UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: TRACKING AND RECOGNITION OF MOVING HUMAN SILHOUETTES USING REINFORCEMENT LEARNING

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniera en Tecnologías de la Información

Autor:

Amanda Ulloa Uprety

Tutor:

PhD. Chang Tortolero Oscar Guillermo

Urcuquí, Diciembre, 2021

Urcuquí, 21 de diciembre de 2021

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2021-00040-AD

A los 21 días del mes de diciembre de 2021, a las 14:30 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. ANTON CASTRO , FRANCESC , Ph.D.
Miembro No Tutor	Dr. LEIVA , HUGO , Ph.D.
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

El(la) señor(ita) estudiante **ULLOA UPRETY, AMANDA RAQUEL**, con cédula de identidad No. **1717628711**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **TRACKING AND RECOGNITION OF MOVING HUMAN SILHOUETTES USING REINFORCEMENT LEARNING**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Presidente Tribunal De Defensa	Dr. ANTON CASTRO , FRANCESC , Ph.D.	9,0
Miembro Tribunal De Defensa	Dr. LEIVA , HUGO , Ph.D.	10,0
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	10,0

Lo que da un promedio de: **9.7 (Nueve punto Siete)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

Certifico que *en cumplimiento del Decreto Ejecutivo 1017 de 16 de marzo de 2020, la defensa de trabajo de titulación (o examen de grado modalidad teórico práctica) se realizó vía virtual, por lo que las firmas de los miembros del Tribunal de Defensa de Grado, constan en forma digital.*

Firmado electrónicamente por:
AMANDA RAQUEL ULLOA UPRETY

ULLOA UPRETY, AMANDA RAQUEL
Estudiante

FRANCESC ANTON CASTRO
Digitally signed by FRANCESC ANTON CASTRO
Date: 2021.12.23 15:12:40 +01'00'

Dr. ANTON CASTRO , FRANCESC , Ph.D.
Presidente Tribunal de Defensa

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Tutor

Dr. LEIVA , HUGO , Ph.D.
Miembro No Tutor

TATIANA BEATRIZ TORRES MONTALVAN
Firmado digitalmente por TATIANA BEATRIZ TORRES MONTALVAN
Fecha: 2021.12.22 12:41:43 -05'00'

TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

Autoría

Yo, **Amanda Ulloa Uprety**, con cédula de identidad **171762871-1**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Noviembre del 2021.

Firmado electrónicamente por:
AMANDA RAQUEL
ULLOA UPRETY

Amanda Ulloa Uprety
CI: 171762871-1

Autorización de publicación

Yo, **Amanda Ulloa Uprety**, con cédula de identidad **171762871-1**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Noviembre del 2021.

Firmado electrónicamente por:
AMANDA RAQUEL ULLOA UPRETY

Amanda Ulloa Uprety
CI: 171762871-1

Resumen

La detección de humanos en videos en vivo ha sido un campo bastante explorado debido a su amplia usabilidad en seguridad, automatización de actividades, asistentes de conducción, entre otras. Pero de igual manera es un tema bastante desafiante, ya que las imágenes siempre varían en diferentes aspectos como son la iluminación, color, calidad de imagen, también existe la variedad de poses, colores, tamaños y velocidades de las personas, esto hace que la detección sea una tarea complicada y que requiera mucho tiempo. Otros desafíos pueden ser el consumo de recursos informáticos, ya que procesar una gran cantidad de información puede saturar el sistema. El procesamiento de imágenes o de un video, los algoritmos de inteligencia artificial, entrenamientos y otros procesos necesarios para la correcta detección en la imagen, pueden ser una tarea exigente para nuestra máquina.
En este proyecto de tesis, proponemos un detector de humanos capaz de operar en videos en vivo utilizando redes neuronales, basadas principalmente en el algoritmo de detección de objetos YOLO. El sistema propuesto usa redes neuronales convolucionales y optimiza técnicas innovadoras del algoritmo YOLO para que presente un resultado de reconocimiento humano bastante preciso pero al mismo tiempo una menor demanda en nuestra computadora, reduciendo así el consumo de recursos, reduciendo tiempo de procesamiento de imágenes y el tiempo de reconocimiento.

Palabras Clave: Redes neuronales convolucionales, reconocimiento humano video, detección de objetos, recursos informáticos.

Abstract

The detection of humans in live videos has been a quite explored field due to its wide usability in security, activities automation, driving assistants, medical helpers etc. At the same time, it is quite a challenging subject, since images always vary in many aspects such as lighting, color, quality, poses, overlapping, sizes and speeds of people, and thus, it makes detection a complicated task and a time and resource consuming process. In general image processing requires large computer resources and advanced tools in artificial intelligence and machine learning. This thesis work proposes a human detector that can operate in live video by using neural networks based mainly on the YOLO object detection algorithm. The constructed software system uses convolutional neural networks and focuses on in optimizing some of the YOLO algorithm techniques so that its accuracy in human recognition is improved while lowering the demand of computer power and reducing the recognition time.

Keywords: Convolutional Neural Networks, human recognition, video, object detection, computer resources.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In the last decade, computer vision has become a wider and important field, with many more practical applications, supported with new technologies and the increase of people using digital images. The main objective of computer vision is to extract varied and useful information from images. [1] Whether in videos or photographs, every day we have contact with them, and as the relevance of computer vision increases, the need and applications for detecting objects, animals and people in images also increase. Detecting and tracking humans in particular has a wide variety of uses, such as in surveillance videos, security, self-driving cars, people counting, human-robot interaction and immersive interactive entertainment, smart homes, assistance for people that live alone, and searching for people for military applications. [2]

The wide range of applications and underlying intellectual challenges of the detection has attracted many researchers.

In security systems, human detection is essential, because its purpose is mainly to identify and monitor people in places with a high attendance of people, such as shopping centers, streets, airports and stations. In general, a full body monitoring is required, which may later be accompanied by pose recognition or facial recognition. In the case of car automation and driving assistance, the aim is for the car system is to detect, dodge or warn about the presence of objects or pedestrians. [3]

1.2 Problem statement

In the same way that object detection is important, it is challenging, specially referring to human detection and tracking. This is because there are so many different appearance, colors, sizes and countless poses that a person can be in.[2] Furthermore, people wear different accessories and clothes, also they could be carrying babies, animals or objects like bags, suitcases or walking sticks. Another problem is the overlapping of objects and people, it is challenging for a machine to detect humans when them are close to each other or when only a portion of they is visible. Another problem is the image quality of the camera and its speed of image capture. Also, there are external factors that can affect image quality, such as lighting, weather and noise. All these factors make it difficult to process the data and therefore the task of detecting humans [4].

An additional problem is the consumption of computer resources, since processing a large amount of information can demand an excessive use of the memory of our computer. In object recognition, the use of machine learning and artificial intelligence algorithms for processing images or videos, is essential. Training them or training a convolutional neural network is usually quite demanding with the use of the computer's memory and power. [5]

1.3 Objectives

1.3.1 General Objective

To develop a method based on artificial intelligence that is capable of recognizing human bodies in live videos and keep track of them using convolutional neural networks.

1.3.2 Specific Objectives

- To design, train and implement a new convolutional neural network for video processing based in YOLO algorithm, reducing the demand for computing resources.
- To determine the performance of the proposed algorithm in terms of precision, accuracy, training and execution time.
- To implement a system for human bodies recognition and tracking in videos, using the trained model.

Chapter 2

Theoretical Framework

This chapter will present all the concepts necessary to understand the development of this project. It will cover computer vision fundamentals, general concepts about neural networks and convolutional neural networks, object detection and the algorithm will be introduced.

2.1 Computer Vision

2.1.1 Definition

Computer vision is a wide and interdisciplinary field of study that seeks to develop techniques to help computers computers to extract meaningful information from digital images and videos, and perform tasks on that information. The aim is to imitate the perception of human sight, or to give the computer the power to see and understand images using machine learning. Computer vision works in a similar way to human vision, except humans have an advantage. The human eye has the advantage of the lifetime of the context to train how to distinguish objects, their distance, size, whether they are moving and if there is something rare in an image.[6]

Using computer vision, machines can be trained to perform these kind of functions, but it has to do it using cameras, data, and artificial intelligence rather than retinas, optic nerves, and a visual cortex. Also, it must process information and pass it to a system much faster, because a system, for example, trained for the production sector can analyze thousands of products or processes per minute, noticing imperceptible defects or problems,

it can quickly exceed human capabilities.[7]

2.1.2 Examples and Applications

If we classify computer vision in its types and applications, there is a great variety, below we will present only some examples of uses and applications since they seem relevant to this research.

Image Classification

It obtains an image and classifies it, precisely, it is able to predict whether the received image corresponds to a specific class, for example a dog, an apple, etc.

Object detection

It uses the classification of images to identify classes and detect if one or more objects of the class or classes of interest are found in an image or video, it is able to analyze the entire image and locate the exact place where the object is.[8] From here you can also track objects in sequence images or live video.[9]

Security

We all know video surveillance, usually, it consists of cameras that are implemented in public places such as shopping centers, busy streets, banks, among others. The goal is to have this information stored and to be able to use it at some point when it is needed. In general, these sites have a large number of cameras, therefore they have a lot of information, videos, and images, which makes it difficult for a person to manually analyze this information. Therefore, this task is entrusted to machines, computer vision is responsible for extracting the relevant information from these images using algorithms and artificial intelligence mechanisms that allow it to process a large amount of information quickly and accurately. The information of interest can be suspicious events, facial recognition, detection of specific motions, among others.[10]

Health

Image information is a key element for diagnosis in medicine because it represents 90% of all medical data. Many health diagnoses are based on image processing, X-rays, MRI,

and mammography, just to name a few. And image segmentation proved it worthwhile analyzing medical scans. For example, Computer Vision algorithms can detect diabetic retinopathy, the fastest-growing cause of blindness. Computer Vision can process images of the back of the eye and classify them according to the presence and severity of the disease.[11]

Self-driving cars

A car that drives itself or has an automatic driving assistant needs to identify people, traffic signs, other cars, animals, and cyclists quickly, almost instantly, since the event is happening at the moment [12]. It requires a reliable artificial vision that can plan routes, make correct movements and avoid collisions and accidents [13]. An object detector suitable for this task requires precision, speed, and low consumption of resources, remember that we are talking about a car and not a robust PC.

2.2 Neural Networks

A neural network(NN) or artificial neural network is a computational interconnected set of simple processing units or nodes whose functionality is based on the mechanism of biological neurons. The neurons that we all have in our brain are composed of dendrites, the soma, and the axon. The dendrites are responsible for capturing the nerve impulses emitted by other neurons. These impulses are processed in the soma and transmitted through the axon that emits a nerve impulse towards the neighboring neurons [14].

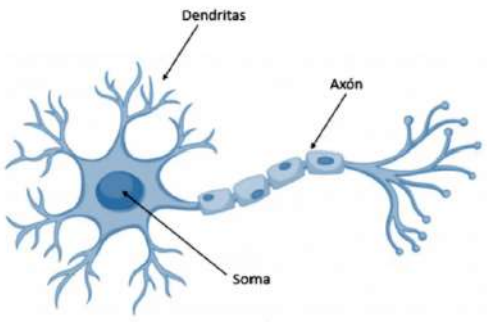

Figure 2.1: Biological Neuron

As mentioned, the functioning of networks is similar to that of the human brain. The networks receive a series of input values, simulating the ”nerve impulse”, and each of these

inputs reaches a node called a neuron. The neurons of the network are grouped in layers: which are sets of neurons whose inputs come from a previous layer (or from the input data in the case of the first layer) and whose outputs are the input from a later layer, forming a net. The value of each neuron is processed inside the cell by a trigger function that returns a value that is sent as the neuron's output. The new values are obtained to leave the neurons and continue their way through the network [14].
This operation can be seen in Figure 2.2, which shows a 4 layers network.

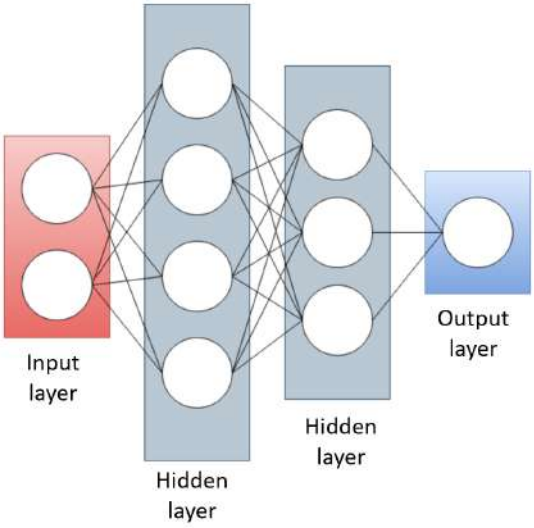

Figure 2.2: 4-layer NN structure

The first layer is known as the input layer since it receives the first data that feeds the neural network. The output of the last layer is the visible result of the network, which is why the last layer is known as the output layer. The layers that are between the input layer and the output layer are known as hidden layers since both the input and output values are unknown. A neural network is always composed of an input layer, an output layer (if there is only one layer in the neural network, the input layer matches the output layer) and can contain several hidden layers or none at all.[15]

2.2.1 Training a Neural Network

For a neural network to work, it needs to be trained. Training consists of adjusting each of the weights of the inputs of all the neurons that are part of the neural network so that the

responses of the output layer adjust as much as possible to the data that we know or that manages to extract the desired results. For this, what is done is to introduce training data into the network, depending on the result obtained, the weights of the neurons are modified according to the error obtained and depending on how much each neuron has contributed to said result. This method is known as Backpropagation. This method enables the network to learn, achieving a model capable of obtaining very successful results even with data different from those used for its training. [16]

2.2.2 Underfitting and Overfitting

Underfitting and overfitting are the two main generalization problems that occur in machine learning and degrade the performance of the machine learning models.

Overfitting

It occurs when our model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset. If we obtain Overfitting, we can solve it by obtaining a greater number of data, dividing our data in training, adjusting the parameters of our models, validation, and testing, using simpler models, or lowering the number of iterations if it is an iterative algorithms [17].

Underfitting

It occurs when our model is not able to learn enough from the training data, it is unable to identify or obtain correct results due to lack of sufficient training samples or very poor training, this reduces the accuracy and produces unreliable predictions. To solve or avoid Underfitting we can use more complex models, adjust the parameters of our models or increase the iterations in iterative algorithms [17].

2.2.3 Basic Concepts

In this section, some basic concepts necessary to better understand the definition of the model used in this project will be exposed, as well as terms that will be mentioned later.

Activation Function

It is a function that is used to get the output of the node. It is also known as the transfer function. It is used to determine the output of the neural network as yes or no. Maps the resulting values between 0 to 1 or -1 to 1. Some of the most used functions are Sigmoid, tanh, Softmax, ReLU, Leaky ReLU [18].

Learning Algorithm

The main objective of the ANN is to learn to solve problems by itself, this is achieved using a learning algorithm. There are several types of algorithms, but all have the objective of adjusting the network parameters in order to mimic the expected behavior and progressively reduce the error observed during training [19].

Weights

The weights are values that represent the connections between neurons. These weights represent the nerve impulses transmitted from one neuron to another. These values are not fixed, with neural network training, the weight values will change. After a network goes through the learning process, the weighting values represent all the information learned.

Loss Function

The loss function is an important component of the ANN. It is a method of evaluating how well the algorithm models the data set. [20] If the predictions are not accurate at all, the loss function will generate a large number. If they are fairly accurate, they will generate a small number. The loss is used to calculate the gradients and the gradients are used to update the weights of the neural network. As the algorithm adjusts to try to improve the model, its loss function will let us know whether or not it is improving. It helps us understand how the predicted value differs from the actual value. [17, 21]

2.3 Convolutional Neural Networks

A CNN is a type of ANN that uses convolutional layers to mimic the vision of the human eye to identify different characteristics in the inputs and make it possible to obtain useful information for the network. This type of network is generally used to recognize visual patterns such as characters, symbols, figures, and objects from pixel images. [10] For this,

CNNs contain several specialized hidden layers that have a hierarchy: this means that the first layers can detect basic objects such as lines, curves and are specialized until they reach deeper layers that recognize more specific and complex shapes such as a face. or the silhouette of an animal or a person. [22]

2.3.1 Convolutions

The convolutions consist on taking groups of neighboring pixels from the input image and operating mathematically with a small matrix called a kernel. That kernel goes through all the input neurons, from left to right, from up to down, and generates a new output matrix, which will ultimately be our new layer of hidden neurons.

2.3.2 Filter

It is a set of kernels, since not only one kernel is applied, but we will have many kernels. As the kernel moves, we get a filtered "new image". The kernel values or weights, as in traditional neural networks, have an initial value at the beginning of the training and later, during the process, these weights are updated until they approach the optimal conditions to make good predictions.

2.3.3 ReLu Activation Function

Rectified linear unit (ReLU), it is the most used activation function for this type of neural networks, it allows a faster and more efficient training by assigning negative values to zero and maintaining positive values, it consists of f (z) = max (0, z).[18]

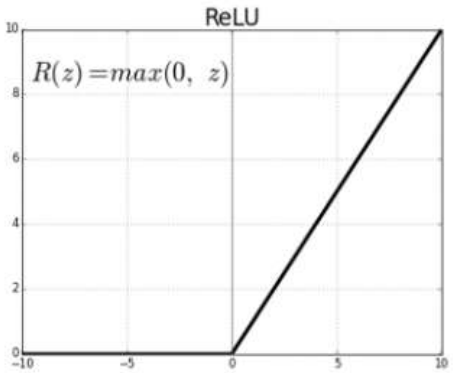

Figure 2.3: ReLu function

As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

2.3.4 Padding

Padding is the number of pixels added to an image when it is being processed, allowing for more accurate analysis. This padding adds some extra space around the image, which helps the kernel improve performance. When the image is going through the convolution process, the kernel scans every pixel and in this process, it scans some pixels several times (centered pixels) and some pixels a few times (edges).
When filters try to center pixels at the edge, the filter is out of range trying to convolve values outside of the image. Therefore, border values are ignored or this problem is overcome by using padding.[23]

2.3.5 Pooling

Pooling is an operation that is performed after each convolution. The goal of these layers is to reduce the dimension of the data. The pooling layer takes the output of the previous convolutional layer and applies a filter (following the same process as convolutional filters), this filter calculates the maximum or the average between the values captured by the filter. There are two main types of pooling layers: maximum pooling and average pooling.[7]

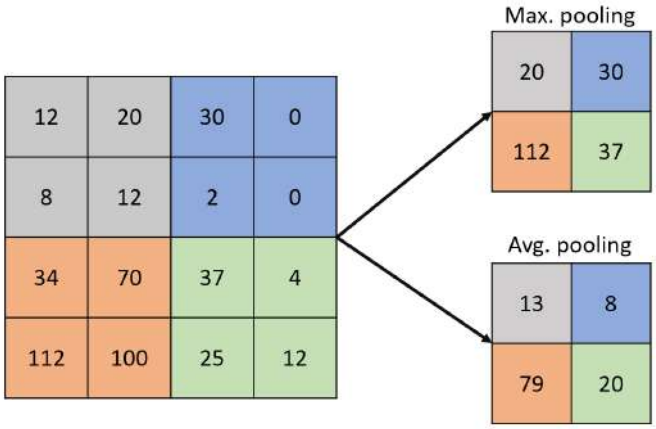

Figure 2.4: Max and average Pooling

2.3.6 Stride

Stride is the amount of pixels a filter moves over the input matrix. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time.

2.3.7 Convolutional Neural Networks for Object detection

CNNs are generally used in visual applications. Recognizing objects and locating them in images is one of the challenging problems in computer vision. In recent years, several models have been developed to solve this problem using CNN, these are known as object detectors. Below are some recent and common CNN-based models that were designed for detection and localization.[7]

R-CNN

Object detection consists of two independent tasks which are classification and location. R-CNN stands for Region-Based Convolutional Neural Network. The key concept is, as the name implies, the extraction of features by region using deep CNNs and learning independent linear classifiers for each object class.[24] A region proposal network takes an image and provides a set of rectangular object proposals that are later used for detection.[25, 7]

Fast R-CNN

The same author of the R-CNN optimized and solved some of the drawbacks of R-CNN to build a faster object detection algorithm and named it Fast R-CNN. The approach is similar to the R-CNN algorithm. But instead of submitting the region's proposals to CNN, the input image is sent to CNN to generate a convolutional feature map. From this map, the region of the proposals is identified and transformed into squares and, using a grouping layer, they are reshaped to a fixed size so that they can be fed to a fully connected layer. From the feature vector, a softmax layer is used to predict the class of the proposed region and also the offset values for the bounding box.[26, 7]

RPN and Faster R-CNN

The Regional Proposal Network simultaneously predicts object bounding boxes and objectivity scores at each position. RPN is a fully convolutional network, end-to-end trained to produce high-quality region proposals for object detection using Fast R-CNN. Combining

RPN with the Fast R-CNN object detector results in a new model called Faster R-CNN. In particular, in Faster R-CNN, the RPN shares its computation with Fast R-CNN by sharing the same convolutional layers, allowing joint training. [27, 7]

2.4 YOLO Algorithm

YOLO (You Only Look Once) is an algorithm that uses a convolutional neural network to detect objects in real-time. YOLO uses techniques that have been developed in recent years. And it implements most of the current ideas about computer vision like batch normalization, residual networks, data augmentation, among others, and does a good job in terms of accuracy.[28] In addition, the system applies a single neural network to the entire image, which makes the model quite fast. This network divides the image into regions and predicts multiple bounding boxes and the probability of detection of the training classes for each bounding box. Finally, it uses a non-maximum suppression method to eliminate multiple detections of the same object. As a result, the system prints the objects it detected, their confidence, and the execution time. Figure 2.5 shows an execution of the system on an input image. [29, 30]

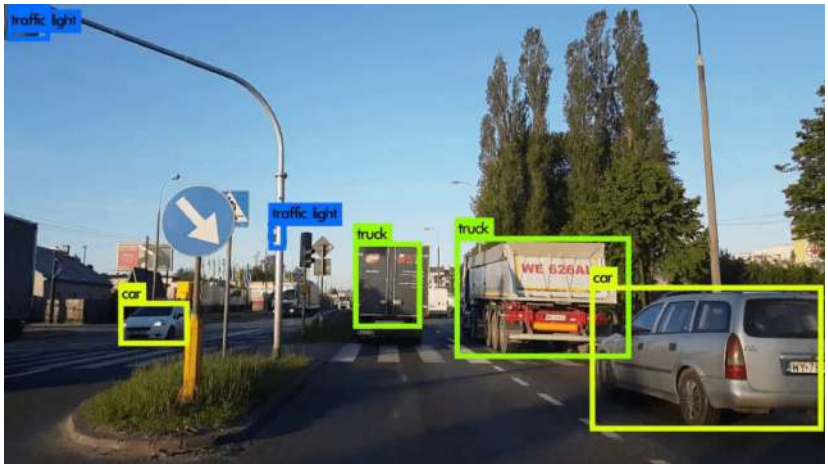

Figure 2.5: YOLO detection example

2.4.1 Techniques

The YOLO algorithm has evolved to a fourth version, but for this project YOLO, YOLOv2 and YOLOv3 were taken into account, those are the first 3 versions, since the latest version does not have much information or experimental content. The algorithm in general uses

quite a few modern techniques of computer vision and CNN, below is a comparison of the most relevant techniques that each version has, as well as a brief description of each one. [29, 30, 31]

Techniques	YOLO	YOLOv2	YOLOv3
Data Augmentation	X	X	X
Non-max suppresion	X	X	X
Bounding Box prediction	X	X	X
Batch Normalization		X	X
Anchor Boxes		X	X
Dimension Clusters		X	X
Location Prediction		X	X
Scale prediction			X
Residual Net			X

Table 2.1: Comparison of techniques of each version

Data Augmentation

Data augmentation is an effective technique to obtain more image variability based solely on the information from the training data we have to avoid overfitting. With this technique, the available images are duplicated and modified by flipping, zooming, rotating, cropping, among others; to create more data for the training process. The advantage of this is that the algorithms do it automatically and manual data editing and augmentation is avoided.[32, 33]

Non-max suppresion

Non-maximum suppression (NMS) is a key post-processing step in many computer vision applications, primarily in object detection. In this context, it is used to transform a smooth response map that triggers many imprecise object window hypotheses into, ideally, a single bounding box for each detected object.[34]

Bounding Box prediction

YOLO uses functions on the whole image to predict each bounding box. It also predicts all bounding boxes in all classes for an image simultaneously. The YOLO design enables end-to-end training and speeds in real-time while maintaining high average accuracy. The system divides the input image into $S \times S$ grids. If the center of an object falls on a cell in the grid, that cell is responsible for detecting that object. Each cell in the grid predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate the bounding box is.[29]
If no object exists in that cell, the confidence scores must be zero. Otherwise, the confidence score will be equal to the intersection over the union (IoU) between the predicted frame and the real information. So, confidence is defined by

$$Pr(Object) * IoU^{real}_{predict} \tag{2.1}$$

Each bounding box consists of 5 predictions: $x, y, w, h, and\ confidence$. The coordinates (x, y) represent the center of the box relative to the cell boundaries of the grid. The width and height are predicted relative to the entire image. And finally, the confidence prediction. Each cell in the grid also predicts C conditional class probabilities, Pr (Classi — Object). These probabilities are conditioned by the cell in the grid that contains an object. Only one set of class probabilities per cell is predicted, regardless of the number of cells B. When testing, the conditional class probabilities and the confidence predictions of the individual cells are multiplied, [30]

$$Pr(Class_i|Object) * Pr(Object) * IoU^{real}_{predict} = Pr(Class_i) * IoU^{real}_{predict} \tag{2.2}$$

which gives class-specific confidence scores for each box and encode both the probability of that class appearing in the box and if the predicted box fits the object.

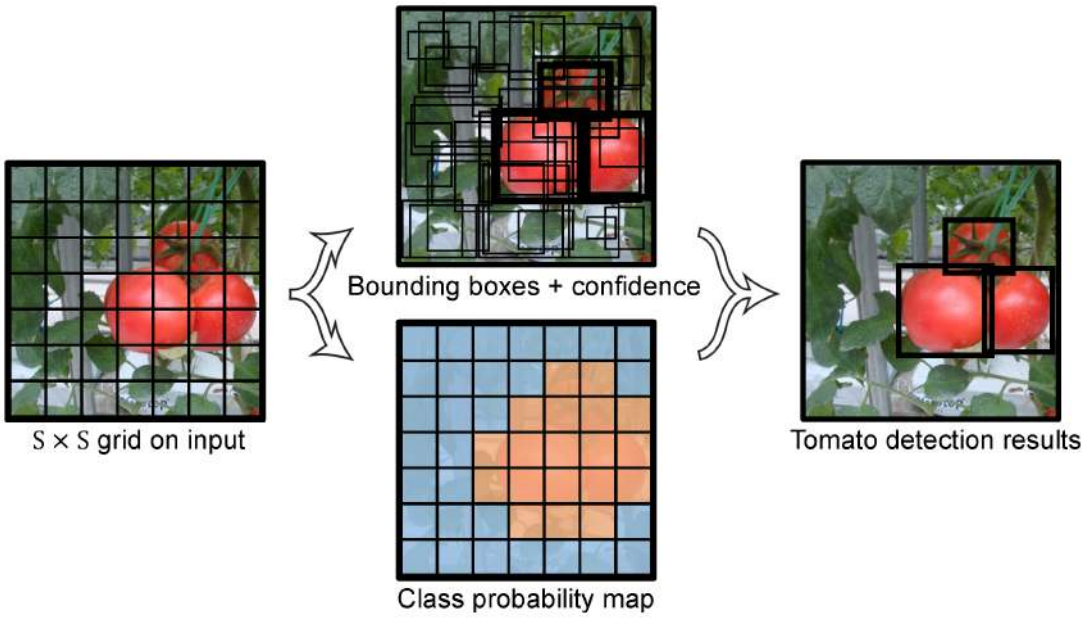

Figure 2.6: Bounding Box Prediction

Batch Normalization

Batch normalization is a technique to make neural networks faster and more stable by adding additional layers. The new layer performs the standardization and normalization operations of the inputs to one layer for each mini-batch. Since batch normalization acts as a regularizer, it has the effect of stabilizing the learning process and reducing the number of training epochs required, eliminating model attrition, and controlling overfitting.[35]

Anchor Boxes

The term anchor boxes refers to a predefined collection of boxes with widths and heights chosen to match the widths and heights of objects in a data set. The proposed anchor boxes cover the possible combination of object sizes that could be found in a dataset. Naturally, this should include different ratios and scales present in the data. It is common to select between 4 and 10 anchor boxes to use as proposals at various locations in the image.[30]

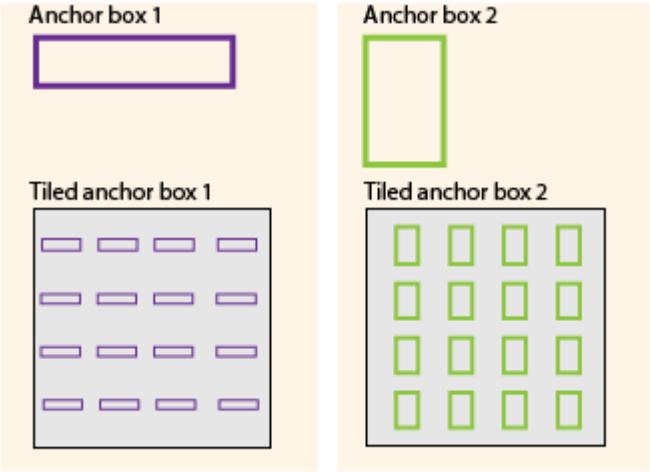

Figure 2.7: Predefined Anchor Boxes

Scale Prediction

It refers to the algorithm making predictions of the same image but at different scales. YOLOv3, for example, works with predictions at 3 different scales, this is achieved by downsamplings to the input image dimension by 32, 16, and 8. [30] This is done to detect objects of different sizes. Downsampling dimension by 32 gives us a small image, which offers a general view in which large objects are highlighted and observed, while downsampling dimension by 8 gives us a larger and more detailed image, which also contains more cells. and therefore the network is capable of detecting small objects. In Figure 2.8 we can see the representation of the 3 scales of an image of dimension 416 x 416.

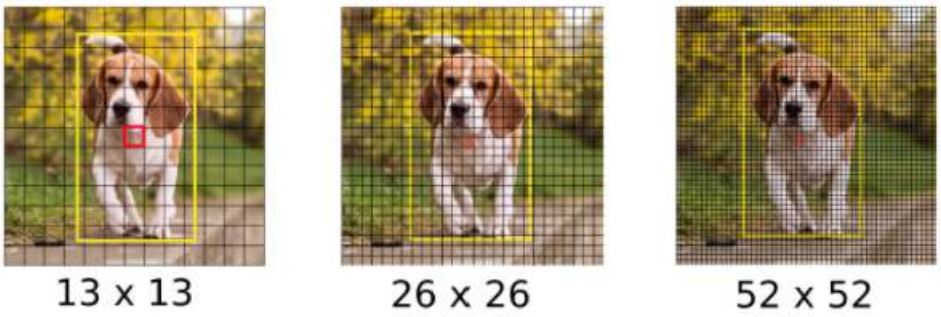

Figure 2.8: Image Downsamplig an Cell division for YOLOv3

Residual Network

A residual network (ResNet) is formed by stacking several residual blocks together. A residual block is a stack of layers configured in such a way that the output of one layer is taken and added to another layer deeper in the block. The non-linearity is applied after adding it along with the output of the corresponding layer in the main path. Deep ResNets are capable of forming an identity function that maps to a previous activation in the network when the activation of a specific layer tends to zero deeper in the network. This identity mapping created by these residual blocks is why adding additional layers does not affect the performance of a residual network. The performance improvement is achieved whenever the additional layers get meaningful information from the data. Whereas, the presence of the residual blocks avoids the loss of performance whenever the activations tend to disappear.[36]

2.5 YOLO Architecture

The network has 24 convolutional layers followed by 2 layers with full connections. 1 x 1 reduction layers are used followed by 3 x 3 convolutional layers. Figure 2.9 shows the general structure of the first version of the YOLO detector. This model has changed over time as YOLO introduced the different techniques described above.

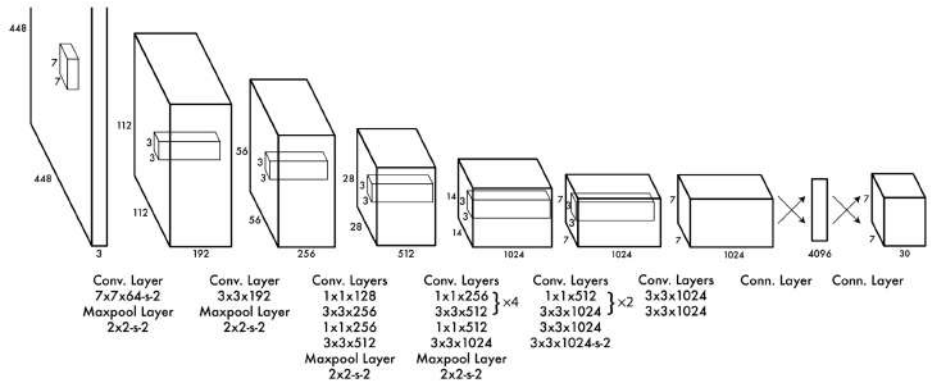

Figure 2.9: YOLO general Architecture

2.6 YOLOv3 Architecture

The development of this project was based mainly on the techniques and architecture of YOLOv3. Therefore, the architecture of this model is presented. Later, in section 3, we will explain the construction and development of the new model.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.10: YOLOv3 DarkNet

Chapter 3

Methodology

This chapter presents the process followed and all the details that were carried out to achieve the objectives of this project. Also, this section explains the main structure of the project.

First, an arduous investigation was carried out on the detection of objects and issues related to the detection of humans in images. Useful and specialized libraries in the use and development of convolutional neural networks (CNNs) such as pytorch were studied, as well as the opencv library, which was the main tool for image processing and video manipulation.

When researching about CNNs in object detection, the YOLO algorithm was found as the best option to fulfill the objective of this project, but when trying to implement it, due to its complexity, the result was a collapsed system, with a slow and erratic computer functioning, due to high memory requirements and a lot of CPU involvement in performing certain calculations. Also, the GPU was taken to the limit of its processing capacity.

The YOLO algorithm was studied in-depth and a YOLO-based algorithm was designed so that fewer calculations were required. In particular, some aspects of image processing that are not very relevant when detecting objects were considered, especially because the project aims to the detection of humans only, these calculations and aspects will be described in depth later.

Thus, the new architecture for the detection of humans in images was obtained. The algorithm was implemented in parts and from scratch using the python language and the pytorch library.

The image dataset was prepared for training and testing. After some training, corrections, and testing, an extension was made to the image database, and a readjustment of parame-

ters to correct some errors and improve the functioning of the neural network was obtained. Finally, for the proposed neural network to work in videos, the videos were processed and separated into frames using OpenCV.

3.1 Resources

3.1.1 Software

The language used to implement the proposed model is Python, the algorithm was written from scratch in this language and making use principally of the Pytorch library for the creation of the CNNs and the OpenCV library for the processing of images and videos.

3.1.2 Hardware

- Processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz
- RAM: 8 GB

3.2 Data Collection

For the development of this project, it was necessary to collect a large amount of data and images. First, the type of images that would be appropriate was identified, which are images that contain humans, in full, performing different activities and in different poses, with covered areas, sitting, running, etc. The images needed to be varied and different, to achieve this, images from 4 different datasets were used: CrowdPose dataset, PETA dataset, PennFundanPed, and Animal-10 dataset From these datasets, a total of 1000 images were taken and a new dataset was created for training, then 100 images were taken for testing and validation data.

Figure 3.1: Examples of the collected images

All these images were manually processed to obtain the format for working with YOLO algorithm, which requires a frame that enclose the object of interest, the human silhouette in this case, and a file that obtains the coordinates of the 4 corners of each box that encloses the object. If there is more than one object of interest in the image, the coordinates of the rest of the boxes will also be needed.

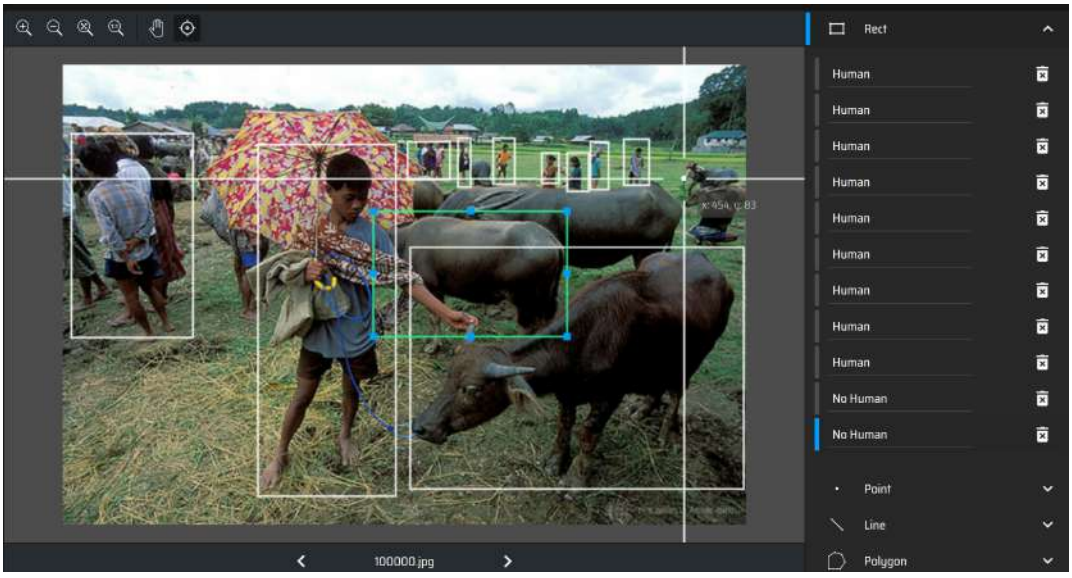

Figure 3.2: Example of image labeling

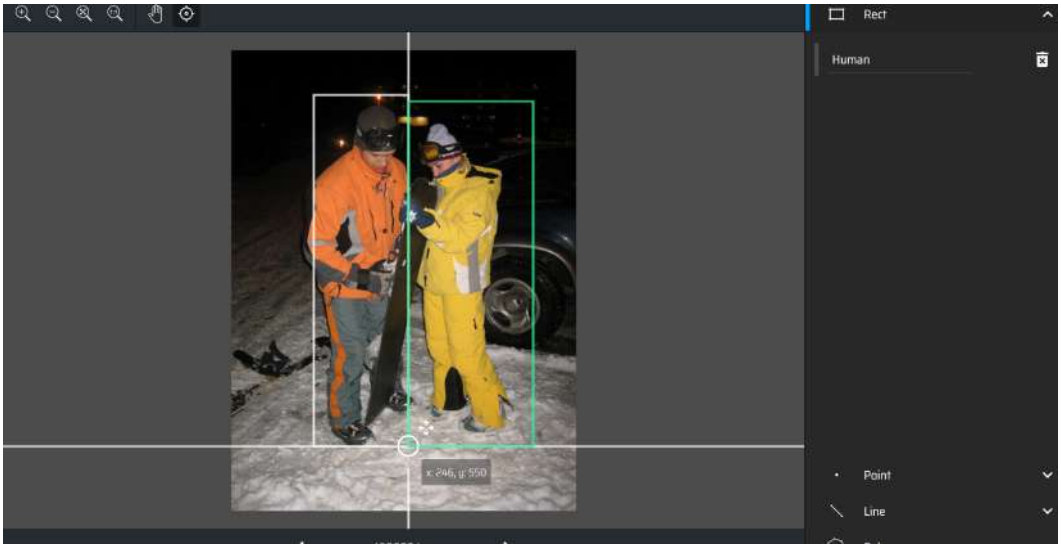

Figure 3.3: Example of image labeling

To achieve this, the MakeSense.ai software was used, where the names of the labels of each category were created, in this case, we had two categories: Human and NoHuman. A box is drawn containing the object of interest and it is given the corresponding label (see Figures 3.2 and 3.3). The output of the software is a folder with images and each image with its respective text file that contains the category number of each object and the coordinates of the object. For example for Figure 3.3 where we have 2 humans, this would be its label text file .

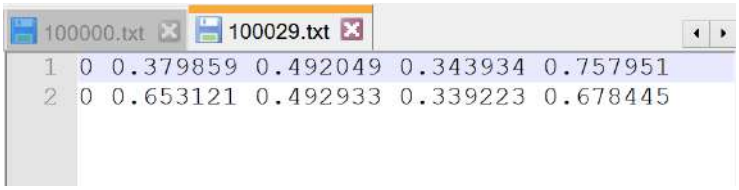

Figure 3.4: Image coordinates

Where each line represent one object of interest, and in each line, the first number represents the class, the second and third numbers are the midpoint in X and Y of the object respectively, while the fourth and fifth numbers represent the width and height of the box (See Figure 3.5). Something important to mention is that these sizes and these coordinates are taken concerning the entire image, the advantage of this is that if the image is resized, it would continue to maintain the same numbers.

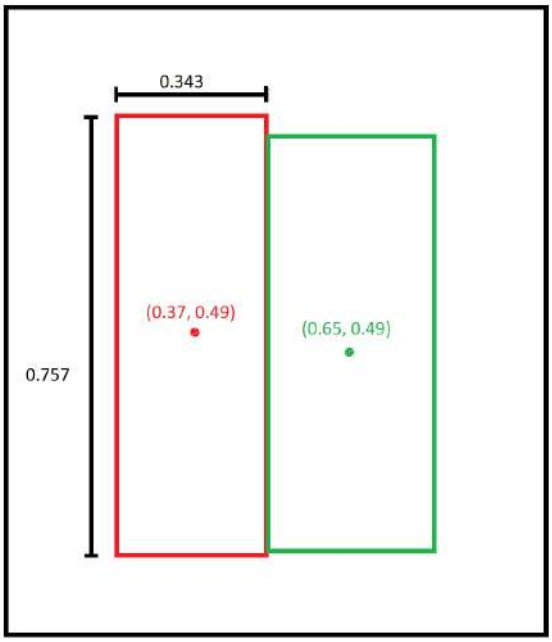

Figure 3.5: Graphical representation of coordinates from Figure 3.4

3.3 Defining the architecture

Although YOLO manages to detect objects with greater precision and speed compared to other algorithms, it needs to be used in a system with a powerful single GPU. There are computing and converting processes. All of the computing processes are done on the GPU side and the converting processes are conducted on the CPU side. [37]

The YOLO algorithm outputs at least 3 anchor boxes per cell or detected object, due to the model performs detection in 3 scales to obtain objects of different sizes[30], it would imply the implementation of a massive amount of anchor boxes. These were not considered relevant, since what is sought is that the detection is fast and efficient, the creation of these anchor boxes takes more time and also demands greater use of CPU resources. Therefore in the proposed neural network, it was suppressed and simplified, using only simple bounding boxes.

From the paper "Concurrent Real-Time Object Detection on Multiple Live Streams Using Optimization CPU and GPU Resources in YOLOv3", [37] the conversion of images was also taken into account, which in the paper is suggested as a step that does not contribute

in a relevant way to the detection process. Here is considered the conversion of RGB image to a different format unique for the operation of the neural network, after the conversion, the detection is carried out, the bounding boxes are drawn and the image is transformed back to the RGB format, this step is the one that is suppressed, working and detecting directly on the image in RGB format. These additional calculations and conversions can be considered small, but improvements in terms of CPU usage were obtained by removing them. An evaluation was made on 10 different videos with different qualities to check the CPU memory usage. [37]

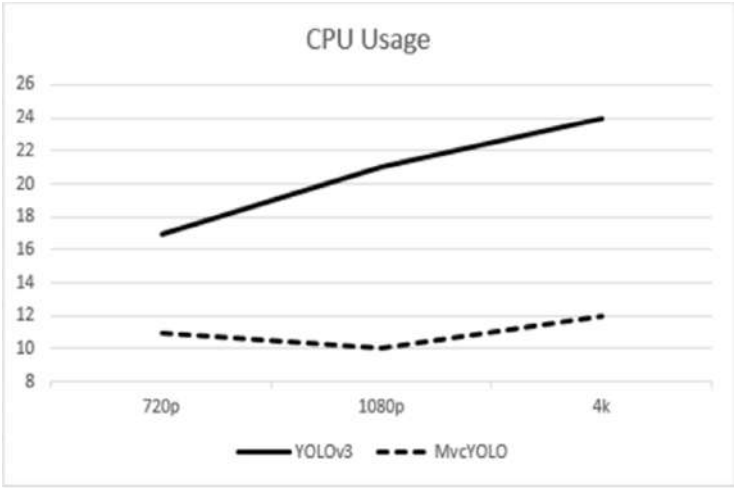

Figure 3.6: Comparison of CPU usage between YOLOv3 and MvcYOLO

The percentage of CPU usage was compared between (See Figure 3.6) YOLOv3 and MvcYOLO (it is the model proposed in the paper mentioned above) in videos with different qualities. In general, the use of the CPU increases with increasing video quality since it requires more computational power to handle large frames. The CPU usage on the MvcYOLO is lower than the general video qualities of YOLOv3.[37]

Another calculation that was suppressed was the concatenation of channels, what this refers to is that when upsampling the image, it takes a feature map from earlier in the network and merges it with the upsampled features. In addition to the fact that concatenation already requires additional operations, since it is searches in previous parts of the neural network, it requires constant revision and backtracking, which somehow interrupts the detection and flow of the neural network. By removing the concatenations, the model works with information that has been slightly altered by the neural network, but that still maintains the important information and also allows the process to be linear, without the

need to go back to the previous steps in search of information.

The Yolo Original makes 4 predictions on each scale, in this model one prediction is made on each scale. Additionally, for this specific project only 2 classes were used, which are ”Human” and ”NoHuman”.

3.3.1 Activation Function

For this model we are gonna use Leaky ReLU activation function that is an attempt to solve the dying ReLU problem that provokes f(y) equal to zero when y is less than zero. The leak helps to increase the range of the ReLU function.

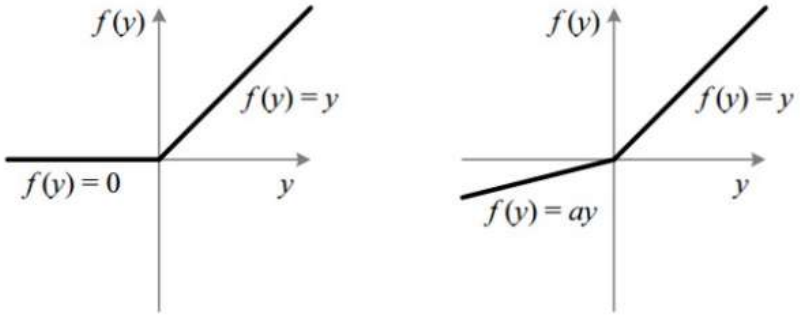

Figure 3.7: ReLU vs Leaky ReLU

Using the value of $a = 0.01$.

3.3.2 Loss Function

As we will not use anchor boxes in the new model, it was decided that the loss function that will be used will be that of the first version of YOLO which is:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

(3.1)

where 1_i^{obj} denotes if object appears in cell i and 1_{ij}^{obj} denotes that the jth bounding box predictor in cell i is "responsible" for that prediction. x and y are coordinates that show where is the center of the box in relation to the grid. w and h represent the width and height, and c is the confidence[29] .

3.3.3 New architecture

Taking into account all the previous points, this is the architecture of the proposed CNN. It consists of 36 layers. The first convolutional layer receives an input of 416x416 pixels and 3 channels (RGB). The first 28 are those that make up the modified darknet, which consists of several convolutional layers, as well as layers of residual blocks. Layers 29, 32 and 35 are also convolutional. In layers 30, 33 and 36 the scale predictions are made, in each scale the Neural Network detects the objects and defines if them belong to one of the two classes. In layers 31 and 34, upsampling is performed for 3-scale prediction.

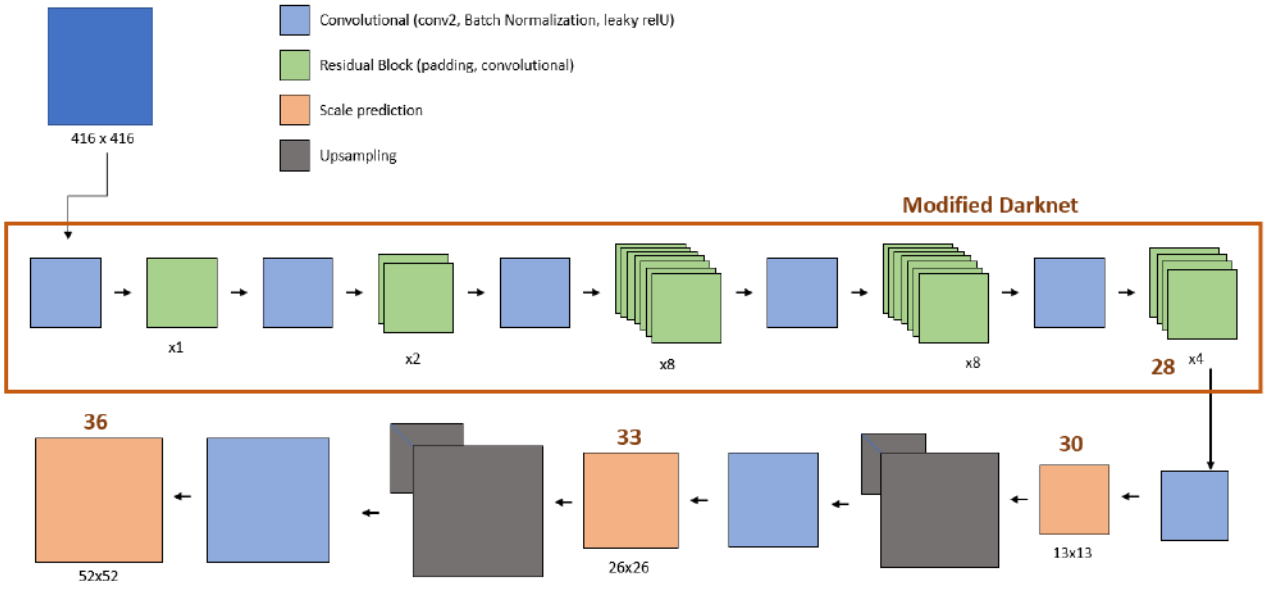

Figure 3.8: Architecture of the proposed CNN

The convolutional block consists of conv2, batch normalization, and Leaky RelU. The residual block is made up of a padding and a convolutional block. The scale prediction layers are the output layers. In the upsampling layers the size is doubled.

3.4 Dataset

As mentioned before, the dataset consists of 1000 images for training and 100 for testing, divided into 2 classes ”Human” and ”NoHuman”. This was achieved after having trained the neural network and having obtained unfavorable results.
The first time the network was trained with a dataset of 800 images and a MAP (mean average precision) of 74% was obtained to recognize humans, but when tested on images containing animals, it also detected the animals as humans.

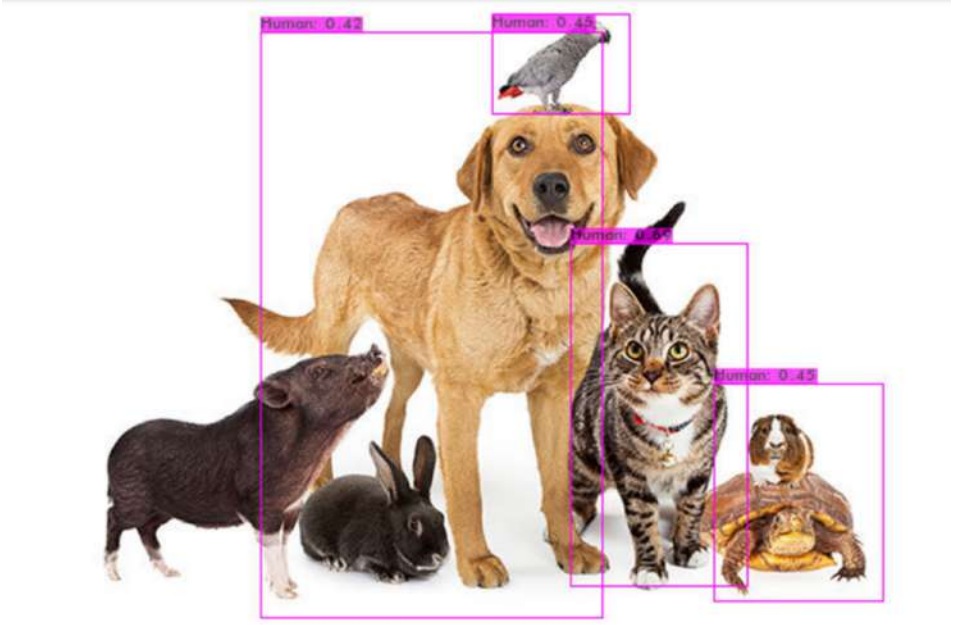

Figure 3.9: Example of error in recognition

For this reason, the ”NoHuman” class was added and images of animals were added, which were entered in that class.

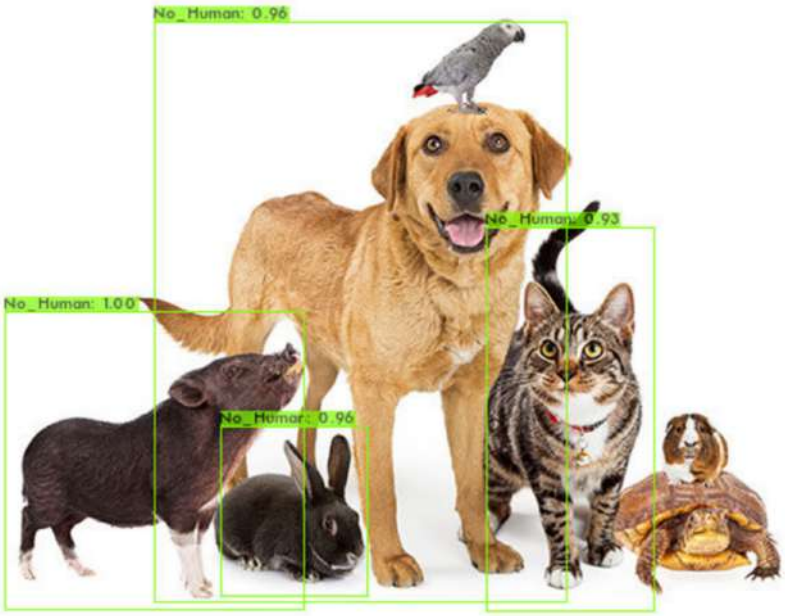

Figure 3.10: Example of error solving

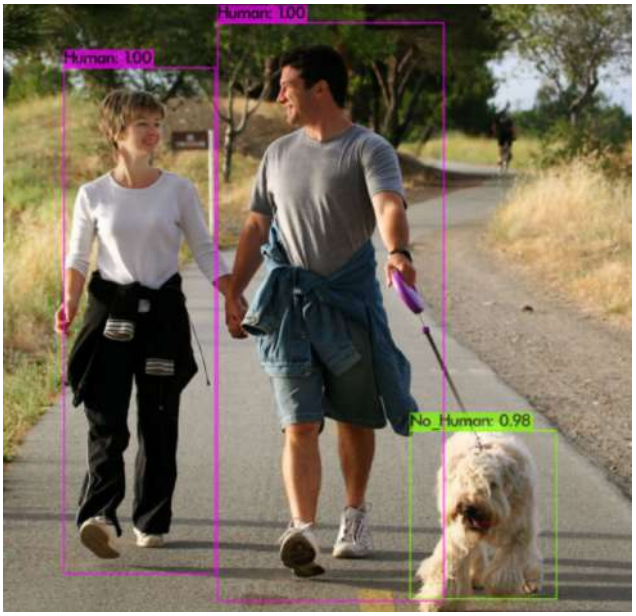

Figure 3.11: Example of the system working with both classes

Afterward, the database was expanded again since the neural network did not recognize black people, so images with people whose skin color is more varied were added, thus obtaining the final database that yielded the best results.

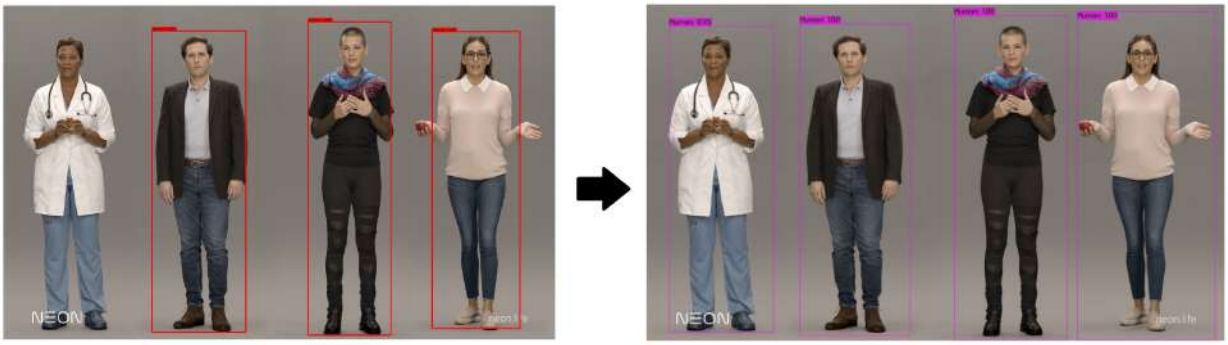

Figure 3.12: Example of error solving

For the algorithm to obtain the images and the files with their coordinates, .csv documents were created for both training and testing. The files contain two lists, one with the names of the image file and the other with the name of its corresponding .txt file that contains the coordinates.

3.5 Training and Testing

3.5.1 Hyper-parameters

For training, YOLO algorithms base their parameters on a configuration file, depending on the algorithm version. Since the proposed algorithm is different, some parameters have been withdrawn and others have been modified. The parameters to be taken into account for training in this project are:

- **Batch Size:** Batch Size determines how much data will pass through the network, or the number of images that will be processed by the network before updating the weights which is also the number of images per iteration [38].
  For this project the batch size is set to 16.

- **Epochs:** One epoch is to pass the entire training data set through the neural network once. Generally, neural networks need several epochs to achieve good predictions because, in each epoch, the weights are modified to improve the model. This hyperparameter determines the number of epochs that will be executed to train the neural network.[38]
  For this project, the number of epochs is set to 2000.

3.5.2 Metrics

- **Intersection over Union (IoU)**
  IoU is one of the most popular and used metrics in visual tasks such as segmentation, object detection, object tracking, etc. What it does is to determine how optimal the prediction of bounding box (BB_p) is compared to the real bounding box (BB_r) previously entered, it is calculated using the following formula. [39, 40]

$$IoU = \frac{area(BB_p \cap BB_r)}{area(BB_p \cup BB_r)} \tag{3.2}$$

  This formula is comparing the area that is intersected by both squares over the area of the Union. In most cases an IoU of 0.50 is considered good enough to say that an object has been correctly detected, there is the possibility of obtaining an IoU of 0.9 considered perfect since it is practically impossible to obtain a value of 1.

  With this metric we can determine the following categories:

  - True Positive (TP): The Object was detected correctly.
  - False Positive (FP): The bounding box did not detect the object in the image correctly and has a very low IoU value.
  - False Negative (FN): Failed to predict an object that was there or the object is detected as a class to which it does not actually belongs to.

- **mean Average Precision (mAP)**
  mAP is a popular metric that measures the precision of object detectors. To understand how it works, is necessary to put other important concepts into context.

  - Recall: Measures how well all positives were found.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

  - Precision: Measures how accurate the predictions, how many of the predictions that the model made were actually correct.

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

The aim of the Average Precision (AP) is to find the area under the precision-recall curve, and mAP is the average of AP. The mean Average Precision score is calculated by taking the mAP over all classes and over all IoU thresholds, in this case, it was decided that the confidence score threshold is 0.5 and for the IoU threshold is also 0.5.

The precision and recall must be obtained for each category, once this is done, the precision-recall curve can be plotted. The recall is on the x-axis while the precision is on the y-axis. To plot this curve, we take all detections from a single category and list them in descending order with respect to the class probability given by the detector. As the first element has the highest probability, there is a high probability that it is a TP (this is not always the case, since we must also consider the IoU), so the precision, in this case, would be 1, while that the recovery would be less since it is calculated against the number of fundamental truth objects. As we review all the detections made, accuracy will decrease while recovery will increase. [41]

For training the Neural Network, the 1000 training images are resized to 416 x 416 pixels. The images were then transformed into tensors to be able to work with PyTorch and were placed in batches of 16. Finally, the training was run for 2000 epochs, to avoid overfitting since in previous attempts to carry out more iterations, 2500 and 3000 to be specific, unfavorable values were obtained. During the neural network training, the metric values were monitored, the IoU value greater than 0.5 and the decrease in the average loss value were taken into account. The mAP value that was calculated every 500 iterations was also tracked. When obtaining the best results, the data, the model, and the weights were saved. The neural network was trained to complete the 2000 iterations in an approximate time of 15 hours.

To check the performance and accuracy of the neural network, the 100 images of the test data were used and the predictions were compared with the original labels that had been previously created. All the image processing from these steps were done using the Pillow library.

3.6 Working on videos

This section presents how the videos were processed so that the system can be implemented and the neural network detects humans in live videos, which is the main objective of the project.

First, appropriate material was sought to test the functioning of the neural network, the data set consists of 5 videos in .mp4 format. These videos are free to download and were found on the Vimeo website(8). The videos are of people in parks, pedestrians, people sitting, people running, etc., and have a duration of between 5 and 8 seconds each.
Then, using OpenCV, a function was created to extract the frames from the videos. The function reads the video, makes sure that the video has not finished yet, and captures the frame of that moment in .jpg format. It is important to clarify that the number of frames depends on the quality and duration of the video, in a high-quality video the function manages to extract up to 23 frames per second. The program that contains the neural network was modified so that it receives the image corresponding to the extracted frame, the image is processed and displayed in a window. Immediately enter the next frame, the previous image is closed and the new image is shown, so on until the video is finished.

Figure 3.13: Examples of frames extracted from a video

Figure 3.14: Examples of processed frames

In order to observe the results, the processed images are saved in sequence, then using the VideoWriter function of OpenCV, the frames are joined and written in a video in .mp4 format.

Chapter 4

Results and Discussion

This chapter presents all the experimental, numerical and final results obtained by the neural network that was proposed and developed. The results of all the images selected for the training and test dataset will be displayed, as well as the values obtained in the dataset that contains the videos.
The main objective of this thesis is to propose a neural network that is responsible for recognizing and tracking humans in videos, however, because the network was trained in images, it is of the utmost importance for this project to expose the data obtained from these training. As mentioned above, several pieces of training had to be executed and the content of the training database was changed, it was expanded, varied images were added and a second class was also added, as well as modified and experimented with some numerical parameters. This is to correct errors and improve the detection results of the neural network. Some previous pieces of training were performed, but we will focus on the final results of the training and testing that was done after the data was optimized, with which the best numerical and visual results were obtained.

4.1 Final Results

In the final training 1000 images were processed and 2000 iterations were executed. The mAP was calculated every 500 iterations:

- 500 iterations = 0.442 or 44.2%
- 1000 iterations = 0.679 or 67.9%

- 1500 iterations = 0.8215 or 82.15%
- 2000 iterations = 0.8667 or 86.67%

The average loss is calculated in each iteration, but for simplicity, only the value obtained in every 500 iterations will be presented in the same way:

- 500 iterations = 2360.07226
- 1000 iterations = 14.2098
- 1500 iterations = 5.447
- 2000 iterations = 1.366

As we can see, the MAP value is reliable, since it is considered a good value from 0.5, and 0.837 was obtained, which is very close to 1, which represents that the aggregated data and modifications made to the parameters were correct, also tells us that the neural network had a practically successful training. We can also observe the reduction of the average loss value, since it started with a huge value and was drastically reduced in the first 1000 iterations, from there, the value remained almost constant, but we can observe that the value decreased significantly, from 2360.07 to 1.366, approaching the 0.

To verify these results, the neural network was tested in 100 new images for the neural network, and an analysis of the values was made. The following results were obtained:

Number of detections = 234				
TP	FP	FN	PRECISION	RECALL
198	36	42	0.84	0.82

Table 4.1: Neural Network Test Results.

- average IoU = 0.647
- mAP = 0.7424 or 74.24%
- Accuracy = 0.83
- Execution time = 8.3 sec

Figure 4.1: Example of processed images

Figure 4.2: Examples of processed images

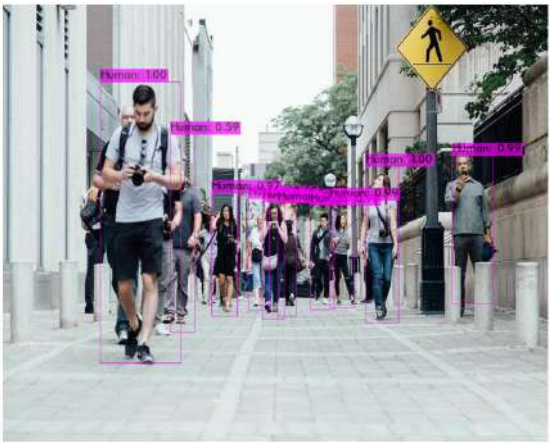

Figure 4.3: Example of processed images

From the average values that we observe in Table 4.1, the mAP was calculated obtaining a result of 78.24, which is a value greater than 0.5, therefore it is reliable. It should be mentioned that what is mainly sought in this project is that the images are processed quickly since it is intended to work with videos. A value of 0.742 is quite accurate considering also that this test took 8.3 seconds for the neural network to perform, which means that it processed approximately 12 images per second.

The accuracy shows us the percentage of the objects in the images that were correctly detected, the neural network was precise with 83% of the labeled objects.

4.1.1 Videos results

As mentioned above, 5 videos were obtained to perform this experimental part. The videos vary in resolution In fig x we can see the average accuracy values obtained from each video. As we can see, video1 is the one that had the best results, this is surely due to the fact that it has a higher resolution to simulate different camera lenses and test results with different video quality, also they vary in duration from 6 to 8 seconds, as we can see in Table 4.2

Video	Format	Resolution	duration	Num. frames	frames/sec
1	mp4	1920 x 1080	6 sec	120	≈ 20
2	mp4	960 x 540	8 sec	134	≈ 17
3	mp4	960 x 540	6 sec	107	≈ 16
4	mp4	480 x 270	8 sec	145	≈ 18
5	mp4	480 x 270	6 sec	93	≈ 16

Table 4.2: Videos specifications

The neural network was tested in the 5 videos, because we do not have videos with frames already labeled, the IoU could not be calculated and therefore neither the mAP, so the processing time of the videos and the accuracy that suggests the neural network were taken into account for the feasibility analysis, the following results were obtained.

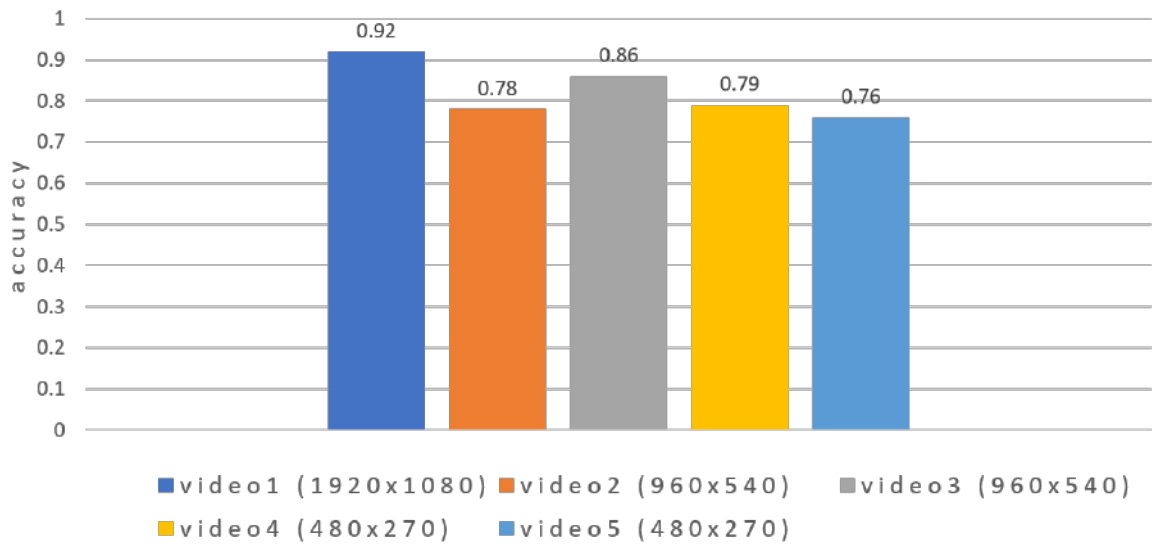

Figure 4.4: Average accuracy results

In Figure 4.4 we can see the average accuracy values obtained from each video. As we can see, video1 is the one that had the best results, reaching a value of 0.92, the next video that obtained the best results is video 3 with a value of 0.86, finally we have videos 2, 4 and 5 that oscillate between 0.76 and 0.79. We can say that video 1 obtained the best results due to the high resolution, as we can see, the lowest values were obtained by the videos with lower resolution.

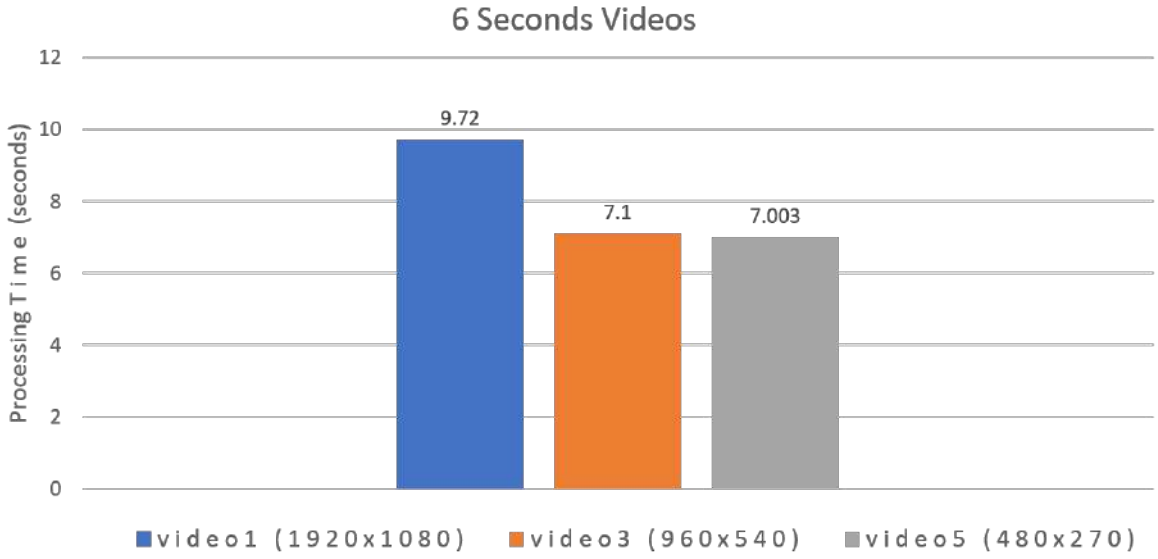

Figure 4.5: Time results in 6 second videos

Figure 4.5 shows the results of the time in seconds it took the model to process the

6-second videos. As we can see, despite the fact that the 3 videos have the same duration, the processing time of each one varies. Video1 is the one that took the longest processing time, taking 9.72 seconds, which varies a lot from videos 3 and 5, which took 7.1 and 7.0 seconds each.
Using this information, the number of frames that were processed per second of each video, presented in Table 4.3, was calculated. For the first video, human detection was performed at an average of 12 frames per second, in video 3, 15 frames were processed every second and in the last video 14 frames per second were processed. This is also due to the resolution, as the video1 has a higher resolution, it also has more information that needs to be analyzed by the neural network, we can also see that video1 has a higher number of frames, for these reasons it takes the model more time to perform the detection.

Video	Resolution	duration	Num. frames	Processing time	Processed frames/sec
1	1920 x 1080	6 sec	120	9.72 sec	≈ 12
3	960 x 540	6 sec	107	7.1 sec	≈ 15
5	480 x 270	6 sec	93	7.003 sec	≈ 14

Table 4.3: Processed frames per second

Moving on to the videos with a duration of 8 seconds, we can see in Figure 4.6 that the detection time does not vary much from both videos, the video2 took 10 seconds to be processed while the video4 took 10.2 seconds. As we can see in this case, the video with lower resolution, video4, took longer to be processed, this is because the video contains more frames than video2, so each image has less information to proceed, but more of pictures.

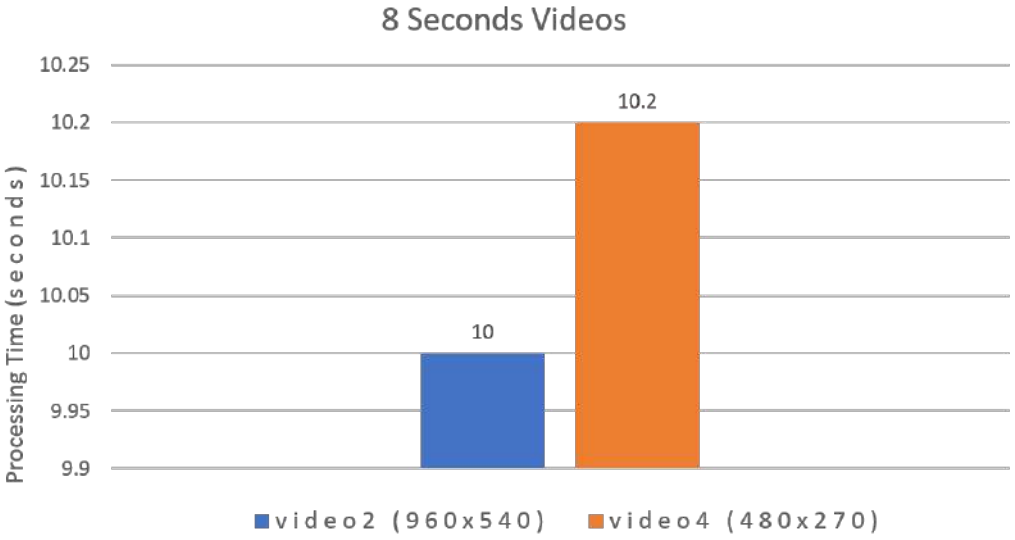

Figure 4.6: Time results in 8 second videos

This can also be verified with the information in Table 4.4, since in the same way as with the 6-second videos, the estimated number of frames processed per second was calculated, we can see that they only vary by 1.

Video	Resolution	duration	Num. frames	Processing time	Processed frames/sec
2	960 x 540	8 sec	134	10 sec	≈ 13
4	480 x 270	8 sec	145	10.2 sec	≈ 14

Table 4.4: Processed frames per second

Chapter 5

Conclusions and Recommendations

From the research carried out for the development of this thesis, we can conclude that the detection of objects is a subject that has been widely studied since it has a great variety of uses, and there are always researchers trying to innovate in some way. Speaking specifically of the detection of humans in videos, there are several applications, mainly in security matters. The best object detection algorithms, such as YOLO, can achieve their objective successfully and in a very short time per frame, but they require equipment with high processing power and memory, as well as a good graphics card.

The YOLO object detection model can be adapted to detect human silhouette, requiring an analysis of the components of the algorithm, a review of the calculations it performs, and also a reduction of unnecessary or irrelevant processes. In this work, a new dedicated model of YOLO was created and adapted to specifically detect humans.
It is also important to mention that the training phase is quite experimental at first, the values of some parameters must be constantly changed and refined, the database modified and several workouts performed until the best results were obtained.

The proposed human detection model presents successful results since it manages to detect humans in the videos with good accuracy. The neural network performs better detection in higher resolution videos, but it takes longer as it has more information to process, while in low-resolution videos it detects more quickly, but it fails to detect all humans correctly. Our experience suggests working with videos that have an average resolution to obtain positive results in terms of precision and the time in which the detection is performed.

For future work we recommend working with machines that have greater computational power, also make a more in-depth review of some parameters and processes that were not studied in this research, to be able to modify and see if precision or speed can be improved.

Bibliography

[1] J. Brownlee, "A gentle introduction to computer vision," https://machinelearningmastery.com/what-is-computer-vision/, 2019, deep Learning for Computer Vision.

[2] S. M. E. H. Manoranjan Paul and S. Chakraborty, "Human detection in surveillance videos and its applications - a review," *EURASIP Journal on Advances in Signal Processing*, 2013.

[3] A. Iqbal, "Obstacle detection and track detection in autonomous cars," *IntechOpen*, 2020.

[4] S. Guang, "Human detection, tracking and segmentation in surveillance video," *University of Central Florida STARS*, 2014.

[5] M. Mandal, "Introduction to convolutional neural networks (cnn)," in *Data Science Blogathon 7*, 2021.

[6] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Spinger London, 2011.

[7] S. A. A. S. M. B. Khan Salman, Hossein Rahmani, *A Guide to Convolutional Neural Networks for Computer Vision*, 15 from series synthesis lectures on computer vision ed. Morgan Claypool, 2018.

[8] IBM, "What is computer vision," https://www.ibm.com/topics/computer-vision, 2020, accessed July 2021.

[9] S. Selvi and A. I. Chellam, "Smart video surveillance: Object detection, tracking and classification," *International Journal of Innovations and Advancement in Computer Science*, vol. 7, 2018.

[10] MatLab, "Cnn design and training with matlab," https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html, 2019, accessed September 2020.

[11] N. Babich, "What is computer vision how does it work? an introduction," https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/, 2020, accessed July 2021.

[12] W. J. Z. H. L. JL, T. SM and W. YK, "A review on object detection based on deep convolutional neural networks for autonomous driving," in *Chinese Control and Decision Conference*, 2019.

[13] P. H. J. B. Wu, F. Iandola and K. Keutzer, "Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, July 2017.

[14] K. Gurney, *An introduction to neural networks*, 2nd ed. University College London, 2004, editorial ROUTLEDGE.

[15] A. G. Nwankpa, W. Ijomah and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *Cornell University*, 2018.

[16] G. B. O. Y. LeCun, L. Bottou and K. R. Muller, *Efficient BackProp*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, available:https://doi.org/10.1007/3-540-49430-8 2.

[17] W. Koehrsen, "Aoverfitting vs. underfitting: A complete example," Towards Data Science, 2018, accessed July 2021.

[18] S. SHARMA, "Activation functions in neural networks," https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6l, 2017, accessed July 2021.

[19] J. M. S. A. M. J. Emilio Soria, Marcelino Martinez, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. University of Valencia, Spain, 2009.

[20] M. Z. Mulla, "Cost, activation, loss function—— neural network—— deep learning," https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de, 2020, accessed apr 2021.

[21] P. Christoffersen and K. Jacobs, "The importance of the loss function in option valuation," *Journal of Financial Economics*, vol. 72, 2004.

[22] P. S. C. P. B. F. Lorena, G. Robinson and M. Franco, "Automatic microstructural classification with convolutional neural network," in *Conference on Information Technologies and Communication of Ecuador. Springer*, 2018, p. 170–181.

[23] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation," *Journal of Big Data*, vol. 6, 2019.

[24] C. Oinar, "Object detection explained: R-cnn," https://towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76, 2020, accessed Mar 2021.

[25] R. G. S. Ren, K. He and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis Machine*, vol. 39, p. 1137–1149, 2017.

[26] R. Gandhi, "R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms," https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e, 2018, accessed Aug 2021.

[27] R. G. Shaoqing Ren, Kaiming He and J. Sun, "Faster r-cnn: Towards real-timeobject detection with region proposal networks," *In Advances in Neural Information Processing Systems*, p. 91–99, 2015.

[28] R. G. J. Redmon, S. Divvala and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[29] R. G. A. F. Joseph Redmon, Santosh Divvala, "You only look once: Unified, real-time object detection," *University of Washington*, 2016.

[30] A. F. Joseph Redmon, "Yolov3: An incremental improvement," *University of Washington*, 2018.

[31] ——, "Yolo9000: Better, faster, stronger," *University of Washington*, 2016.

[32] S. A. A. S. S. Khan, H. Rahmani and M. Bennamoun, "A guide to convolutional neural networks for computer vision," in *Synthesis Lectures on Computer Vision*, vol. 8, 2018, p. 1–207.

[33] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," in *Convolutional Neural Networks Vis. Recognit*, 2017.

[34] M. G. Rasmus Rothe and L. V. Gool, "Non-maximum suppression for object detection by passing messages between windows," in *Asian Conference on Computer Vision (ACCV 2014)*, 2015.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*, vol. 37, 2015, p. 448–456.

[36] Z. X. R. S. y. S. J. He, K., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 770–778.

[37] A. H. Y. Samira Karimi Mansoub, Rahem Abri, "Concurrent real-time object detection on multiple live streams usingoptimization cpu and gpu resources in yolov3," *Hacettepe University*, 2019.

[38] J. Brownlee, "What is the difference between a batch and an epoch in a neural network?" *Deep Learning; Machine Learning Mastery: Vermont*, 2018, australia.

[39] S. Nowozin, "Optimal decisions from probabilistic models: the intersection-over-union case," *IEEE Xplore*, 2014.

[40] X. S. L. J. Zheng Y., Zhang D. and Z. J., "Rotation-robust intersection over union for 3d object detection," in *Computer Vision: 16th European Conference, Glasgow, UK*, 2020, p. 467.

[41] S. Yohanandan, "map (mean average precision)," https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2.