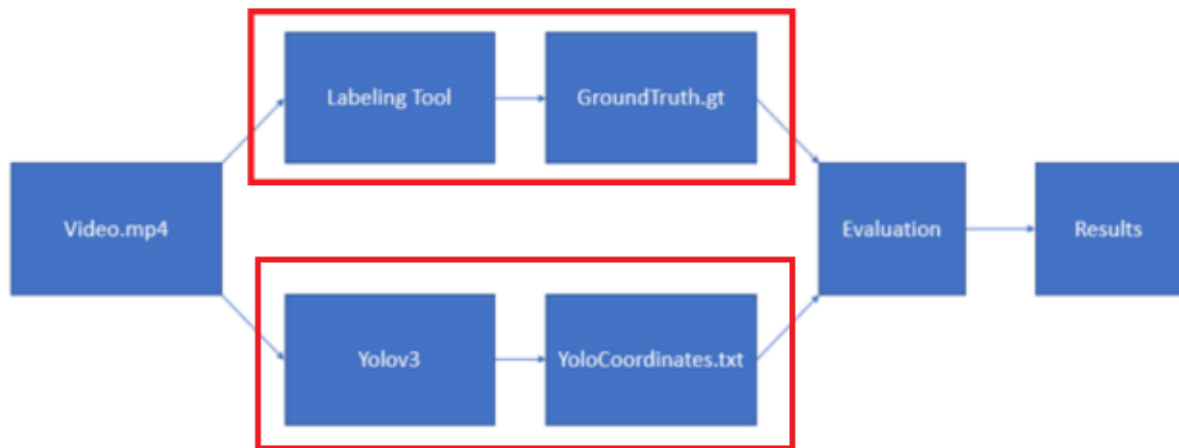


Figure 3.1: Flowchart of this TFG

This chapter distribution includes the following:

1. **Dataset analysis:** a detailed explanation of the [HARS](#) dataset is presented, this includes the recording setup used and the main characteristics of it.
2. **Labeling tool:** the labeling tool used for the labeling process usage is explained as well as the modifications made to its code and interface to adapt it to the [HARS](#) dataset labeling needs.
3. **Labeling process:** the criteria used for the labeling process is presented as well as a brief showcase of how it is performed with the labeling tool.
4. **YOLO implementation:** [YOLOv3](#) implementation in Google Colab and obtention of data outputs used in the evaluation are explained.
5. **Processing of the data obtained:** the files obtained from the labeling and implementation of [YOLO](#) processes are transformed in order to ease its use in the *Experimental results* section.

## 3.2 HARS dataset

In this section the setup used for the recording, the persons and actions that includes and the main characteristics of the [HARS](#) dataset are presented.

### 3.2.1 Recording setup

The [HARS](#) dataset contains a total of 74 videos with an average length of 40 seconds each. Those videos are equally divided into [RGB](#) and Infrared (IR) as the used camera provides [RGB](#) and [IR](#) recordings which have been taken in different scenarios included in the cruise ship.

The camera used in the recording of the dataset is the Intel Real Sense D435 [50]. This camera contains a low-power vision processor for real-time depth sensing. An [IR](#) projector and [RGB](#) module are also integrated, which allows the camera to output the same video recorded in the two different formats in which the dataset is acquired: [IR](#) and [RGB](#). However, each of this modules has its own characteristics,

image 3.2 and table 3.1 show an image of the used camera, its main components and the specifications of the integrated modules respectively.

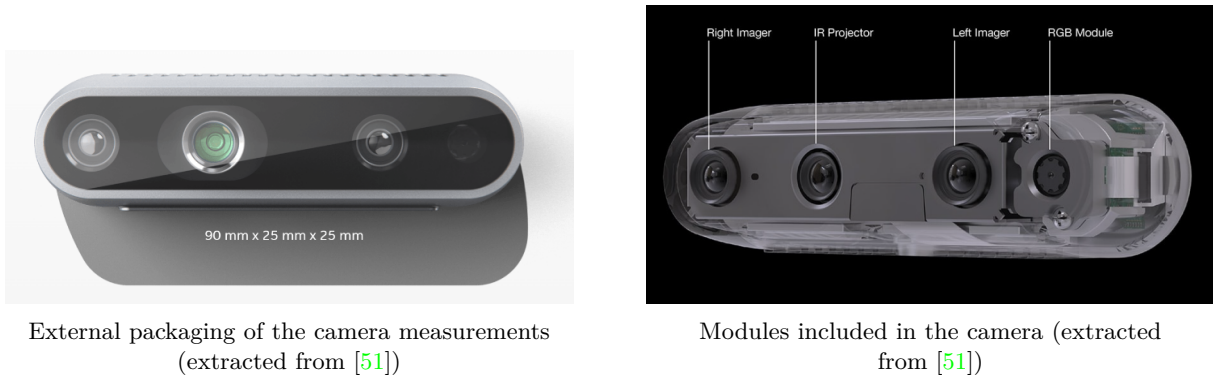


Figure 3.2: Relevant information about the camera used for the recording process

Module	Resolution	Frame rate	Field of View (FOV) (HxV)
IR	1920x1080	90 fps	87°x58°
RGB	1280x720	30 fps	69°x42°

Table 3.1: Specifications of the modules included in the Intel D435

The scenarios in which the recordings of the dataset have taken place are shown along with a brief description of them, representing the output of the camera for each of the modules used.

1. **Common area:** consists of a long corridor with mirrors at its sides that has three main exits (sections of the scenario in from which persons can enter and leave the scene): the beginning and the end of the corridor and the stairs located at the beginning of it. Figure 3.3 shows the scenario. It is worth noting that all the sample images shown in this section includes both RGB and IR images from each scenario.

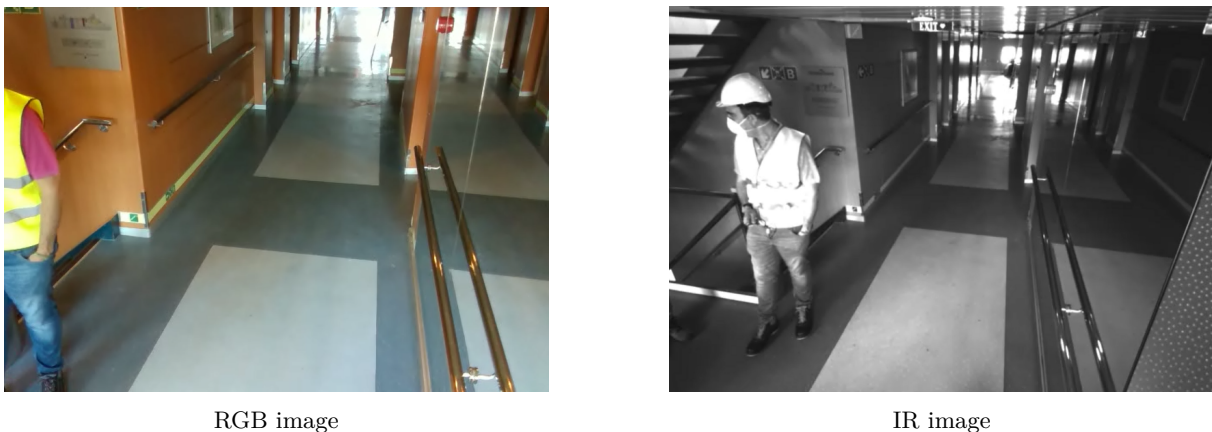


Figure 3.3: Common area scenario shown from RGB (left) and IR (right) camera perspective

2. **Cabin corridor:** consists of a long corridor that has two main exits, the beginning and the end of the corridor. It is important to highlight that the middle section of the corridor has different lighting that the rest of it, as it can be seen in Figure 3.4, that shows the scenario.



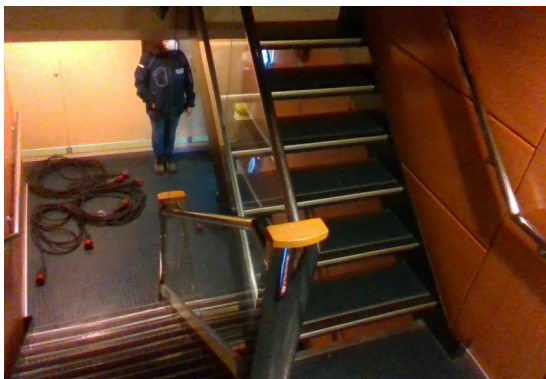
RGB image



IR image

Figure 3.4: Cabins corridor scenario shown from RGB (left) and IR (right) camera perspective

3. **Exit Stairs:** consists in a set of two stairs. The viewpoint of the camera is located at the middle of both and it has two main exits: a door located in the lower floor and the end of the stairs located in the upper floor. Figure 3.5 shows the scenario.



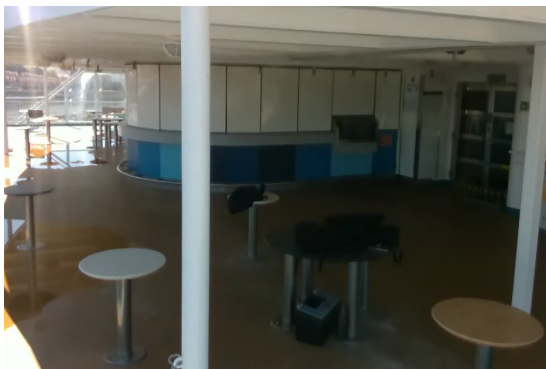
RGB image



IR image

Figure 3.5: Exit stairs scenario shown from RGB (left) and IR (right) camera perspective

4. **Exterior:** consists in an open area located in the surface of the cruise ship. It contains some tables and columns and it has three main exits, the door, the left side of the scenario and the right side. Figure 3.6 shows a picture of this scenario.



RGB image



IR image

Figure 3.6: Exterior scenario shown from RGB (left) and IR (right) camera perspective



5. **Lifeboat** consists of a section of the cruise ship that contains the lifeboats. This scenario has two exits: behind the camera and at the end of the hallway. Figure 3.7 shows the scenario.



RGB image



IR image

Figure 3.7: Lifeboat scenario shown from RGB (left) and IR (right) camera perspective

6. **Stair Corridors** consists of a staircase, the camera point of view allows to see the lower floor, this scenario has two exits, the lower floor and behind the camera. Figure 3.8 shows the scenario.



RGB image



IR image

Figure 3.8: Stair corridors scenario shown from RGB (left) and IR (right) camera perspective

### 3.2.2 HARS dataset characteristics

The dataset is balanced taking into account that it contains the same number of videos recorded in **RGB** and in **IR**, (as the camera provides two outputs of the same video) and the camera view point of both is the same, with the exception than the **IR** camera has a larger **FOV** than the **RGB** one. This causes that the video obtained from the **IR** camera contains more information as it is capable of providing a wider view from x and y axis. Figure 3.9 display the same frame from the same video in order to have a visual representation of the **FOV** characteristic explained before.



Figure 3.9: Exterior scenario shown from RGB (left) and IR (right) camera perspective

The actions performed along the dataset can be included in two groups:

1. **Individual actions** performed by each of the individuals independently from each other that include: walking, running, sitting, falling down and going through stairs.
2. **Group actions** performed by all the individuals or the majority of them in the video, including: stampede and blockade.

All of the actions included as well with the criteria to consider if an action belongs to a certain group are explained along this chapter.

Most of the videos included in the dataset contain three to four persons performing different actions in the scenario at the same time. This group of four people includes two men and two women with different heights. This selection of persons try to cover the diversity of the people that can be found inside a cruise ship in a real-world scenario.

It is important to highlight that the dataset is not balanced as different scenarios contain a different number of videos, different number of persons performing actions and different number of frames per scenario. Table 3.2 includes a summary of the distribution of videos of the dataset by scenario with the number of persons and the number of frames.

Scenario	Number of videos	Average number of persons	Number of frames
Common Area	10	4	8628x2
Corridor Cabins	5	4	6874x2
Exit Stairs	5	6	2739x2
Exterior	6	4	7413x2
Lifeboat	5	4	2336x2
Stair Corridors	6	4	5854x2

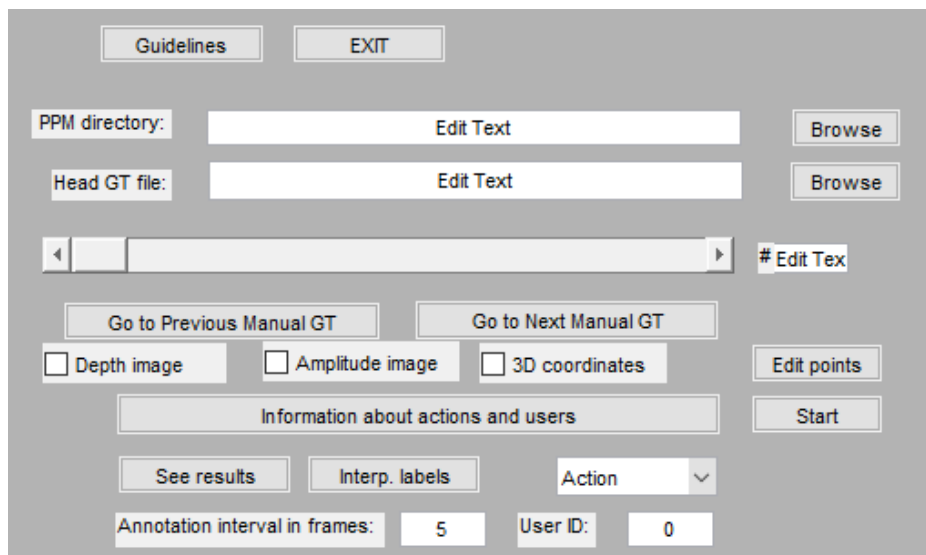
Table 3.2: Video, persons and frames distribution of the HARS dataset

## 3.3 Dataset labeling

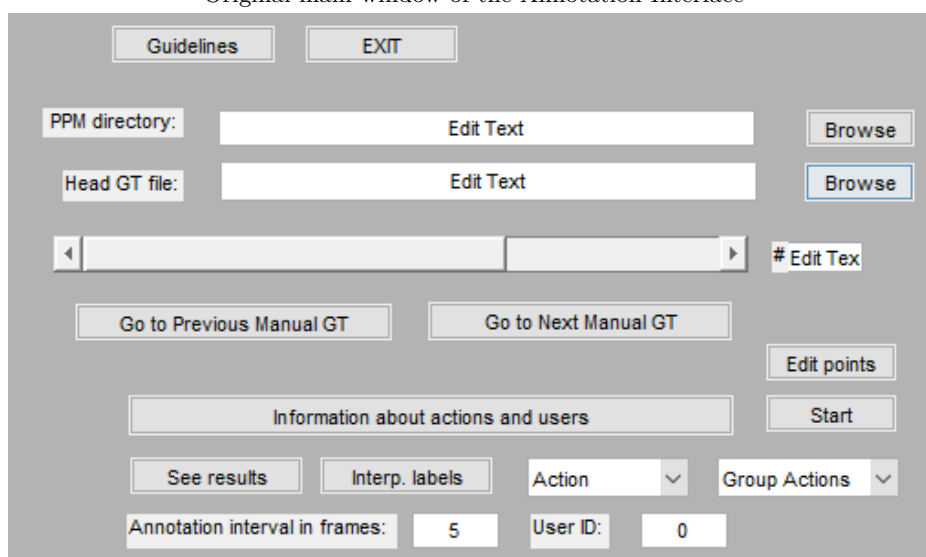
### 3.3.1 Labeling tool

For labeling the dataset, there is used a labeling tool developed in the [GEINTRA](#) research group, that has been adapted to the [HARS](#) dataset. This section briefly describes the main characteristics of the labeling tool.

The available labeling tool has been developed in MATLAB and has a Graphical User Interface (GUI) to ease its use. The application can be started by running the code included in *annotationinterface.m* in MATLAB, what opens two different windows: one in charge of displaying the frames (images that compose the video) and the other one for the setup of the labeling procedure. As the labeling tool has been adapted to the [HARS](#) dataset a display of the original interface and the one resulting of the modifications made are shown in figure 3.10.



Original main window of the Annotation Interface



Main window of the Annotation Interface

Figure 3.10: Main windows of the annotation interface window before and after modifications made to adapt the tool to the HARS dataset

Once the modifications of the labeling interface had been presented other modifications to continue to adapt the labeling tool to the [HARS](#) dataset are presented:

1. **Interpolation of labels problems**, the linear regression solution was originally thought for uses in which there was a continuous labeling without being stopped more than 30 frames. As the average length of the dataset videos is 1200 and the complexity of the actions performed commonly include some of the persons disappearing for more than 30 frames, the labeling tool had problems using again interpolation of frames once a person came back from the image. This meant that interpolation of labels could not be performed again for the user that came back to the image. The applied solution consisted on modifyintg the interpolation procedure in order to increase the number of frames between available labels.
2. **Addition of actions of interest**, originally the actions that could be labeled by the tool did not cover all the actions performed in the dataset, therefore the individual action “stairs” and the group actions “stampede” and “blockade” were defined. Since in each frame there can exist both, individual and group actions, this change has meant a modification in the original interface of the labeling tool, adding an sliding window for group actions as well as functions definitions for it and establishing a relation between the user interaction with the window and the structure which contains the information picked up from the labeling tool interface and a new modification into the indexes used by the start labeling and interpolation buttons associated functions. Furthermore, it has been necessary to modify the structure of the ground truth files, as well as the functions used to read and write those files in order to add the group actions data.

The labeling tool allows to input a video and extract the frames of it as a .jpg file in a folder called *FRAMES*. These frames are the baseline of the labeling process and the labeling tool allows to create bounding boxes and define actions to each one of the bounding boxes included a group action per frame in the frames that are obtained from the original video. It also provides users information that include a number associated to each one of the persons that can be seen in the dataset videos and information about the color display of the bounding boxes depending on the action selected. Figure 3.11 displays this information.

```
COLOUR OF ANNOTATION BOX  
  
0 Walk: BLUE  
1 Run: RED  
2 Sit down: YELLOW  
3 Fall down: GREEN  
4 Stationary: PINK  
5 Stairs: BLACK
```

Figure 3.11: Annotation Interface information

To start labeling, it must be chosen the frame number, the user ID and the performed action. For example, figure 3.12 shows the labeling GUI in which the chosen frame is the 81, the user ID is 0 and the action is “Walk” and the group action is “Blockade”.

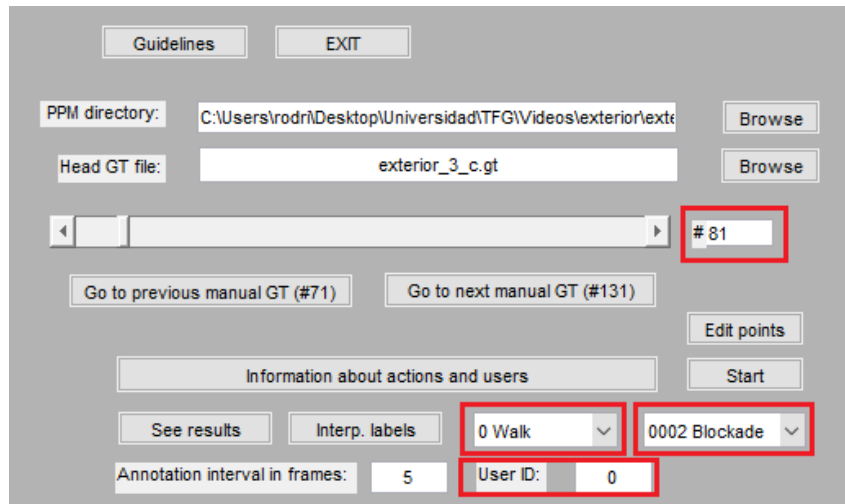


Figure 3.12: Use of the Annotation Interface

Each person is labeled by clicking with the left mouse button in the upper left and lower right corners of the corresponding bounding box. An example with two people is shown in figure 3.13, and as mentioned before the colour of the bounding box depends on the selected action, being the bounding box of user 0 pink as the labeled action is *stationary*, and the user 2 bounding box blue as the labeled action is *walking*.

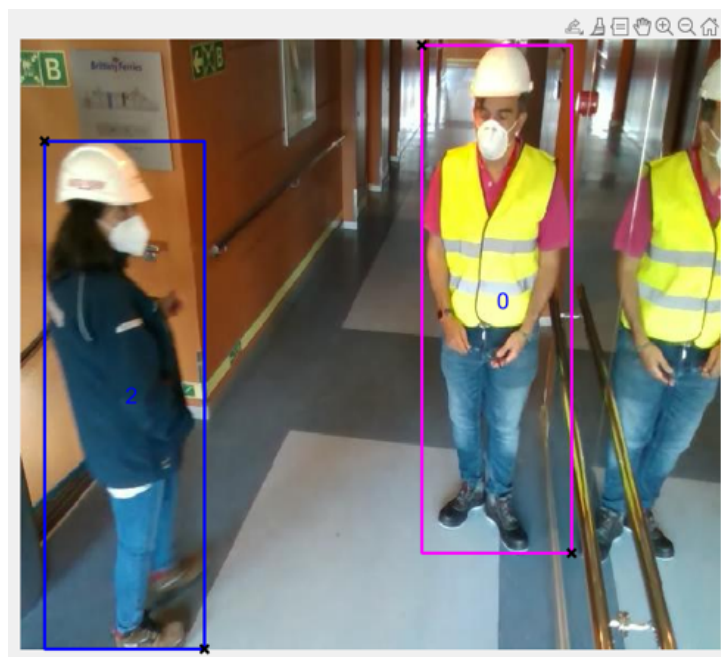


Figure 3.13: Example of the labeled Bounding Boxes.

It is worth highlighting that it is not necessary to label people and actions for each frame, since the change between frames is small in position, the labeling tool can predict how the bounding box moves between two non-consecutive frames, with a maximum interval of 30 frames. This means that the labeling tool labels automatically all the frames included between two labeled frames using linear regression at a maximum interval of 30 frames, (that correspond to 1 second in the videos).

The ground truth is then saved to a text file (with extension .gt) whose name can be chosen in the Annotation Interface (figure 3.14).

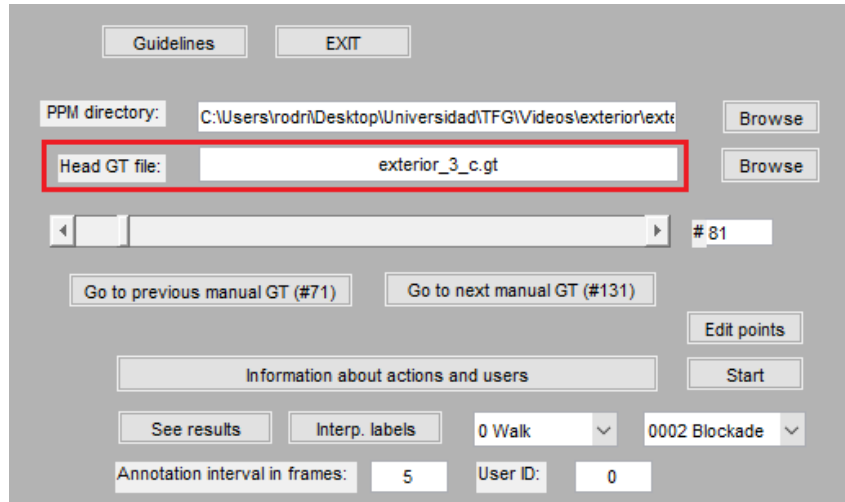


Figure 3.14: Generating GT file using the Annotation Interface

This file includes a line for each image (frame) whose structure is shown in figure 3.15, the information displayed in the cases when more than one person is included in the frame can be seen in figure 3.16.

```
000106 1 0000    2.0434    110.0749    85.3875    480.9436 4
000107 0 0000    2.0434    109.6756    85.5418    480.8992 4
000108 0 0000    2.0434    109.2764    85.6962    480.8549 4
```

Figure 3.15: GT File distribution

```
000001 1 0000    402.3006    156.2116    480.5000    356.7292 4 0001    132.7186    173.9566    202.6865    381.5721 4 0000
000002 0 0000    403.1238    156.2116    480.7058    357.8826 4 0001    132.9244    173.6904    202.6865    381.1285 4 0000
000003 0 0000    403.9469    156.2116    480.9116    359.0360 4 0001    133.1302    173.4242    202.6865    380.6849 4 0000
000004 0 0000    404.7701    156.2116    481.1174    360.1895 4 0001    133.3360    173.1581    202.6865    380.2412 4 0000
000005 0 0000    405.5932    156.2116    481.3232    361.3429 4 0001    133.5418    172.8919    202.6865    379.7976 4 0000
000006 0 0000    406.4164    156.2116    481.5290    362.4963 4 0001    133.7475    172.6257    202.6865    379.3540 4 0000
000007 0 0000    407.2396    156.2116    481.7347    363.6497 4 0001    133.9533    172.3595    202.6865    378.9104 4 0000
000008 0 0000    408.0627    156.2116    481.9405    364.8031 4 0001    134.1591    172.0933    202.6865    378.4668 4 0000
000009 0 0000    408.8859    156.2116    482.1463    365.9566 4 0001    134.3649    171.8272    202.6865    378.0231 4 0000
000010 0 0000    409.7090    156.2116    482.3521    367.1100 4 0001    134.5707    171.5610    202.6865    377.5795 4 0000
```

Figure 3.16: GT File distribution including more than one person per frame

Starting from left to right, each line includes the number of frame, if it was manually labeled (1 Yes, 0 no), and for each labeled user: the user ID, four numbers that are the coordinates of left, top and right, bottom points of the bounding box, the number associated with the action that the user is performing and finally the corresponding number to the group action developed. The data for each person in a frame are included in the same line. Finally, an extra number at the end of the line is added that corresponds to the group action performed in the frame.

### 3.3.2 Labeling criteria

Once the functionality of the labeling tool and the modifications made are explained, this section introduces the criteria for labeling both, people and actions. Due to the characteristics of the dataset, it presents several difficulties for labeling related to the high degree of occlusions and the characteristics of the different scenarios. Furthermore, it is necessary to establish a criteria to define when an action starts and finish.

### 3.3.2.1 Person Labeling criteria

The first aspect to consider while performing the labeling process is when to label a person. As explained before, the characteristics of this dataset and its scenarios often result in having people overlapping or being partially occluded by objects or other people. If the form of a person is recognizable, this means that at least 50% of the persons body is visible, then the person is labeled.

Left image in figure 3.17 displays two persons partially overlapping, in this case both of them are being labeled. Right and left pictures in image 3.17 show persons being partially covered by objects. In this cases if, for example, the bottom part of the person is completely covered the bounding box is applied to an estimation of its form, meaning that eventhough the legs can not be seen the bounding box includes the space they are supposed to occupy.

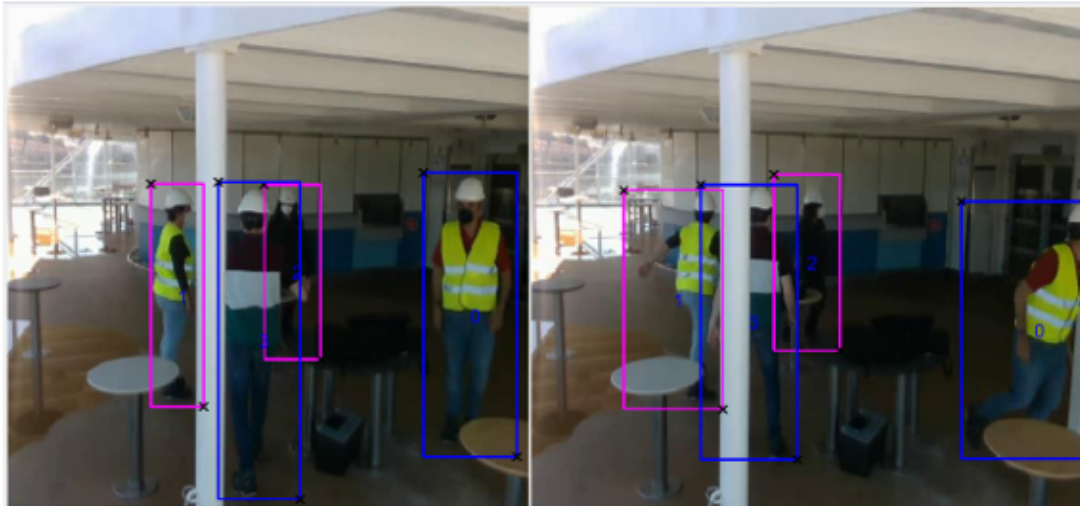


Figure 3.17: Bounding Boxes delimitations defined for partially covered persons in the scenario

As seen in the section 3.2 all the scenarios contain places that have been denominated *exits* that are places where persons in the video can leave or enter the scene. This means that the same criteria explained before must be applied to choose when to start labeling a person when enters the scene and when to stop labeling a person when its leaving the scene. When a person enters the scenario the labeling process begins when at least 50% of its body is visible and when it leaves the scenario the labeling process ends when the percentage of visible body shown drops from 50%.

1. **Starting the labeling process**, as explained before, the person that enters the scenario is going to be labeled if more than 50% of the body is shown. Figure 3.18 shows a case where a person is passing by the camera in the lower edge of the frame, it can be told by the helmet seen, but as a human figure cannot be seen due to the low percentage of the persons body showing the person passing by is not being labeled, therefore barely distinguishable due to a low percentage of the body being shown in the frame users are not considered.

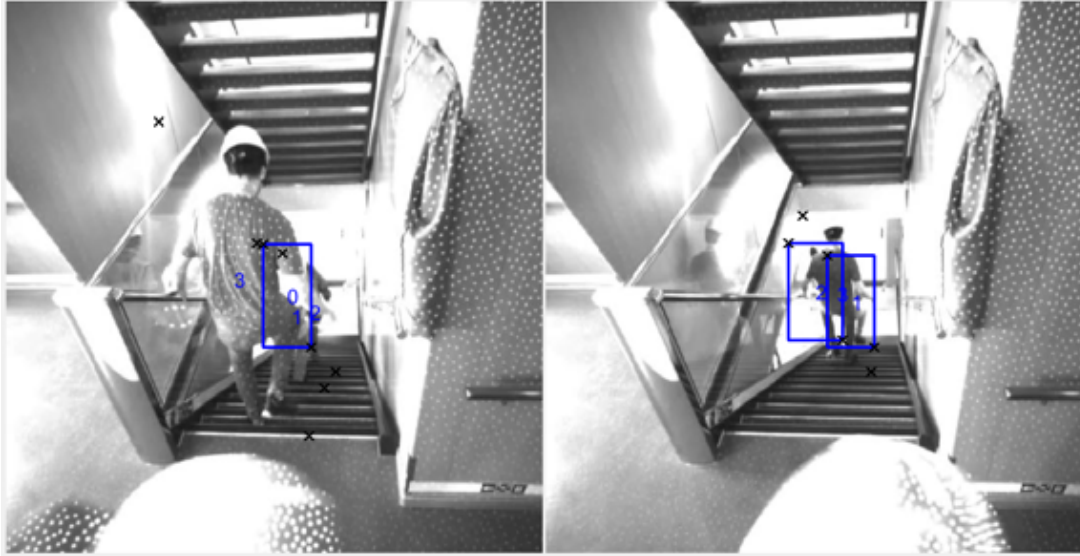


Figure 3.18: Choosing when to start labeling a user

2. **Ending the labeling process**, the same criteria explained for the start of the labeling process applies here, when a person is no longer recognizable due to the percentage shown decrementing as it is progressively leaving the frame it will no longer be labeled. Figure 3.19 displays two consecutive frames in which a person can be seen leaving the scene and, as its shape is no longer distinguishable due to progressively showing less percentage, the labeling process ends.



Figure 3.19: Choosing when to end the labeling process of a user

Finally, there are two special cases that must be considered. The first one is when we can see that there is a person in the video, but it cannot be distinguished which user is. In this case, the user identification can be obtained from other frame in the same video, what allows assigning the corresponding label since the first frame in which the person appears. Figure 3.20 displays the case in which a person can be seen but not recognized in a scenario.

The second case is the following, a user is labeled until it leaves the frame, this is a special and important case because the [HARS](#) dataset includes videos taken in narrow spaces such as corridors, and



as it could be seen in the previous images used as examples, the users have different heights with user 0 being the tallest. This means that in a narrow space, the user that is nearest to the camera can occlude other people. In this situation, the occluded people is still labeled during a maximum of 300 frames even if they are not visible, making an estimation of their position.



Figure 3.20: Special case 1 example (A person can be seen in the frame but not identified)

This is a very important aspect, as the main goal of this dataset is to train a model in charge of detecting anomalies such as a stampede, and if we do not train the model in order to understand that even if they are not visible there are still users in the frame then there could be some problems. A different criterion is applied when it is only known that there are people there going forward 300 frames apart. This means that once a person starts being labeled the labeling process continues until the person leaves the frame even if the body percentage shown is 0%.

### 3.3.2.2 Individual actions labeling criteria

As it has been explained before in section 3.2, that explains the main characteristics of the HARS dataset, it contains 6 different individual actions: walk, run, fall down, sit down, stationary and stairs, each one of them and the criteria used will be presented up next.

1. **Stationary action:** stationary definition is to stand still or not to move, so the action is considered as stationary if the person is not moving from a point in the ground, this means that no displacement is allowed. As it can be seen in the example in figure 3.21, the position of the body of the people in the frame may change but as there is no displacement made the action is considered *stationary*.



Figure 3.21: Stationary action example where a person moves but does not perform any displacement

Relating this action with the other individual actions included in the dataset, it is considered that a person is performing a stationary action when the displacement performed finishes and do not fall down. Figure 3.22 shows two consecutive frames when it can be seen that user 3 is going from performing a *walking* action until it stops its displacement and starts a *stationary* action.



Figure 3.22: Stationary action considerations coming from performing other of the actions included in the dataset

2. **Walking action:** it is considered that the action performed is *walking* when there has been a displacement between frames at a certain rate. Figure 3.23 displays the difference between *walking* action and *stationary* action in two consecutive frames user 1 and user 2 have not experience any displacement between those two frames, but user 0 has, so the *walking* action is assigned to user 0 and the *stationary* action to the rest of the users.



Figure 3.23: Walking action being labeled due to a displacement between frames from user 0

It is considered that a user has started the action of *walking* from a stationary position when the first step is made, in image 3.24 it can be seen this criterion applied on user 2.



Figure 3.24: Walking action considerations

3. **Running action**, even though it has been explained that once there is displacement of a body the action becomes *walking*, that is not entirely true as the action of *running* is included in the dataset. Initially a differentiation between both of these actions was based on a threshold applied to displacement between a set of frames: if the displacement of frames was greater than the threshold the action labeled would be *running* and if it did not reach it it would be *walking*, however, this criteria application is not that simple as there are some videos in the dataset in which the users walk fast, so the criteria used for making a difference between these two different actions is the motion of the body. Figure 3.25 displays the second criteria used to differentiate both actions as the displacement between frames is similar but the body motion of the persons included is different.

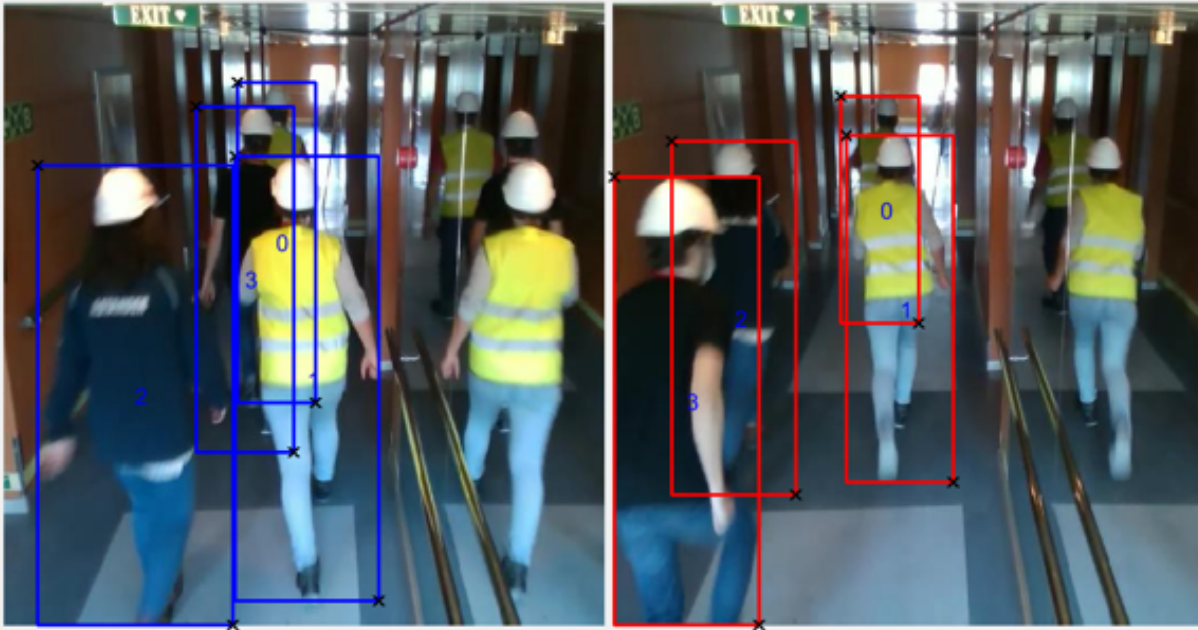


Figure 3.25: Running and Walking differentiation

As in the *walking* criteria explained before it is considered that a user has started the action of *running* from a stationary position when the first step is made.

4. **Falling action**, this action is the act of falling down into the floor, in Figure 3.27 it can be seen that this action is labeled when a user falls down and lays on the floor, not when it the user trips over, the image on the left displays a case where the user almost falls down, but it never reaches the floor or the stairs so the action of falling down is never completed and it is not taken into consideration.

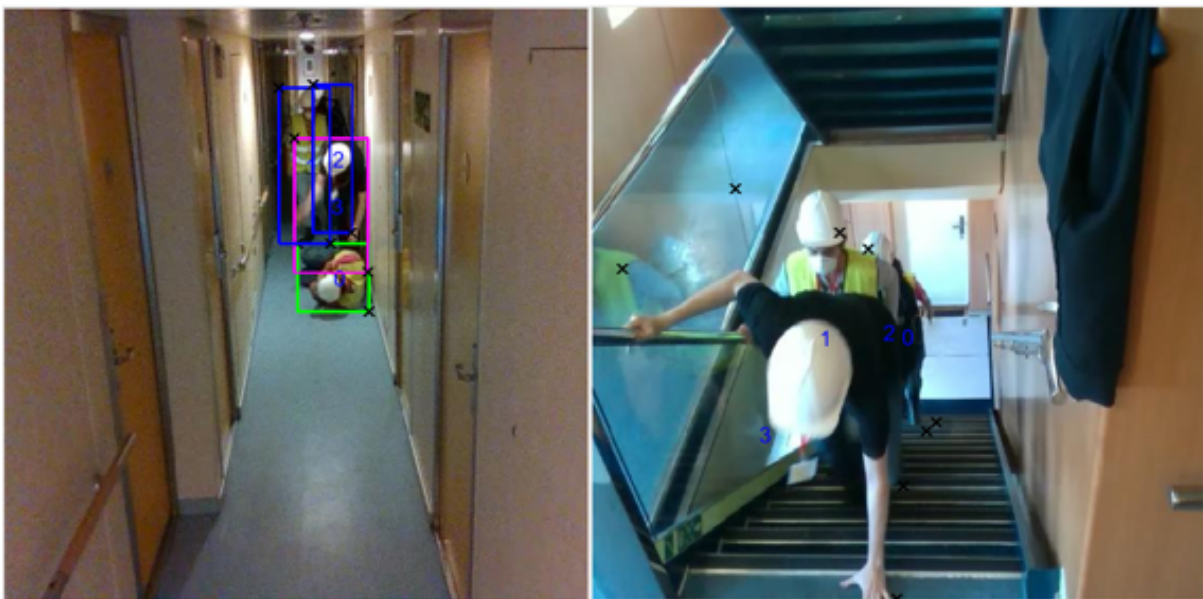


Figure 3.26: Falling Down action

5. **Stairs action**: this action involves a user to climb up or down through a staircase. It has not any extra criteria applied to it with the exception that a person would be considered performing this

action when it takes the first step into a staircase and that it stops performing that action when it takes the first step out of the staircase. Figure 3.27 displays two consecutive frames in which user 0 changes from *stairs* action to *running* action once he steps out the staircase.



Figure 3.27: Beginning and end of stairs action

It is important to consider that when videos are being labeled in which the users use stairs, there is a need to include hierarchical priorities as the main objective of the model trained is to detect anomalies in these videos, so the actions of falling down and stationary are prioritized if they occur in a staircase segment of the scenario. Figure 3.28 shows cases in the dataset in which *stationary* and *falling down* actions are prioritized over the *stairs* action even though they are taking place in a staircase.

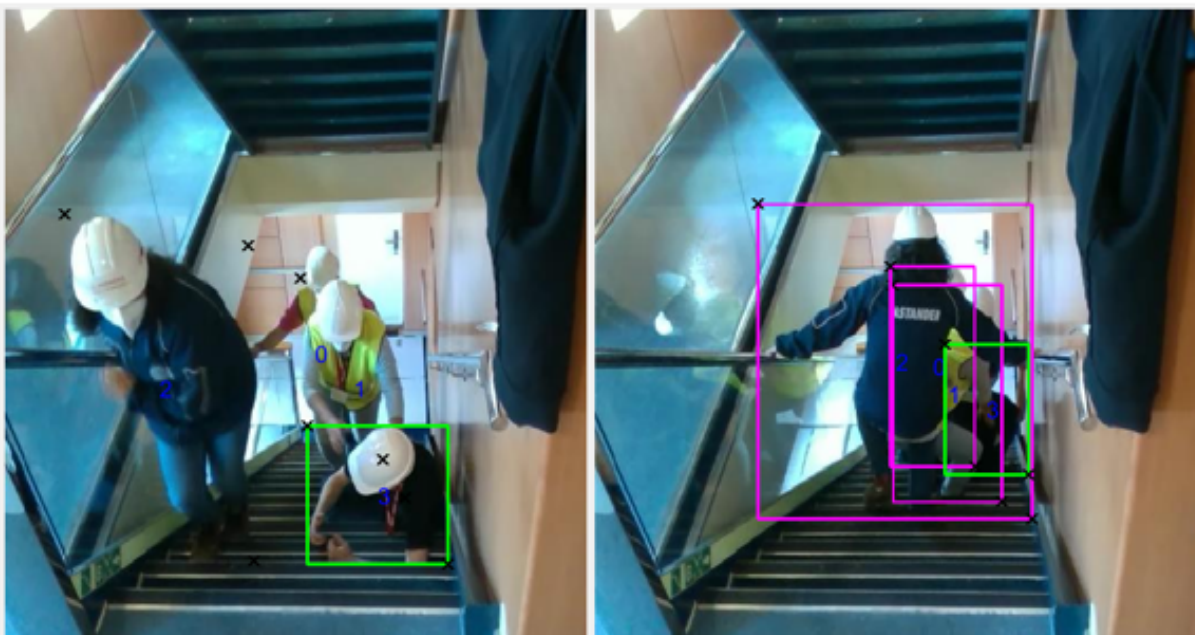


Figure 3.28: Falling Down in stairs action

### 3.3.2.3 Group actions criteria

One important characteristic of the [HARS](#) dataset is that it contains both individual and group actions. In group actions all the users in the video or the majority of them perform a set of actions that, as a whole, can be considered a group action of interest for detection anomalies in large cruise ships. The two group actions that can be seen throughout the dataset are presented below.

1. **Stampede:** a stampede by definition refers to a situation in which a group of people suddenly starts running in the same direction and close to each other. [Figure 3.29](#) displays a case in which the 3 users included in the video start running in the same direction and are close to each other, therefore both shown cases are considered a stampede.



Figure 3.29: Stampede action sample

During the presentation of the labeling criteria for individual actions, a differentiation between *walking* fast and *running* has been defined. The same criteria applies here, only when the group of users is *running* the group action labeled would be *stampede*. [Figure 3.25](#) displays the criteria explained before to differentiate *walking* and *running* actions.

2. **Blockade:** a blockade is, by definition, a situation in which the path that a person is following is suddenly blocked by an object or, in the case of the dataset, another person and the flow of people through a section of the scenario is interrupted. In the dataset, all the blockades occur due the one person falling down and the rest or the majority of the persons included in the video become stationary. [Figure 3.30](#) displays the case in which user 0 falls down and does not get up, therefore as the scenario is a narrow corridor the flow of the rest of persons is being interrupted.



Figure 3.30: Blockade action sample

#### 3.3.2.4 Important considerations about the labeling process

To finish up with the labeling criteria is important to highlight the following.

**IR videos**, due to the complexity of the scenarios included in the dataset and the different lighting conditions that they include, there are some difficulties while labeling the **IR** videos due to their poor characteristics. Figure 3.31 shows two exact frames from the same video, the image displays the **IR** video and the **RGB** video respectively, it can be seen that for the **IR** scenarios there are problems to identify the users that appear in it.

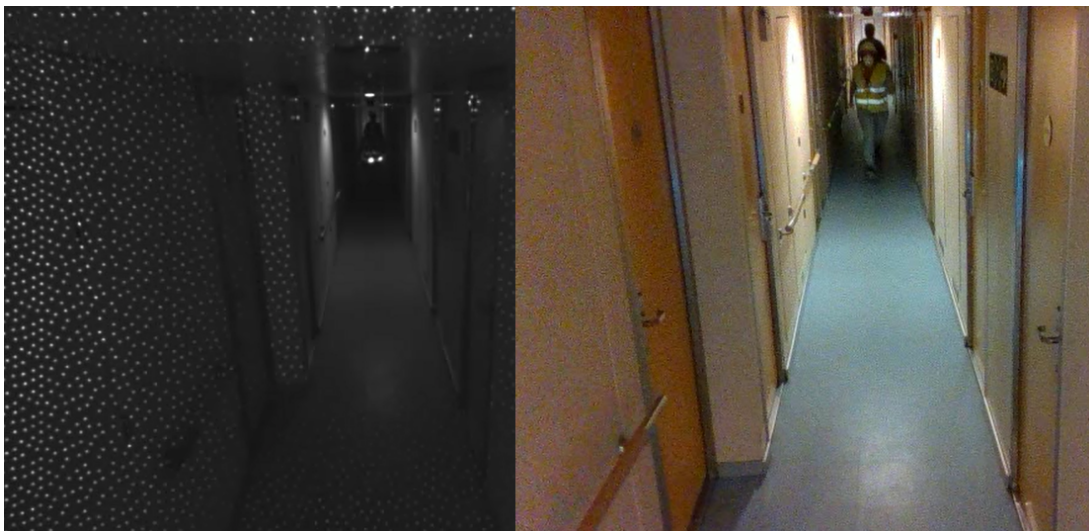


Figure 3.31: Example of poor characteristics of the IR videos

This problem was solved by initially labeling all the **RGB** videos, and then use the resulting gt files for the **IR** videos. As seen in the section that explained the characteristics of the dataset the **IR** camera used to record the videos provide a wider view and a different **FOV** than the one used to record the **RGB**

videos. The labeling tool used allows to modify a gt file, therefore with the IR video as base the gt files were modified and fixed to the dimensions of the IR videos.

### 3.4 Labeling result

Table 3.3 includes the number of frames corresponding for an action per scenario included in the dataset.

Scenario	Walk	Run	Fall Down	Stationary	Stairs	Blockade	Stampede
Common Area	7604	4641	0	5735	0	0	1006
Corridor Cabins	5915	7076	1500	3676	0	1091	2349
Exit Stairs	1941	100	0	443	2338	0	0
Exterior	4237	3029	295	4120	462	332	0
Lifeboat	760	652	0	2458	462	0	0
Stair Corridors	4079	0	1010	368	7440	986	0
Total	24536x2	15498x2	2805x2	16797x2	10240x2	2409x2	3355x2

Table 3.3: Video, persons and frames distribution of the HARS dataset

### 3.5 Dataset evaluation

This section describes the implementation y evaluation of YOLOv3 for people detection using the HARS dataset. It has been chosen due to its ballance between computational cost and accuracy.

#### 3.5.1 YOLOv3 implementation in Google Colab

As it has been explained previously, in order to validate the labeling, it is used a YOLO implementation in Google Colab. In particular, YOLOv3 is used to detect people and objects. In order to validate both, the YOLO implementation and the dataset labeling, people detections are compared to the labels in the ground truth files obtained from the labeling tool, this process is performed for all the videos in the dataset.

For this TFG, YOLOv3 has been implemented in Google Colab [52], since it allows programming and executing Python code in a web browser, with access to remote GPUs, whitout needing any specific configuration. YOLO allows obtaining the coordinates of the bounding boxes corresponding to the people detected in any image, that are saved to a text file. The comparison of the labels and the YOLOv3 detections allows evaluating the HARS dataset. Figure 3.32 shows the flowchart of this TFG highlighting the part of focus of this section of the chapter.



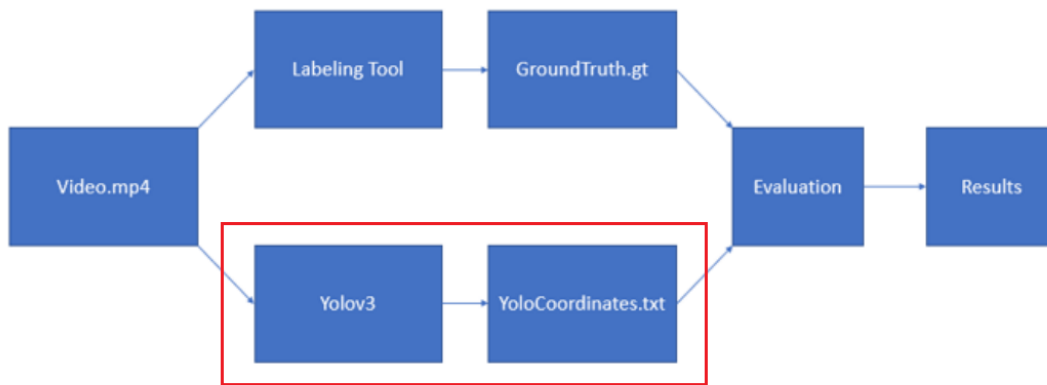
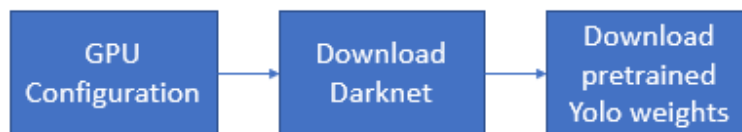


Figure 3.32: General flowchart of this TFG

The steps followed to implement [YOLOv3](#) in Google Colab are now presented. A general block diagram including the different steps are shown in [Figure 3.33](#).

Figure 3.33: Flowchart of the implementation of [YOLOv3](#) in Google Colab

1. **Graphical Processing Unit (GPU) configuration:** due to the characteristics of the PC on which the [TFG](#) has been developed, which does not have a powerful [GPU](#), for the implementation and execution of [YOLOv3](#) it has been used Google Colab. As it has been mentioned, it is an online alternative to program and execute python code in a browser, that does not require any configuration and allows access to [GPUs](#).
2. **Download Darknet,** Darknet, [YOLOv3](#) backbone is downloaded from [\[53\]](#) repository, and then installed and compiled. This [YOLOv3](#) implementation [\[54\]](#) has been chosen because it allows to receive as output a video representing the bounding boxes and probability of each detection, useful for a visual representation and a first take on how accurate [YOLOv3](#) is. The original implementation has been modified in order to also obtain a text file including information about the detection such as bounding box coordinates, probability and frame information. This .txt file obtained is used in the evaluation process. [Figures 3.34](#) and [3.35](#) show the process described above.

```
[ ] import os
os.environ['PATH'] += ':/usr/local/cuda/bin'
!git clone https://github.com/AlexeyAB/darknet/

Cloning into 'darknet'...
remote: Enumerating objects: 15363, done.
remote: Total 15363 (delta 0), reused 0 (delta 0), pack-reused 15363
Receiving objects: 100% (15363/15363), 13.98 MiB | 16.92 MiB/s, done.
Resolving deltas: 100% (10332/10332), done.

[ ] !apt install gcc-5 g++-5 -y
!ln -s /usr/bin/gcc-5 /usr/local/cuda/bin/gcc
!ln -s /usr/bin/g++-5 /usr/local/cuda/bin/g++
```

Figure 3.34: Repository Cloning and Installing

```
[ ] %cd darknet
!sed -i 's/GPU=0/GPU=1/g' Makefile
!sed -i 's/OPENCV=0/OPENCV=1/g' Makefile
!make
```

Figure 3.35: Compile

3. **Download pretrained YOLO weights:** the evaluation and obtaining of the output files and videos for the dataset is performed with a pretrained YOLOv3 version, that has been trained with COCO dataset. It is important to highlight that since the YOLOv3 model used has been trained with COCO [37] dataset it contains 80 different classes as seen in section 2.2.3. From all of those classes, the only one of interest is *person*. COCO dataset weights has been chosen due to the characteristics of the dataset itself, it contains dense pose, as for the characteristics of the HARS dataset, it is important to simultaneously detect people and perform a mapping between pixels that belong to a specific person as along the HARS dataset there are a lot of mapping between persons in the scenarios. Image 3.36 displays the process in which this weights are downloaded.

```
[ ] !wget https://pjreddie.com/media/files/yolov3.weights
!chmod a+x ./darknet

--2021-11-18 10:31:31-- https://pjreddie.com/media/files/yolov3.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)[128.208.4.108]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248007048 (237M) [application/octet-stream]
Saving to: 'yolov3.weights'

yolov3.weights      100%[=====] 236.52M  23.1MB/s   in 11s

2021-11-18 10:31:42 (21.6 MB/s) - 'yolov3.weights' saved [248007048/248007048]
```

Figure 3.36: Downloading of pretrained YOLOv3 weights with COCO dataset

4. Once YOLOv3 is implemented the process of feeding the system with the videos of the dataset begins, the videos of the HARS dataset are in a .mp4 format with a size of 640x480 and 30 frames per second. Figure 3.37 represents the flowchart followed for this process.

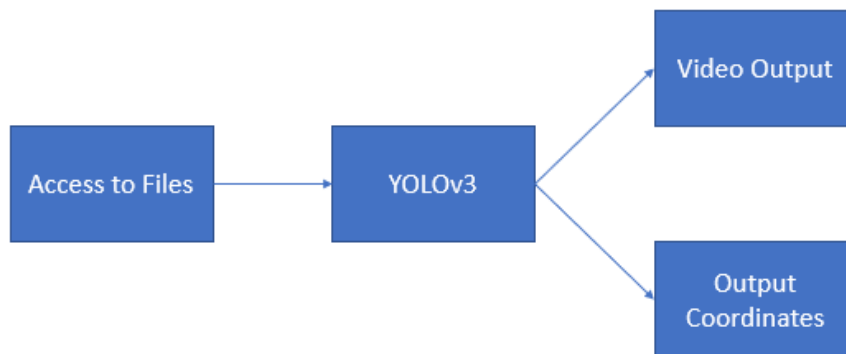


Figure 3.37: Flowchart representing how to obtain the different outputs

5. **Create access to computer files**, as the interest of implementing **YOLOv3** in Google Colab is to input the dataset videos and obtain the result files and output videos that include a visual representation of the bounding boxes predictions a way to access the dataset videos is created. Figure 3.38 displays the necessary process to extract videos directly from the computer working with.

```
[ ] from google.colab import files
    uploaded = files.upload()
```

Figure 3.38: Importing the videos directly from the computer

6. **Extraction of the output video provided by YOLOv3**, the next line shown allows to obtain the output video in an .avi format.

```
!./darknet detector demo cfg/coco.data cfg/yolov3.cfg
yolov3.weights -dont_show commonarealc.mp4 -i 0
-out_filename stairscorridor3c.avi -thresh 0.7
```

This process outputs a video that shows the bounding boxes predicted by **YOLO** and the percentage that represents how certain **YOLO** is that the object detected is, in this case, a person. The obtained video gives an idea of how well **YOLO** performs the detection and how well it performs in a dataset of this type. Figure 3.39 shows a frame in which the information that **YOLO** provides can be seen.

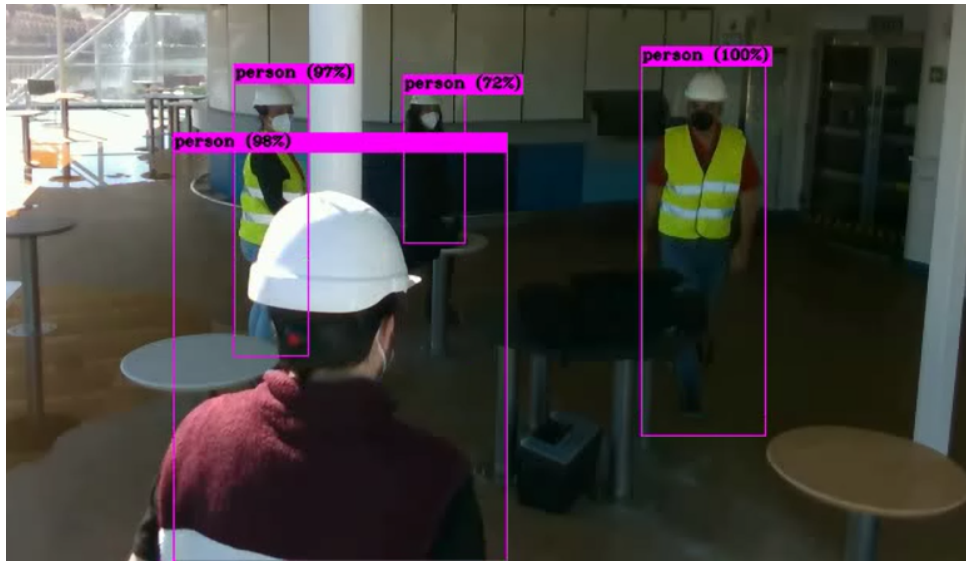


Figure 3.39: YOLOv3 video output example

7. **Extraction of the output coordinates provided by YOLOv3**, the next line shown allows to obtain the output coordinates of the detections in an .txt format.

```
!./darknet detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights
stairs corridors6i.mp4 -dont_show -ext_output stairs corridors6i.mp4 >
stairs corridors6i.txt
```

The video output does not provide specific information needed to evaluate the model with metrics such as Intersection over Union (IoU). For this purpose, it is needed to extract the coordinates of the bounding boxes that YOLO generates and write them into a .txt file containing those coordinates for later evaluation. These coordinates are output in the format shown in figure 3.40, that displays the format of the information provided by YOLO. It provides the percentage related to the detection and the corresponding coordinates, as well as a number related to the frame in which the detection has been made.

```
FPS:2.6          AVG_FPS:0.0
Objects:

person: 99%      (left_x: 405  top_y: 162  width: 69  height: 143)
person: 40%      (left_x: 133  top_y: 196  width: 61  height: 148)

FPS:3.7          AVG_FPS:0.0
Objects:

person: 99%      (left_x: 406  top_y: 162  width: 68  height: 143)
person: 34%      (left_x: 133  top_y: 196  width: 61  height: 149)
```

Figure 3.40: YOLO GT file information format

### 3.5.2 Data processing for evaluation

Once obtained the ground truth files corresponding to the manual labeling and the YOLOv3 predictions, it is necessary to process those files fitting them into the same format. For this, MATLAB is used. Below, the detailed process and the important factors to consider for each one of these files is explained.

### 3.5.2.1 Preparing Ground Truth file for evaluation

The first thing considered for preparing the ground truth files is the format in which the labeling tool gives back the coordinates of the ground truth bounding boxes, action, users and frames. Figure 3.41 displays, once again, the format in which this data is obtained.

```
000001 1 0000 392.7015 145.5951 471.9129 339.0551
000002 0 0000 393.2623 147.6635 472.6139 344.6521
000003 0 0000 393.8231 149.7320 473.3149 350.2490
000004 0 0000 394.3839 151.8004 474.0159 355.8460
000005 0 0000 394.9447 153.8689 474.7169 361.4429
000006 1 0002 189.4157 168.7129 257.4113 351.2224
000007 0 0002 190.1167 170.1730 256.2897 350.1273
000008 0 0002 190.8177 171.6331 255.1681 349.0323
000009 0 0002 191.5186 173.0931 254.0466 347.9372
000010 0 0002 192.2196 174.5532 252.9250 346.8422
```

Figure 3.41: Ground Truth coordinates distribution

First, file information is read and stored line by line. Figure 3.42 shows the array generated from reading the contents of the file.

	1												
1	000001	1	0000	392.7015	145.5951	471.9129	339.0551	4	0001	132.6358	179.0551	208.3423	...
2	000002	0	0000	393.2623	147.6635	472.6139	344.6521	4	0001	131.5142	177.3517	208.2021	...
3	000003	0	0000	393.8231	149.7320	473.3149	350.2490	4	0001	130.3926	175.6483	208.0619	...
4	000004	0	0000	394.3839	151.8004	474.0159	355.8460	4	0001	129.2711	173.9448	207.9217	...
5	000005	0	0000	394.9447	153.8689	474.7169	361.4429	4	0001	128.1495	172.2414	207.7815	...
6	000006	1	0002	189.4157	168.7129	257.4113	351.2224	4	0001	127.0279	170.5380	207.6413	...
7	000007	0	0002	190.1167	170.1730	256.2897	350.1273	4	0001	126.7475	170.1730	207.7815	...
8	000008	0	0002	190.8177	171.6331	255.1681	349.0323	4	0001	126.4671	169.8080	207.9217	...
9	000009	0	0002	191.5186	173.0931	254.0466	347.9372	4	0001	126.1868	169.4429	208.0619	...
10	000010	0	0002	192.2196	174.5532	252.9250	346.8422	4	0001	125.9064	169.0779	208.2021	...

Figure 3.42: Example of the generated array with Ground truth file information

The next step done to prepare the data for evaluation is to fill empty coordinates with zeros. This process has been chosen due to the interest on working with arrays as the use of the specific toolbox functions used for the evaluation process is eased if the work is done using arrays indexing. Therefore, as the format selected is arrays, the size of the arrays must be constant through all its content, so the string generated after this process contains the coordinates of the file plus the coordinates (0,0,1,1) to indicate an empty detection. The number of extra coordinates included in this process stack up to 4, in order to have a maximum of 5 detections in the worst-case scenario (just one person detected), doing this if just one person is detected the remaining ones would have a non-existing bounding box (this is very important and is explained in detail along this chapter).

It is needed to extract the number of frames included in this file for the evaluation as the labeling tool used to obtain the ground truth coordinates does not include frames without detections and YOLO text file does. Figure 3.42 shows a case in which there are no labeling process between frames 171 and 281, therefore it can be seen that the gt file does not contain empty frames.

```

000170 0 0002 236.9502 87.9824 304.0370 256.8253 0
000171 1 0002 238.6994 87.8937 305.5804 256.4704 0
000281 1 0001 225.3232 91.4427 291.1752 260.0194 1
000282 0 0001 219.0810 92.3299 288.9115 260.2560 1
000283 0 0001 212.8387 93.2172 286.6479 260.4926 1

```

Figure 3.43: Reading and storing corresponding frame number

Due to the frame information display in the manual ground truth file, an extraction of the string between positions is performed and the result is converted to an integer number in "double" format. This information is stored in an array that contains all the frames with a detection of the ground truth in an array.

Once obtained the number of frames, the detection coordinates are extracted.

Again extracting between two different markers and performing a data type conversion allows to obtain the coordinates of the detections in an array format.

### 3.5.2.2 Preparing YOLO coordinates file for evaluation

Finished the processing of the data of the ground truth bounding boxes, the processing of the **YOLO** bounding boxes information begins. It is important to take into account the format in which **YOLO** outputs the information corresponding to the video input. Figure 3.40 shows the format of the information provided by **YOLO**.

As seen in the figure, **YOLO** gives information regarding the type of object detected, how sure it is that is a certain object in a percentage representation and the coordinates of its bounding box.

The first step done, as before, is to read the file, it is done by using functions that output a character array containing all the characters of the file Image 3.44 displays the content of the character array obtained by using these functions.

```

FPS:0.0   AVG_FPS:0.0
Objects:

person: 100% (left_x: 405 top_y: 161 width: 69 height: 141)
person: 45% (left_x: 133 top_y: 193 width: 62 height: 151)

FPS:1.3   AVG_FPS:0.0
Objects:

backpack: 25% (left_x: 350 top_y: 292 width: 106 height: 54)
person: 99% (left_x: 405 top_y: 162 width: 69 height: 144)
person: 36% (left_x: 133 top_y: 196 width: 61 height: 149)

```

Figure 3.44: Content of the character array obtained

**YOLO** does not only detect persons, but it also detects objects, since **COCO** dataset weights were used, this leads to the implemented **YOLO** model to make detections included in a total of 80 different classes, once implemented in this dataset apart from persons it detects other different classes, such as *backpacks*, *suitcases*, *dining tables*, *handbags*, *skateboards* and *chairs*. The next is to remove these detections as they do not provide relevant information. Figure 3.45 shows the array that contains the information once the non-useful detections are removed, in this case, the coordinates of the *backpack* detection are removed.

```

FPS:0.0    AVG_FPS:0.0
Objects:

person: 100%    (left_x: 405 top_y: 161 width: 69 height: 141)
person: 45%    (left_x: 133 top_y: 193 width: 62 height: 151)

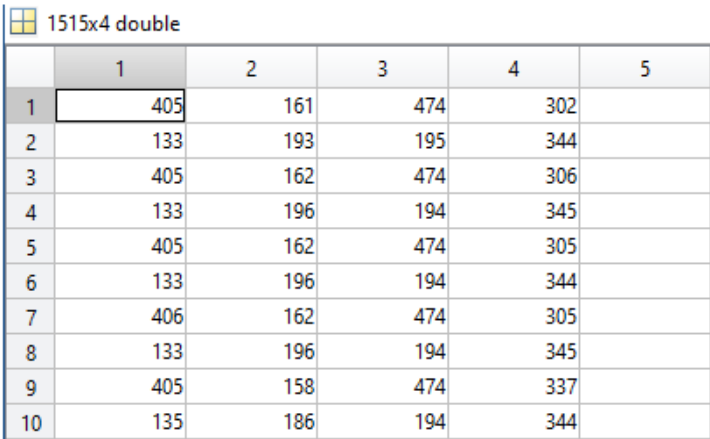
FPS:1.3    AVG_FPS:0.0
Objects:

backpack)
person: 99%    (left_x: 405 top_y: 162 width: 69 height: 144)
person: 36%    (left_x: 133 top_y: 196 width: 61 height: 149)

```

Figure 3.45: Example of array in which non-useful detection coordinates have been removed.

The next step is to extract the coordinates corresponding to each one of the detections included in the output .txt file that YOLOv3 implementation provides. It is important to highlight that YOLO provides coordinates in a different format than the labeling tool. It provides the height and width of the bounding box and the coordinates of one of the axis, therefore to convert to the same format as the ground truth files (wich includes the coordinates of one of the lower and upper axis) those two values must be added to the coordinates of the lower axis of the bounding box. This results in an array containing the coordinates in the same format as the labeling tool. Figure 3.46 displays the content of the array that contains the coordinates of the detections.



	1	2	3	4	5
1	405	161	474	302	
2	133	193	195	344	
3	405	162	474	306	
4	133	196	194	345	
5	405	162	474	305	
6	133	196	194	344	
7	406	162	474	305	
8	133	196	194	345	
9	405	158	474	337	
10	135	186	194	344	

Figure 3.46: YOLO coordinates assigned to each 'person' detection

To compare it with the manual ground truth file and obtain results, it is needed to know how many people per frame there are. YOLO provides one detection per line, therefore it is needed to display them in a horizontal way as in the ground truth file, in which all the coordinates of the bounding boxes included in one frame are displayed in the same line. The way chosen to approach this conversion is counting how many times per frame the string *person* appears, and, with the convenient transformations, an array containing the number of persons per frame is obtained.

Using this array and the one containing the coordinates, and performing the same step used in the manual coordinates to fill empty positions, the resulting variable is generated. Parallel to this, an array containing the frame number is created counting the times that the string *AVGFPS* appears, as it only appears once per frame, is created. Figure 3.47 displays a nested array in which there are included the number of frame and the coordinates of the corresponding detections in that frame.

	1	2
1	1x25 string	
2	1x25 string	
3	1x25 string	
4	1x25 string	
5	1x25 string	
6	1x25 string	
7	1x25 string	
8	1x25 string	
9	1x25 string	
10	1x25 string	

	1	2	3	4	5
1	405	161	474	302	
2					
3					
4					
5					
6					
7					
8					
9					
10					

Figure 3.47: YoloGTtxt distribution

With the information obtained from the process previously explained, it is possible to obtain different metrics that provide information about the performance in person detection and that allow to validate the dataset labeling. The used metrics are explained in the *Experimental Results* chapter.



# Chapter 4

## Results

### 4.1 Introduction

In this chapter the obtained results are presented. Firstly, the experimental setup is described, this includes the explanation about the used metrics for this process. After that, there is presented the procedure followed for the evaluation. Finally, the obtained results are shown and analyzed in detail.

### 4.2 Experimental setup

This section describes the used metrics as well as the data processing in order to obtain the experimental results.

#### 4.2.1 Metrics used for evaluation

##### 4.2.1.1 Classification metrics

In a classification process, a positive detection is something that has been detected and negative detection not being detected. With this in mind, some basic concepts used to define metrics are defined below:

- True Positive (TP): correct detection, the case in which the systems declare a positive and it is truly positive.
- False Positive (FP): wrong detection, the case in which the system declares a positive but it is truly a negative.
- False Negative (FN): a negative detection that truly is a positive detection.
- True Negative (TN): a negative detection that truly is a negative detection.

Based on the previously mentioned concepts, there can be defined some important parameters used to evaluate the performance of a model:

- **Precision:** shows the quality of a model representing the truly positive detected cases among all the detections performed by the model. It is defined in the equation 4.1.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \quad (4.1)$$

- **Recall:** it is the ability of a model to detect all the ground truth true detections. It is defined in equation 4.2.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}} \quad (4.2)$$

- **Accuracy:** it is the fraction that was correctly predicted among the total number of cases. It is defined in equation 4.3.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (4.3)$$

#### 4.2.1.2 Intersection over Union (IoU)

Other widely used metric is the **IoU**, that allows measuring the behavior of detection systems on a particular dataset. Applied to the context of this **TFG**, this metric allows comparing the results obtained from **YOLO** with the available ground truth, product of the labeling process.

The **IoU** represents the area of overlap between two bounding boxes divided by the area of union, as it can be seen in figure 4.1, in which it is shown the mathematical relationship and a graphical representation.

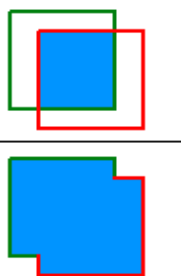
$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 4.1: IoU mathematical definition

#### 4.2.1.3 AUC and ROC curves

Another metric widely used to evaluate the performance of classification models are Area Under ROC Curve (AUC) represents the separability. It correspond to the area under the Receiver Operating Characteristic (ROC) curve.

There are the performance measurements done at different thresholds and tell us how good the model is at predicting the classes, the higher the **AUC** the better the model is at distinguishing classes.

The **ROC** curve is obtained by plotting the True Positive Rate (TPR) on the Y axis and False Positive Rate (FPR) on the X axis for different values of a chosen parameter, that is usually a threshold. Figure 4.2 displays an example of a **ROC** curve, with the axis label, name of the curve and name of the area under the curve.

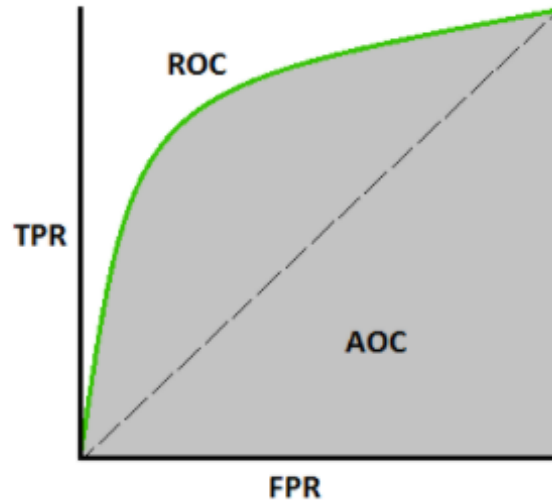


Figure 4.2: ROC curve example

The [TPR](#) and [FPR](#) are defined as the expressions displayed in equations [4.4](#) and [4.5](#) respectively:

$$TPR = \frac{TP}{TP + FN} \quad (4.4)$$

$$FPR = \frac{FP}{TN + FP} \quad (4.5)$$

To understand more about the AUC-ROC curve, figure [4.3](#) displays the [ROC](#) curves corresponding, from left to right, to a perfect model (in which all the classifications are correct), a model without classification capacity and a model in which all predictions are wrong.

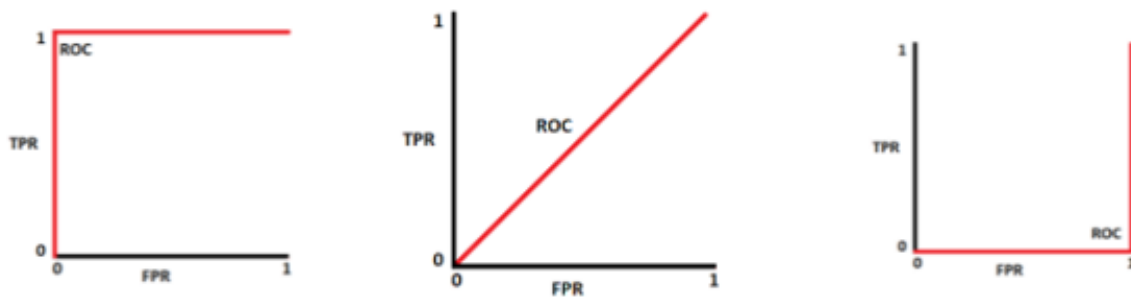


Figure 4.3: ROC curve visual examples for specific cases.

The first case corresponds to the perfect classifier, as it can be seen the [ROC](#) curve corresponds to a model with 0 false detections and an accuracy of 100% in the detections. The middle one corresponds to a classification model that performs random classifications, and as it can be seen the area under the curve corresponds perfectly to a 50%, this means that in a case of a binary classification problem by probability half the detections falls down into a class and the rest to the other. Finally, the right one corresponds to a model in which all the classifications are wrong (worst case scenario), this can be seen in the area under the curve that is 0%.

## 4.2.2 Preparing the data for evaluation

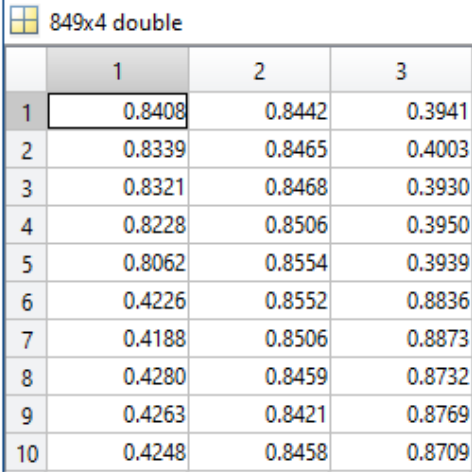
Once the data has been prepared, the evaluation is carried out, obtaining different metrics. To do that, there are used two functions included in two different MATLAB toolboxes: *Computer Vision Toolbox* and *Statistics and Machine learning Toolbox*. The output of the evaluation process are two different matrixes. This two matrixes are needed in order to use the functions included in the toolboxes and obtain the results. It is important to highlight that eventough MATLAB allows other formats to work with the matrix format was chosen due to the function *bboxOverlapRatio* only accepts matrixes including the coordinates as inputs. The content of these two matrixes include **IoU** percentage corresponding to the labeled bounding box and its corresponding **YOLO** one and the percentage related to how sure **YOLO** is that the detections performed belong to the class *person*.

### 4.2.2.1 IoU matrix

One of the metrics of interest in the evaluation process of the dataset is the **IoU**, that is obtained using the function *bboxOverlapRatio* included in the *Computer Vision Toolbox*. Given the coordinates of two different bounding boxes, this function returns the percentage of overlap ratio.

It is important to take into account that the order of the people's bounding boxes is not the same in the **YOLO** results file and in the manual ground truth one. This means that the first person detection displayed in the **YOLO** file might be third person detected in the labeled ground truth. The solution for this is to evaluate the bounding box **IoU** between each of the manually labeled bounding boxes and every one of the **YOLO** bounding boxes, saving the results in an array. Once obtained the array a selection of the maximum value in the array is performed, this selects the bounding box with the most overlap.

Figure 4.4 shows an example of the result of this process for a given frame. It is an array with dimensions: number of frames by number of persons contained in the frame, where each value corresponds to the **IoU** of each pair of bounding boxes in each frame.



	1	2	3
1	0.8408	0.8442	0.3941
2	0.8339	0.8465	0.4003
3	0.8321	0.8468	0.3930
4	0.8228	0.8506	0.3950
5	0.8062	0.8554	0.3939
6	0.4226	0.8552	0.8836
7	0.4188	0.8506	0.8873
8	0.4280	0.8459	0.8732
9	0.4263	0.8421	0.8769
10	0.4248	0.8458	0.8709

Figure 4.4: IoU matrix example

Figure 4.5 represents in a histogram the level of overlap between the **YOLO** .txt output file bounding boxes and the manual ground truth file ones for the case of **RGB** videos. Considering that a 1 would be a perfect fit of the bounding boxes along all the frames and the worst-case scenario would be a 0 it can be seen that in this case scenario precision wise **YOLO** is very accurate considering the characteristics of the dataset as the majority of the dataset predictions fall in the 0.7 or greater bins.

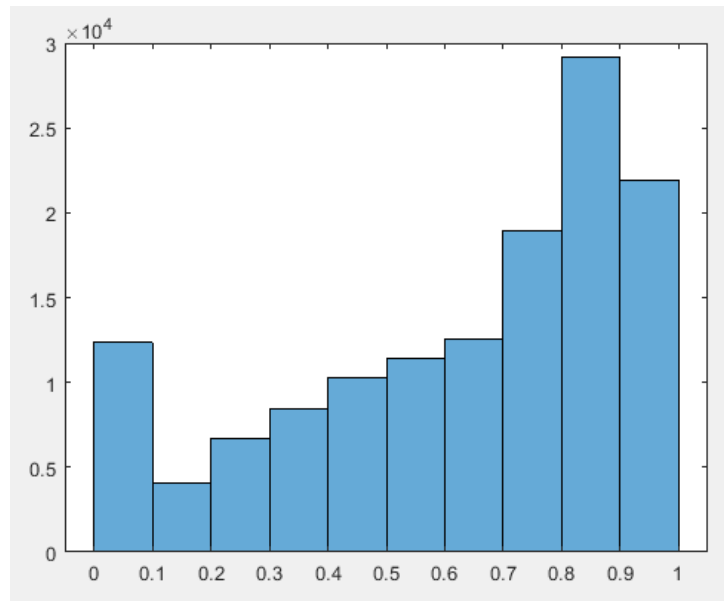


Figure 4.5: IoU results matrix plot

#### 4.2.2.2 YOLO accuracy matrix

Other significant data needed for evaluation is how certain **YOLO** is that a detection corresponds to a class. In this case it is taken into consideration the percentage assigned to each detection that **YOLO** considers it is a person. This percentage can be obtained for the **YOLO** results text file.

Considering the format in which **YOLO** displays the percentage of detection values by just applying an extraction of string and a conversion to double, the probability can be obtained and stored in a variable, then the remaining step would be to escalate those values in a range from 0 to 1 for the *perfcurve* function to understand the values, since the *perfcurve* only understands numeric values between 0 and 1. Figure 4.6 displays the last step explained where the percentage obtained from the **YOLO** output text file is fixed to be between 0 and 1.

1501x1 double		1501x1 double	
	1		1
1	100	1	1
2	45	2	0.4500
3	99	3	0.9900
4	36	4	0.3600
5	99	5	0.9900
6	40	6	0.4000
7	99	7	0.9900
8	34	8	0.3400
9	100	9	1
10	99	10	0.9900

Figure 4.6: YOLO accuracy matrix

### 4.3 Experimental results

To finish with the results chapter, below there are presented the experimental results obtained after evaluation the whole dataset. There are shown the ROC curves for both, the IoU thresholds and the YOLO scores one in the case of RGB.

*Perfcurve* function uses as input a labels vector and a corresponding score associated with those labels, in a classification example would be (how sure the classifier is that an object belongs to that class) and a class to evaluate in a string of characters format. For this function to work properly it is necessary that the labels vector includes more than one class, that's why the code used includes two more labels apart from *person*, which are *versicolor* and *boletus*, other thing to consider is that the function need at least a relation 85-15 between the data evaluated and the complementary one (used to fill the non-used labels), this space is filled in the IoU vector and the porcentaje vector with the *rand* function, which provides a random number between 0 and 1 associated to a porcentaje. The *perfcurve* function returns as output the X and Y coordinates to display de ROC curve, the AUC, and the *optimal operating point value* as “the value whose sensitivity and specificity are the closest to the value of the area under the ROC curve and the absolute value of the difference between the sensitivity and specificity values is minimum” [55].

This process allows obtaining the ROC curves of the IoU and YOLO accuracy for any of the scenarios included in the HARS dataset. This means that an evaluation of the whole dataset or any of the scenarios included can be made based on the ROC curves obtained, for this analysis the most visual parameter to consider is the AUC. And as seen along this chapter, the greater the value of the AUC the best performance by YOLO in that given scenario.

Figure 4.7 shows the ROC curves corresponding to the IoU and the YOLO accuracy percentage obtained from the whole dataset RGB videos respectively.

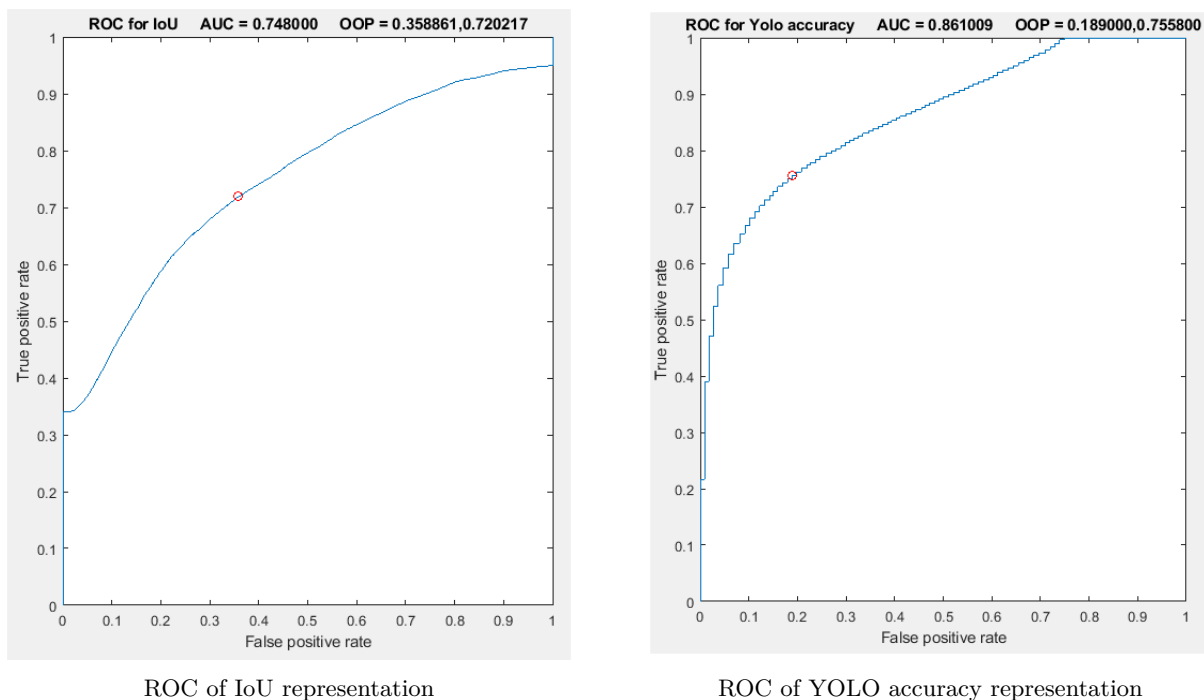


Figure 4.7: ROC curves representation for IoU and YOLO accuracy

As it can be seen by the information provided eventhough HARS is a complicated dataset, YOLO performs good in the precision of the bounding boxes that assigns to the detections made as it can be

seen in the **ROC IoU** graph (**AUC** of 0.748), and the accuracy of **YOLO**, that provides information about how sure it is that a detection belongs to the *person*, has good results as well, in the **ROC** of **YOLO** accuracy an **AUC** of 0.861 can be appreciated.

Figure 4.8 represent the histograms of the matrixes used for the **ROC** curves, the **IoU** and **YOLO** accuracy matrixes. The **IoU** matrix histogram represents the number of values of the matrix included in each of the bins. As it can be seen, the majority of the bounding boxes assigned by **YOLO** had an **IoU** between 0.7 and 1, this means that **YOLO** performance precision in the **HARS** dataset is very good eventhough the difficulties of it. The next histogram, corresponding with the **YOLO** accuracy of the detections made, shows that **YOLO** was, in the majority of the detections made, between 90% and 100% sure that the detection performed belonged to the *person* class, this means that **YOLO** accuracy along the **HARS** dataset was very good.

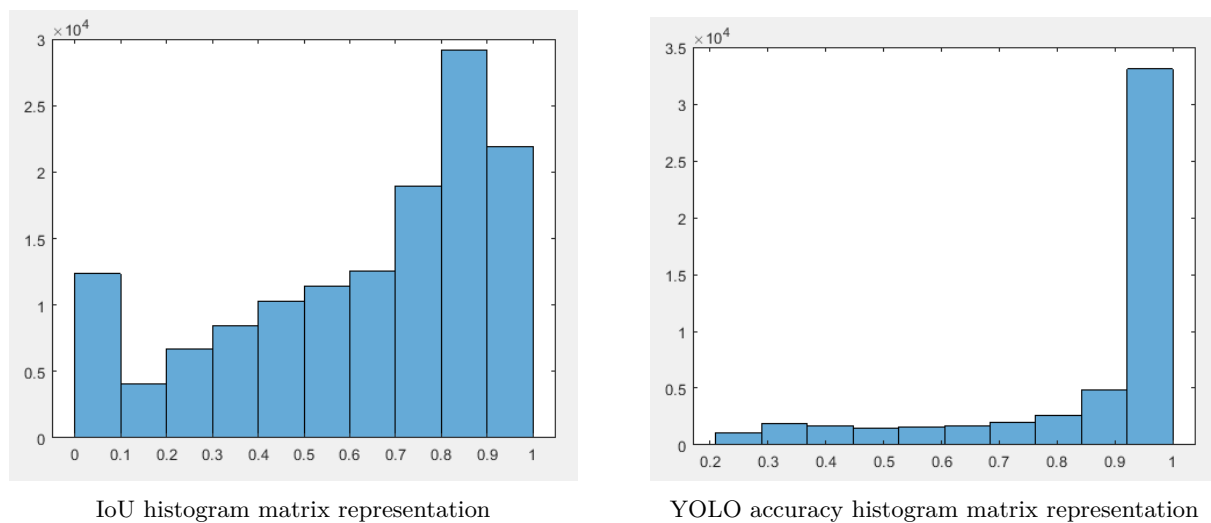


Figure 4.8: Histogram representations of IoU and YOLO accuracy

It is important to highlight that the results displayed correspond to the **RGB** analysis of the dataset as the **IR** videos display poor results.

As previously seen, the distribution by scenarios of the dataset is not balanced, as each of the different scenarios in the dataset include different number of videos. Consequently **YOLO** performs differently depending on the scenario, taking this into consideration the conclusions are separated and differentiated by the different scenarios that conform the dataset.

### 4.3.1 RGB vs IR

As explained in the section describing the characteristics of the dataset, it is composed of **RGB** videos and **IR** videos but, **YOLO** performs differently in these two types of videos working noticeable better in the **RGB** videos. In figure 4.9, there can be seen two frames from the same video recorded with **RGB** and **IR** cameras. As it can be seen, in the **IR** frame **YOLO** bounding boxes for two persons are missing.

This underperformance by **YOLO** in the **IR** recorded videos can be seen better in the **ROC** curves looking at the **AUC** parameter. As explained at the beginning of the chapter the bigger this parameter is the better performance by **YOLO**. In figure 4.10 the different **ROC** curves for **IoU** and **YOLO** accuracy for **RGB** an **IR** videos are shown. The **AUC** of the **IoU** and **YOLO** accuracy in both cases is superior in the **RGB** camera than in the **IR** camera, this means that **YOLO** performs better in the **RGB** videos of

the dataset. Apart from the FOV difference between the two cameras used, due to the lighting YOLO performance deteriorates in comparison with the RGB video.

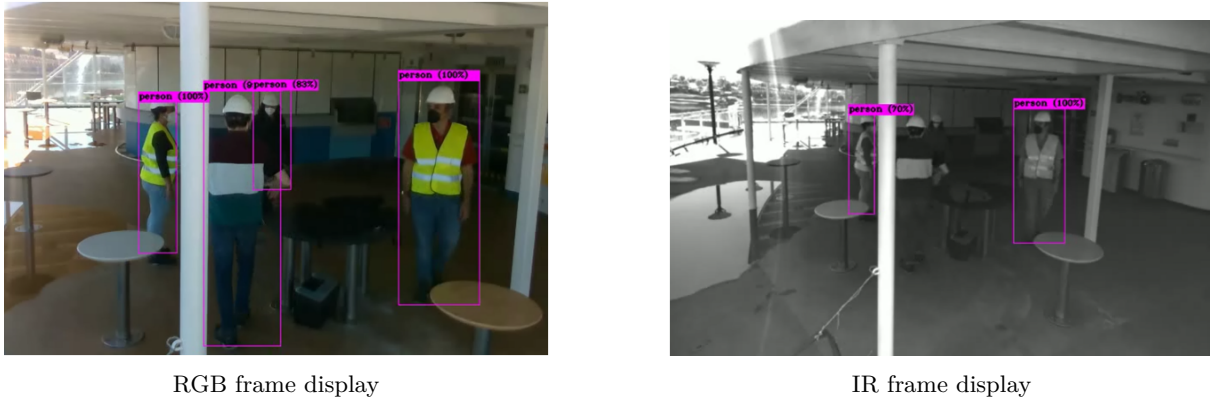


Figure 4.9: Difference between YOLO detection bounding boxes depending on the video recording setup

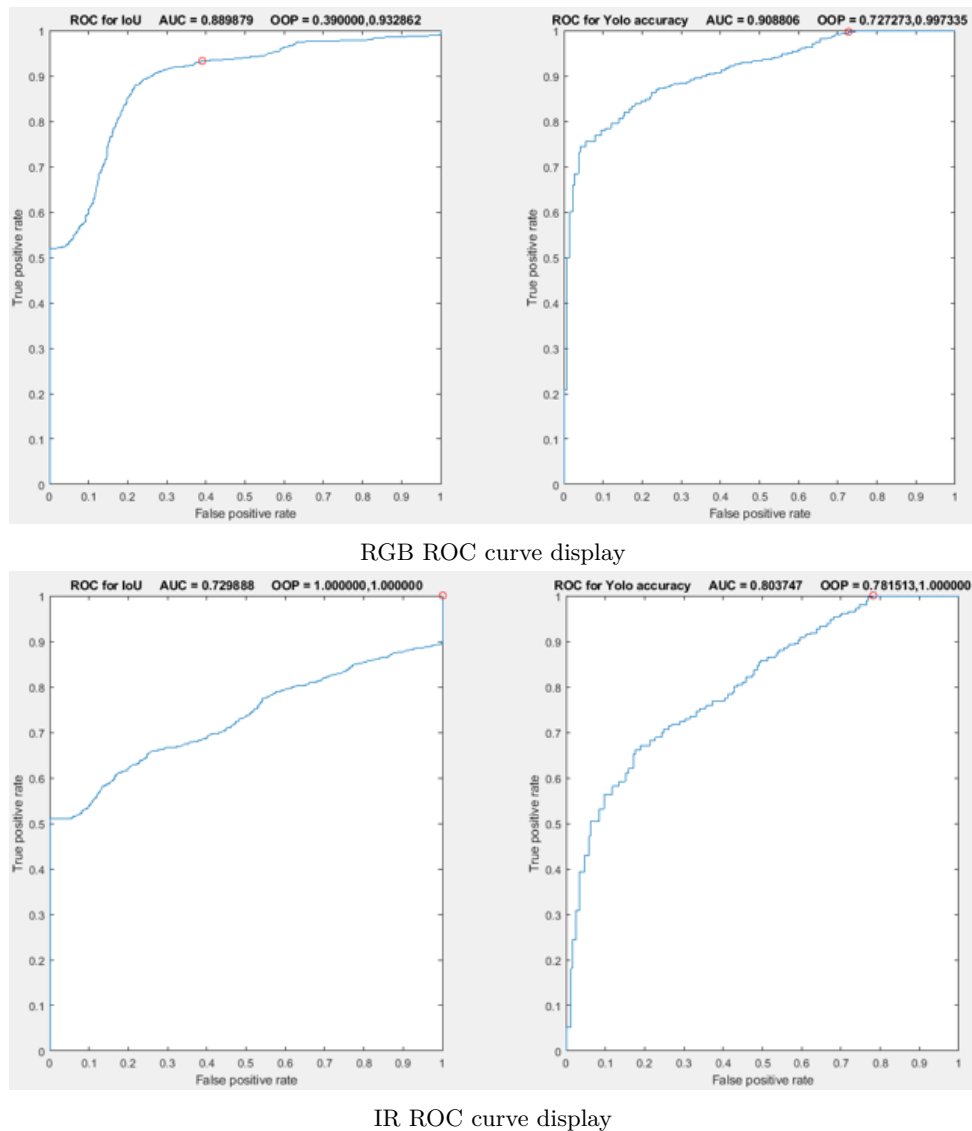


Figure 4.10: Difference between YOLO IoU and accuracy ROC curves



### 4.3.2 Lighting conditions

The previously seen scenarios contain the best lighting in the whole dataset, so to deepen this lighting criteria relationship with the YOLO performance the results obtained from one of the worst lighting scenarios in the dataset are presented. As it can be seen in image 4.11 the only person that YOLO detects in the IR video is the one below the light in the corridor, consequently the most illuminated one, and does not detect the other 3 people in the corridor that are being detected in the RGB video.



Figure 4.11: Difference between YOLO detection bounding boxes depending on the video recording setup

As in the previous example, this can be seen in the IoU and YOLO accuracy ROC curves presented in figures 4.12 and 4.13.

In the case of the RGB footage recorded these types of scenarios with low lighting do relatively well and in the case of low-quality lighting in the IR videos of this scenario there can be seen see a very poor IoU ROC curve and an ROC curve accuracy shaped like a random classifier.

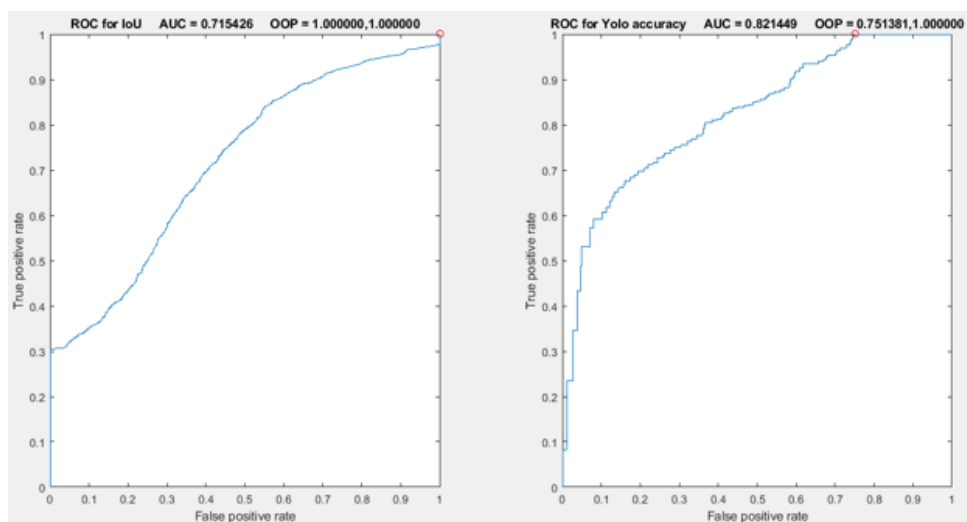


Figure 4.12: RGB ROC curve display

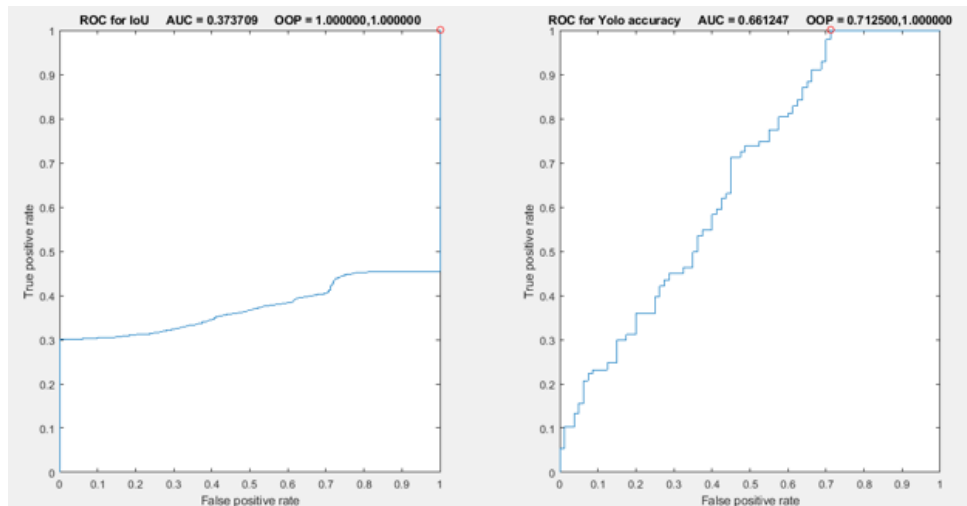


Figure 4.13: IR ROC curve display

### 4.3.3 Depth and occlusions

In this section of the chapter the performance of **YOLO** for two different scenarios in the dataset is analyzed to obtain in which scenarios does it work better depending on the occlusion of the persons in the scenario and link these conclusions obtained for each of the scenarios with its particularities with the limitations of **YOLO** seen in the chapter 2.

1. **Common Area scenario:** this scenario of the dataset consists in a long hallway with mirror on its sides. The fact that it has mirrors on its sides makes **YOLO** perform incorrect detections as it does not understand the concept of reflection, so **YOLO** detects reflections as persons. Figure 4.14 shows an example of the incorrect detections performed in the mirror. However these incorrect detections caused by the mirror are discarded in the processing of the data before evaluation as the bounding boxes with more **IoU** are considered but they are also considered false positives.



Figure 4.14: Wrong people detections made by YOLO due to the mirror

Other significant trait of this scenario is the hallway. The most remarkable feature of this scenario is depth, depth implies that as the persons travel along the hallway are reduced in size and becoming closer other people, resulting in an agglomeration of them. This results in a difficult scenario for an untrained YOLO model in this type of dataset, what causes YOLO to perform poorly. However, as seen in the figure 4.15 YOLO accuracy ROC curve, the detections that YOLO has performed had a percentage of confidence assigned with a high value, in most cases due to the good lighting of the hallway.

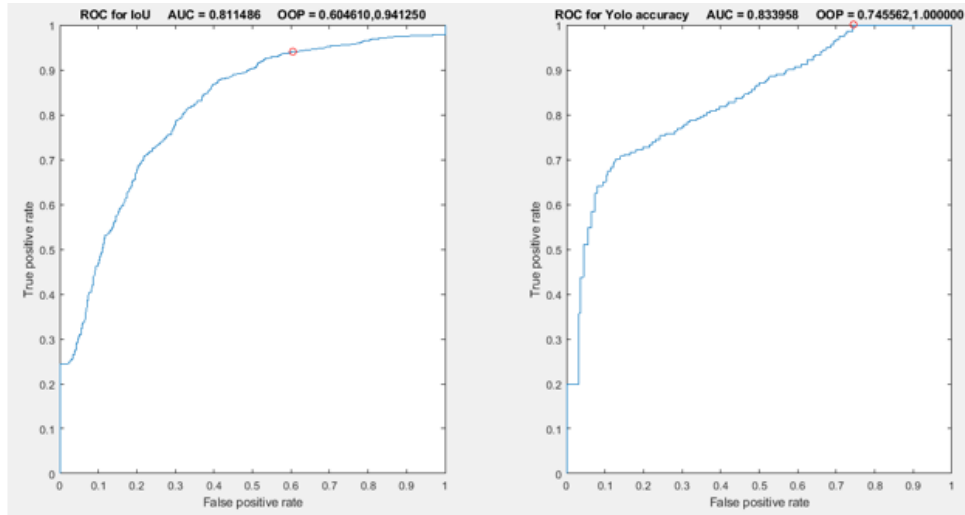


Figure 4.15: ROC curve corresponding to this scenario

2. **Corridor Cabins scenario:** this scenario consists on a narrower hallway with worse lighting than before. Figure 4.16 displays the scenario. Following up the explanation of the previous scenario and considering the theory seen in the state of art chapter that YOLO struggles to differentiate two different people when they are overlapping, so the next conclusions can be obtained, as this is a narrower case as the previously seen the overlap between the persons is greater since beginning to end of the hallway.

This fact can be observed in the IoU ROC curves shown in figure 4.17, as the AUC has decreased from 0.81 (from previous case) to 0.71 (in this case). However, the lighting factor has only affected slightly in the YOLO accuracy ROC curve being a decrement from 0.83 (from the previous case) to 0.82 (in this case). Just by analyzing these two similar scenarios the theory seen in chapter 2 where it is explained that YOLO performance is decreased when objects intersect more is corroborated, and it can also be seen that its accuracy at classification depends on lighting.



Figure 4.16: Corridor cabins scenario showcase

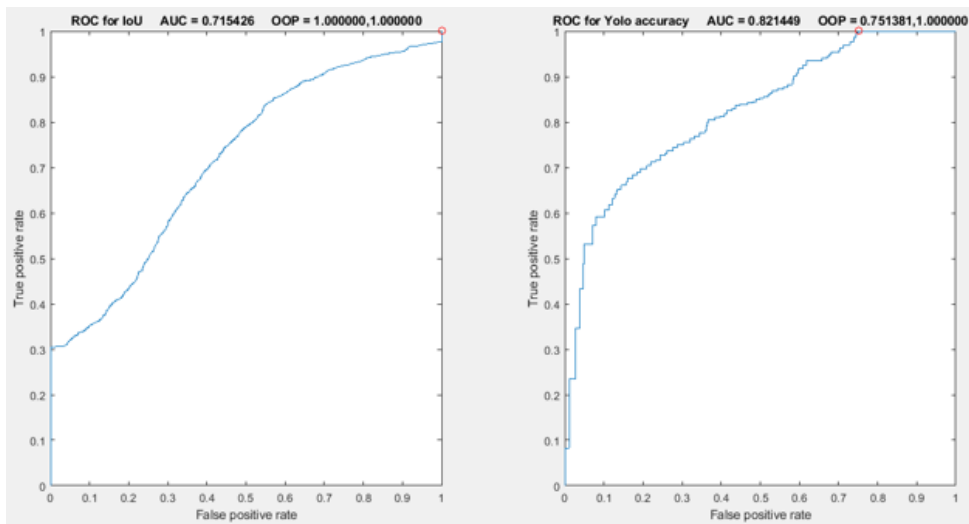


Figure 4.17: Corridor cabins ROC curve display

## Chapter 5

# Conclusions and future issues of research

### 5.1 Conclusions

In this work the [HARS](#) dataset has been labeled using a labeling tool developed by [GEINTRA](#) research group. This labeling tool has been modified accordingly to the labeling necessities of this work. Moreover, there has been carried out an exhaustive experimental evaluation using [YOLO](#) in order to obtain decisive conclusions that show if the [HARS](#) dataset allows neural networks to work with.

During the development of the [TFG](#), there has been carried out a search of different used datasets for people and action recognition and different proposals in the literature in search of the neural network that provides the best features and performance for the [HARS](#) dataset.

The chosen neural network, [YOLO](#), has been implemented in Google Colab using the weight obtained training with [COCO](#) dataset.

Using this neural network, there has been performed an exhaustive evaluation based on the output files provided from the Google Colab implementation and the files product of the labeling tool, for this experimental evaluation [ROC](#) curves and histograms have been used.

For the [HARS](#) dataset set of [RGB](#) videos average values of 70% and 100% predominate the [IoU](#) evaluation and values between 90% and 100% have been obtained in the [YOLO](#) detection accuracy, due to the poor performance of [YOLO](#) in the set of [IR](#) videos included the results of this set are discarded as they do not provide useful information to obtain conclusions.

In summary the [HARS](#) dataset allows neural networks to work with it with [RGB](#) as it has been seen that [YOLO](#) is able to detect persons in a precise and reliable way, also ground truth files of the [HARS](#) dataset have been obtained which serves to *PALAEMON: A holistic passenger ship evacuation and rescue ecosystem* project (H2020-PALAEMON-814962), therefore the objectives of the [TFG](#) have been fulfilled.

### 5.2 Future issues

As shown in this work, it is seen that having a labeled dataset allows for a lot of evaluation processes using different neural networks as well as a training process for them. For that reason a great number

of improvements can be done to it some of them were initially planned but due to the lack of time there were finally not implemented. These improvements are listed below.

1. **YOLOv3 training**, using MATLAB and the labeling process ground truth files, the required file format for training YOLOv3 can be obtained leading to a training of YOLOv3, this would result in a suitable model for the type of scenarios included in the HARS dataset.
2. **YOLOv3 evaluation**, following the previous point, the training of YOLOv3 instead of using the pretrained weights obtained from COCO would lead to an increment in the neural network precision and reliability which would result in better performance in the RGB and IR video sets of the HARS dataset.
3. **Different types of neural networks training and evaluation**, during the process of this TFG the only thing taken into account was person detection but, as individual and group actions were labeled, a neural network capable of detecting individual and group actions could be trained and evaluated and if given suitable results it could be implemented in the real world and directly applied to *PALAEMON: A holistic passenger ship evacuation and rescue ecosystem* project (H2020-PALAEMON-814962).

# Chapter 6

## Tools and resources

In this chapter the needed hardware and software requirements in order to carry out the project are presented.

### 6.1 Hardware requirements

1. **Processor** Any Intel or AMD x86-64 processor.
2. **RAM** 4 GB.
3. **Hard-disk space** 5-8 GB for MATLAB installation.
4. **Graphics** OpenGL 3.3 with 1GB GPU memory.

### 6.2 Software requirements

1. Windows 10.
2. MATLAB 2021a.
3. Computer Vision Toolbox.
4. Statistics and Machine learning Toolbox.
5. TexStudio.
6. MikTeX.
7. Google Colab.





# Chapter 7

## Budget

In this chapter the different costs needed for the development of this project are summarized. The cost of the hardware and software used as well as the cost of employee time is analyzed.

### 7.1 Hardware resources

Concept	Price	Quantity	Total
Intel RealSense Depth Camera D435	284.26€	1	284,26€
Computer	600€	1	600€
<b>Total</b>			884.26€

Table 7.1: Price of the different hardware used in the [TFG](#)

### 7.2 Software resources

Concept	Price	Quantity	Total
Matlab License	2043.14€	1	2043.14€
Windows 10	132.84€	1	132.84€
GitHub	0€	1	0€
Google Colab	0€	1	0€
TextStudio	0€	1	0€
MikTex	0€	1	0€
<b>Total</b>			2175.98€

Table 7.2: Price of the different software used in the [TFG](#)

### 7.3 Human resources

Concept	Price	Quantity	Total
Development	60€/hour	250	15000€
Typescript	15€/hour	150	2250€
<b>Total</b>			17250€

Table 7.3: Price of human resources needed in the [TFG](#)

### 7.4 Total cost budget

Concept	Cost
Hardware cost	884.26€
Software cost	2175.98€
Human cost	17250€
<b>Total</b>	20310.24€

Table 7.4: Total cost budget of the [TFG](#)

# Bibliography

- [1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [2] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov *et al.*, “The open images dataset v4,” *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.
- [3] “The top image datasets and their challenges,” <https://zbigatron.com/the-top-image-datasets/>.
- [4] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.
- [5] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, “Ntu rgb+ d: A large scale dataset for 3d human activity analysis,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1010–1019.
- [6] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, “The kinetics human action video dataset,” *arXiv preprint arXiv:1705.06950*, 2017.
- [7] “Collective activity dataset,” <https://cvgl.stanford.edu/projects/collective/collectiveActivity.html>.
- [8] R. J. Sethi, “Towards defining groups and crowds in video using the atomic group actions dataset,” in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 2925–2929.
- [9] X. Cui, Q. Liu, M. Gao, and D. N. Metaxas, “Abnormal detection using interaction energy potentials,” in *CVPR 2011*. IEEE, 2011, pp. 3161–3167.
- [10] “Artificial neuron model,” [https://www.researchgate.net/figure/Artificial-Neuron-model\\_fig4\\_277774116](https://www.researchgate.net/figure/Artificial-Neuron-model_fig4_277774116).
- [11] “Schematic diagram of ann,” [https://www.researchgate.net/figure/Schematic-diagram-of-ANN-with-input-hidden-and-output-layer-Input-for-this-case-is\\_fig2\\_36428019](https://www.researchgate.net/figure/Schematic-diagram-of-ANN-with-input-hidden-and-output-layer-Input-for-this-case-is_fig2_36428019).
- [12] “Visualize cnn with keras,” <https://www.kaggle.com/code/amarjeet007/visualize-cnn-with-keras/notebook>.
- [13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

- [14] “Yolo, yolov2, and yolov3: All you want to know,” <https://amrokamal-47691.medium.com/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899>.
- [15] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, “Mini-yolov3: real-time object detector for embedded applications,” *Ieee Access*, vol. 7, pp. 133 529–133 538, 2019.
- [16] “YOLOv3: Real-time object detection algorithm (whats new?),” <https://viso.ai/deep-learning/yolov3-overview>.
- [17] B. Xu and Z. Chen, “Multi-level fusion based 3d object detection from monocular images,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2345–2353.
- [18] P. Adarsh, P. Rathi, and M. Kumar, “Yolo v3-tiny: Object detection and recognition using one stage improved model,” in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2020, pp. 687–694.
- [19] D. Liciotti, M. Paolanti, E. Frontoni, and P. Zingaretti, “People detection and tracking from an rgb-d camera in top-view configuration: review of challenges and applications,” in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 207–218.
- [20] C. A. Luna, C. Losada-Gutierrez, D. Fuentes-Jimenez, A. Fernandez-Rincon, M. Mazo, and J. Macias-Guarasa, “Robust people detection using depth information from an overhead time-of-flight camera,” *Expert Systems with Applications*, vol. 71, pp. 240–256, 2017.
- [21] D. Fuentes-Jimenez, R. Martin-Lopez, C. Losada-Gutierrez, D. Casillas-Perez, J. Macias-Guarasa, C. A. Luna, and D. Pizarro, “Dpdnet: A robust people detector using deep learning with an overhead depth camera,” *Expert Systems with Applications*, vol. 146, p. 113168, 2020.
- [22] G. Yao, T. Lei, and J. Zhong, “A review of convolutional-neural-network-based action recognition,” *Pattern Recognition Letters*, vol. 118, pp. 14–22, 2019.
- [23] M. Majd and R. Safabakhsh, “Correlational convolutional lstm for human action recognition,” *Neurocomputing*, vol. 396, pp. 224–229, 2020.
- [24] A. Sánchez-Caballero, S. de López-Diz, D. Fuentes-Jimenez, C. Losada-Gutiérrez, M. Marrón-Romera, D. Casillas-Pérez, and M. I. Sarker, “3dfcnn: real-time action recognition using 3d deep neural networks with raw depth information,” *Multimedia Tools and Applications*, Mar 2022. [Online]. Available: <https://doi.org/10.1007/s11042-022-12091-z>
- [25] D. Tsipras, S. Santurkar, L. Engstrom, A. Ilyas, and A. Madry, “From imagenet to image classification: Contextualizing progress on benchmarks,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 9625–9635.
- [26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [27] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, vol. 1. IEEE, 2005, pp. 886–893.

- [28] R. Stewart, M. Andriluka, and A. Y. Ng, “End-to-end people detection in crowded scenes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2325–2333.
- [29] D. Fuentes-Jimenez, C. Losada-Gutierrez, D. Casillas-Perez, J. Macias-Guarasa, D. Pizarro, R. Martin-Lopez, and C. A. Luna, “Towards dense people detection with deep learning and depth images,” *Engineering Applications of Artificial Intelligence*, vol. 106, p. 104484, 2021.
- [30] T. Bagautdinov, F. Fleuret, and P. Fua, “Probability occupancy maps for occluded depth images,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2829–2837.
- [31] T. Ophoff, K. Van Beeck, and T. Goedemé, “Improving real-time pedestrian detectors with rgb+depth fusion,” in *AVSS Workshop - MSS*. Auckland, New Zealand: IEEE, 2018.
- [32] L. Fei-Fei, J. Deng, and K. Li, “Imagenet: Constructing a large-scale image database,” *Journal of vision*, vol. 9, no. 8, pp. 1037–1037, 2009.
- [33] G. A. Miller, “Wordnet: a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [34] “Wordnet: A lexical taxonomy of english words,” <https://towardsdatascience.com/%EF%B8%8Fwordnet-a-lexical-taxonomy-of-english-words-4373b541cfff>.
- [35] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov, T. Duerig, and V. Ferrari, “The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale,” *IJCV*, 2020.
- [36] “JFT-300 dataset [last accessed 09/06/2022],” <https://paperswithcode.com/dataset/jft-300m>.
- [37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [38] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” CRCV-TR-12-01. UCF Center for Research in Computer Vision, Tech. Rep., 2012.
- [39] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, “The kinetics human action video dataset,” *arXiv preprint arXiv:1705.06950*, 2017.
- [40] W. Choi, K. Shahid, and S. Savarese, “What are they doing?: Collective activity classification using spatio-temporal relationship among people,” in *2009 IEEE 12th international conference on computer vision workshops, ICCV Workshops*. IEEE, 2009, pp. 1282–1289.
- [41] “Collective activity,” <https://paperswithcode.com/dataset/collective-activity>.
- [42] R. J. Sethi, H. Jo, and Y. Gil, “Structured analysis of the isi atomic pair actions dataset using workflows,” *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2023–2032, 2013.
- [43] R. J. Sethi, “Towards defining groups and crowds in video using the atomic group actions dataset,” in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 2925–2929.
- [44] “Activation functions in neural networks explained,” <https://www.mygreatlearning.com/blog/activation-functions/>.

- [45] “Artificial neuron,” <https://intellect.ml/artificial-neuron-200>.
- [46] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
- [47] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “Dssd: Deconvolutional single shot detector,” *arXiv preprint arXiv:1701.06659*, 2017.
- [48] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” *Advances in neural information processing systems*, vol. 29, 2016.
- [49] “YOLO you only look once, real time object detection explained,” <https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006>.
- [50] “Intel d-435,” <https://www.intelrealsense.com/depth-camera-d435/> [Last accessed 03-Dec-2021].
- [51] “Intel® realsense? d435,” <https://www.intelrealsense.com/depth-camera-d435/>.
- [52] “Google colab webpage,” <https://colab.research.google.com/> [Last accessed 09/06/2022].
- [53] “Alexeyab darknet,” <https://github.com/AlexeyAB/darknet>.
- [54] “Real-time object detection using yolo upon google colab in 5 minutes,” <https://medium.com/@artinte7/real-time-object-detection-using-yolo-upon-google-colab-in-5-minutes-fd65a4903df5>.
- [55] “Defining an optimal cut-point value in roc analysis: An alternative approach,” <https://pubmed.ncbi.nlm.nih.gov/28642804/>.



Universidad de Alcalá  
Escuela Politécnica Superior



ESCUELA POLITECNICA  
SUPERIOR



Universidad  
de Alcalá