

# General introduction

Since most road accidents are caused by human error, Advanced Driver Assistance Systems (ADAS) are systems developed to automate, adapt and improve the vehicle for greater safety and better driving.

The automated system provided by ADAS to the vehicle has proven to reduce the number of deaths and damage on the roads remarkably, by minimizing human error. Safety devices are designed to avoid collisions and accidents by providing technologies that alert the driver to potential problems, or to avoid collisions by implementing protective measures and taking control of the vehicle.

We intend to deploy an ADAS (Advanced Driving Assistance System) based on artificial intelligence technologies. To meet these needs, it is essential to predict the trajectories of moving objects.

To provide a framework for our work, we have drawn up a set of specifications covering 4 main areas.

Axis 1: context and presentation of the project, including a brief definition of our project.

Axis 2: Needs analysis, in which we expressed the usefulness of our project.

Area 3: Functional expression of requirements, in which the constraints and functionalities of our system are described.

Area 4: Implementation schedule.

To achieve this objective, we need to:

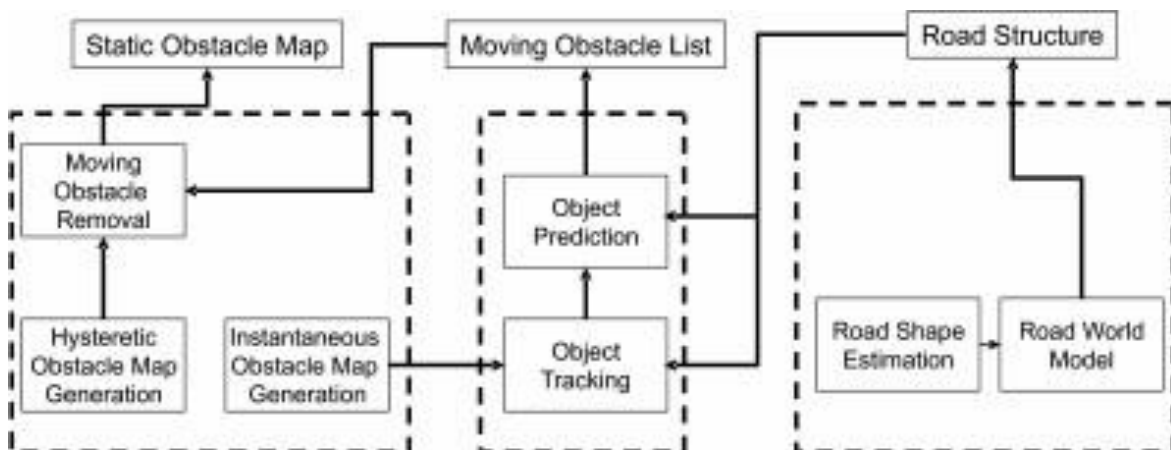
- Detect moving objects.
- Track moving objects.
- Predict the trajectories of moving objects.

# Chapter I: Project presentation and literature review

## I. Project presentation

Driving in an urban environment requires interaction with other vehicles, whether following a slow-moving vehicle, coordinating to take turns with vehicles at intersections, or maneuvering around other vehicles to reach spot parking, it's almost impossible to make a car journey without being affected by another vehicle in some way.

As drivers, assistance systems and autonomous vehicles become more sophisticated, reasoning about these interactions with vehicles becomes increasingly important. Our role begins with predicting the trajectories of moving objects, based on the set of information gathered via sensors, cameras, etc., which will then be processed to help determine the set of future decisions the car will take.



*Figure 1 : Situation of our project*

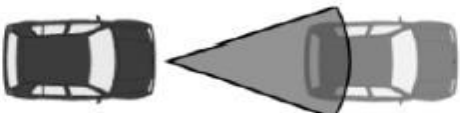
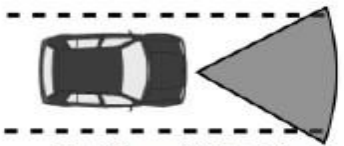
## II. Literature review

### 1. Advanced driving systems

Advanced driver assistance systems are intelligent systems that reside inside the vehicle and assist the main driver in a variety of ways. These systems can be used to provide vital information on traffic conditions, road closures and blockages, congestion levels, suggested routes to avoid traffic jams, and so on.

#### a. Overview of ADAS technologies

To give an idea of what driver assistance systems represent for users, we present an overview of existing technologies. For convenience, they have been divided into sub-categories. This brief overview of existing ADAS technology highlights only the most “common” types of ADAS.

	ADAS		Description
Longitudinal	ACC	Adaptive Cruise Control	<p>ACC is becoming a more and more common accessory in modern cars. Basically, this technology keeps a safe distance between the driver's car and vehicles ahead. The driver can adjust the distance, and the system makes sure it's maintained, using throttle and brake control. Most ACC systems have influence on the driving task (they control brake and throttle), but still allow user take-overs.</p>  <p><i>Fig 1: Adaptive Cruise Control</i></p>
	FCW	Forward Collision Warning	<p>Like the ACC, this system detects vehicles in front of the driver's car. Obviously, it can be integrated with ACC. However, current systems still have problems distinguishing cars from trees, bridges from road signs, etc.</p>  <p><i>Fig 2: Forward Collision Warning</i></p>
	ISA	Intelligent Speed Assistance	<p>ISA influences the speed at which a car is driving. The maximum speed can be pre-set, or acquired from GPS data. Interfacing with the driver is done via the acceleration pedal, or by using visual or audio warnings.</p>

*Table 1 : adas longitudinal*

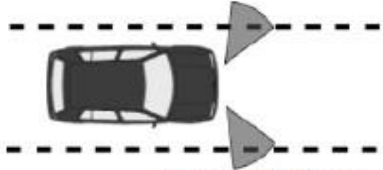

Lateral Support	LDW	Lane Departure Warning	The main task of Lane Departure Warning is to make sure a car is driving safely between road marks (i.e. in a lane). LDW uses cameras and computer systems to detect and process roadsides and lane markings, and warn the driver if necessary. Acceptance of LDW is expected to be a problem because control of the car is given to the computer, and chances of false alarms are still present.
	LKS	Lane Keeping System	<p>An extended version of the LDW system is the Lane Keeping System. Instead of warning the driver about the unintended lane departure, LKS intervenes with the driving task by using steering wheel actuators. LKS can completely take over the steering task of the driver.</p>  <p><i>Fig 3: Lane Keeping System</i></p>
	LCA	Lane Change Assistance	<p>LCA is a collection of technologies taking care of blind spots and rear-view problems. It uses sensors to detect objects and vehicles which normally can't be seen by the driver because of obstructed view. Also, approaching vehicles from behind can be detected in time, and the driver can be informed of this.</p>  <p><i>Fig 4: Lane Change Assistance</i></p>

Table 2 :adas lateral support

Miscellaneous		Night Vision Systems	These systems provide the driver with an enhanced view of the outside world. It's meant to be used during bad weather or night time. Though already implemented in several car models, the system still has a problem with its interface: how to present the enhanced image to the user. Current solutions consist of displaying the image on a monitor on the dashboard.
		Parking Assistance	The Parking Assistance system looks like Lane Change Assistance, but is meant for low speed and short distance, for example when parking a car. Using sensors a car can measure available space, and show this information to the driver. Current systems have limited use because of the low range these sensors operate with. Future developments will let the system take over control of the car during parking, letting the car park itself.
		Fuel Economy Devices	With Fuel Economy Devices the fuel flow and usage can be monitored and analysed per car. A system can intervene by informing the driver about the fuel usage, or by actively intervening, using an active gas pedal or other active systems.

Table 3 :adas miscellaneous

## 2. Machine learning

### a. Definition

Machine Learning (ML) is the study of computer algorithms that improve automatically through experience and the use of data. It is considered part of artificial intelligence. Machine Learning algorithms build a model from a sample of data, called “training data”, to make predictions or decisions without being explicitly programmed to do so.

Machine Learning algorithms are used in a wide variety of applications, such as e-mail filtering and computer vision, where it is difficult or impossible to develop conventional algorithms to perform the necessary tasks.

Machine Learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It includes teaching machines to perform certain tasks, based on the data they are given. For simple tasks entrusted to computers, algorithms can be programmed to tell the machine how to perform all the steps required to solve the problem in question; on the computer's part, no learning is required. For more advanced tasks, it may be difficult for a human to create the necessary algorithms manually. In practice, it may be more efficient to help the machine develop its own algorithm, rather than asking human programmers to specify each necessary step.

### b. Machine Learning approaches

Machine Learning approaches are traditionally divided into three broad categories, depending on the nature of the “signal” or “feedback” available to the learning system:

- Supervised learning: the computer is presented with examples of inputs and their desired outputs, given by a “teacher”, and the aim is to learn a general rule that links the inputs to the outputs.

- Unsupervised learning: No label is given to the learning algorithm, leaving it alone to find a structure in its input.

Unsupervised learning can be a goal (discovering hidden patterns in data) or a means to an end (feature learning).

- Reinforcement learning: A computer program interacts with a dynamic environment in which it must achieve a certain objective (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program receives reward-like feedback, which it tries to maximize.

Other approaches have been developed that don't exactly correspond to this triple categorization, and sometimes more than one is used by the same machine learning system. For example, topic modeling, dimensionality reduction or meta-learning.

By 2020, Deep Learning has become the dominant approach for much of the ongoing work in Machine Learning.

### c. Machine Learning applications

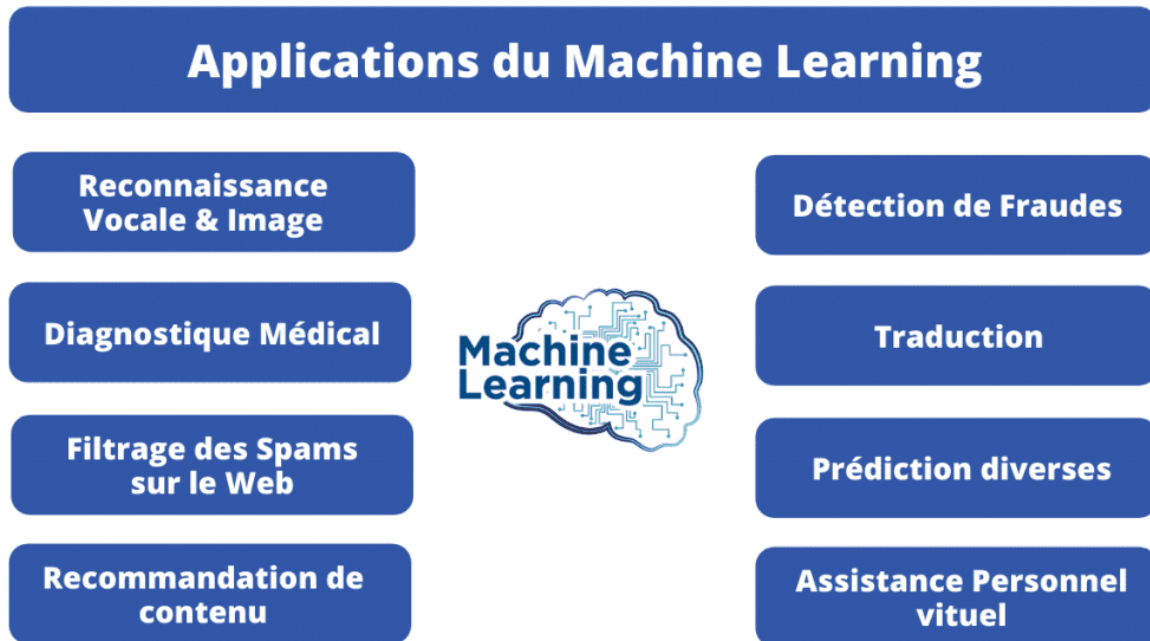


Figure 2: Machine Learning applications

### 3. Deep Learning

#### a. Introduction to Deep Learning

Deep Learning is part of a wider family of Machine Learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep Learning architectures such as Deep neural networks, Recurrent neural networks and Convolutional neural networks have been applied to fields such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, materials inspection and board game programs, where they have produced results comparable to and, in some cases, superior to the performance of human experts.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs differ from biological brains in several ways. Specifically, neural networks tend to be static and symbolic, whereas the biological brains of most living organisms are dynamic (plastic) and analog.

The adjective “deep” in Deep Learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, while a network with a non-polynomial activation function and a hidden layer of unlimited width can. Deep Learning is a modern variant involving an unbounded number of layers of limited size, enabling practical application and optimized implementation, while retaining theoretical universality under mild conditions. In Deep Learning, layers are also allowed to be heterogeneous and to deviate widely from biological connectionist models, in the interests of efficiency, learning and comprehension, hence the term “structured”.

#### b. The difference between Machine Learning and Deep Learning

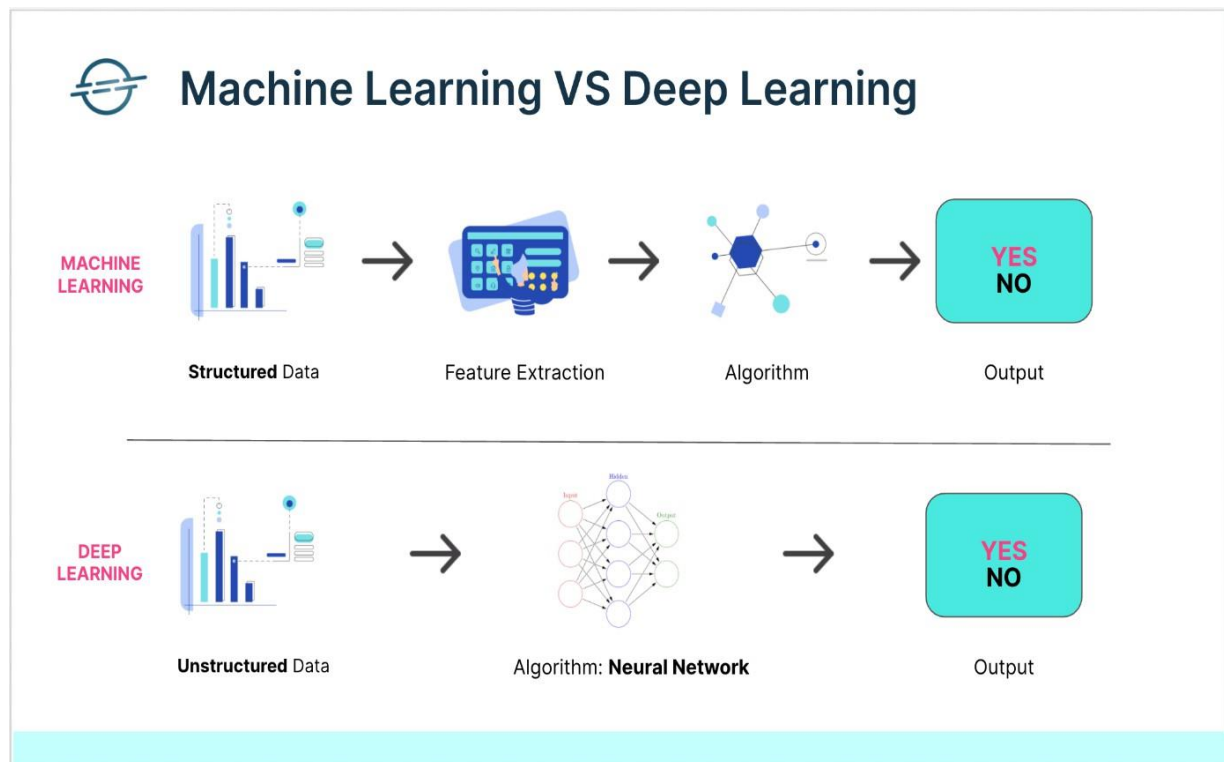


Figure 3 : machine learning vs Deep learning

## 4. Neural Network

### a. Definition of neural network

A neural network is a series of algorithms that strive to recognize the underlying relationships in a data set through a process that mimics the functioning of the human brain. In this sense, neural networks refer to systems of neurons, whether organic or artificial in nature. Neural networks can adapt to changing inputs, so the network generates the best possible result without having to redefine the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is rapidly gaining popularity in today's systems development.



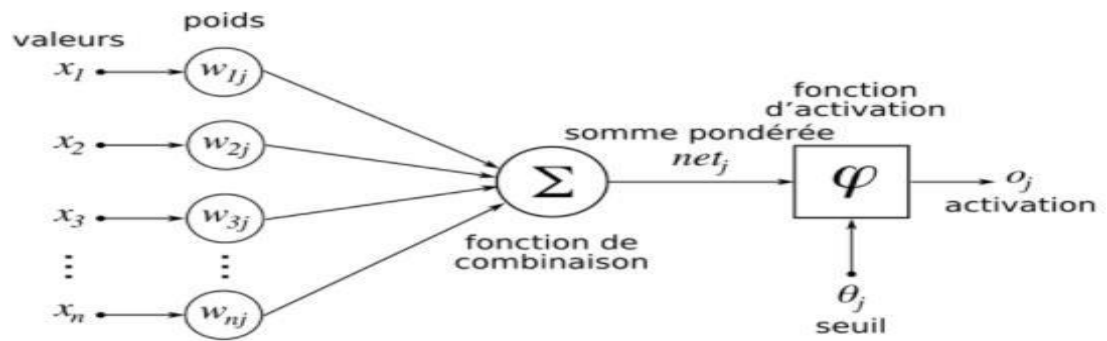


Figure 4 : NEURAL NETWORK ARCHITECTURE

## b. Convolutional Neural Network

In Deep Learning, a convolutional neural network (CNN) is a class of deep neural networks, most often applied to the analysis of visual imagery. They are also known as translation-invariant or space-invariant artificial neural networks (SIANNs), due to the shared-weight architecture of the convolution kernels that analyze the hidden layers and translation-invariance features.

They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons generally refer to fully connected networks, i.e. each neuron in one layer is connected to all neurons in the next layer. The fact that these networks are “fully connected” makes them prone to over-fitting data. Typical regularization methods involve varying weights as the loss function is minimized, while randomly reducing connectivity. CNN networks take a different approach to regularization, leveraging the hierarchical model of the data and assembling models of increasing complexity using smaller, simpler models etched into the filters. As a result, on the scale of connectedness and complexity, CNNs are at the lower end of the scale.

Convolutional networks were inspired by biological processes insofar as the pattern of connectivity between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field, called the receptive field. The receptive fields of different neurons partially overlap, so that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize filters or convolution kernels which, in traditional algorithms, are developed by hand. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

### c. Recurrent neural network

A recurrent neural network (RNN) is a class of artificial neural networks in which the connections between nodes form a directed graph along a temporal sequence.

This enables it to exhibit dynamic temporal behavior. Derived from feed-forward neural networks, RNNs can use their internal state (memory) to process input sequences of variable length. This makes them applicable to tasks such as non-segmented, connected handwriting recognition or speech recognition.

The term “recurrent neural network” is used indiscriminately to refer to two broad classes of networks with a similar general structure, one being finite-pulse and the other infinite-pulse. Both classes of network exhibit dynamic temporal behavior. A finite-pulse recurrent network is a directed acyclic graph that can be unwound and replaced by a strict feedforward neural network, while an infinite-pulse recurrent network is a directed cyclic graph that cannot be unwound.

Both finite-pulse and infinite-pulse recurrent networks can have additional stored states, and the storage can be under the direct control of the neural network. Storage can also be replaced by another network or graph, if it incorporates delays or features feedback loops. Such controlled states are called “gated state” or “gated memory”, and form part of long-term memory networks (LSTMs) and gated recurrent units. They are also known as feedback neural networks (FNN).

#### d. The difference between CNN & RNN

The main difference between CNNs and RNNs is the ability to process temporal information or data in the form of sequences, such as a sentence. In addition, convolutional neural networks and recurrent neural networks are used for completely different purposes, and there are differences in the structures of the neural networks themselves to accommodate these different use cases.

CNNs use filters in conventional layers to transform data. RNNs, on the other hand, reuse activation functions from other data points in the sequence to generate the next output in a series..

CNN	RNN
It is suitable for spatial data such as images.	RNN is suitable for temporal data, also called sequential data.
CNN is considered to be more powerful than RNN.	RNN includes less feature compatibility when compared to CNN.
This network takes fixed size inputs and generates fixed size outputs.	RNN can handle arbitrary input/output lengths.
CNN is a type of feed-forward artificial neural network with variations of multilayer perceptrons designed to use minimal amounts of preprocessing.	RNN unlike feed forward neural networks - can use their internal memory to process arbitrary sequences of inputs.
CNNs use connectivity pattern between the neurons. This is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field.	Recurrent neural networks use time-series information - what a user spoke last will impact what he/she will speak next.
CNNs are ideal for images and video processing.	RNNs are ideal for text and speech analysis.

Table 4: RNN vs CNN

# Chapter II: Technical Work

## I. Comparative study of detection solutions

### 1. OpenCV-based algorithm

En First, we started with an algorithm we found during our research. The objective of the given program is to detect the object of interest (car) in the video images and to continue tracking the same object. This is an example of vehicle detection in Python.

First: Install “opencv - numpy” packages

Second: Download the file cars.xml “this is the already trained model we're going to use”.

Third: Execute code

```
# Loop runs if capturing has been initialized.
while True:
    # reads frames from a video
    ret, frames = cap.read()
    if ret == False:
        break
    # convert to gray scale of each frames
    gray = cv2.cvtColor(frames, cv2.COLOR_BGR2GRAY)

    # Detects cars of different sizes in the input image
    cars = car_cascade.detectMultiScale(gray, 1.1, 1)

    # To draw a rectangle in each cars
    for (x,y,w,h) in cars:
        cv2.rectangle(frames,(x,y),(x+w,y+h),(0,0,255),2)

    # Display frames in a window
    cv2.imshow('video2', frames)

    # Wait for Esc key to stop
    if cv2.waitKey(33) == 27:
        break

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

```

# OpenCV Python program to detect cars in video frame
# import libraries of python OpenCV
import cv2

# capture frames from a video
cap = cv2.VideoCapture('Downloads/video.avi')

# Trained XML classifiers describes some features of some object we want to detect
car_cascade = cv2.CascadeClassifier('Desktop/cars.xml')

# Loop runs if capturing has been initialized.
while True:
    # reads frames from a video
    ret, frames = cap.read()
    if ret == False:
        break
    # convert to gray scale of each frames
    gray = cv2.cvtColor(frames, cv2.COLOR_BGR2GRAY)

    # Detects cars of different sizes in the input image
    cars = car_cascade.detectMultiScale(gray, 1.1, 1)

    # To draw a rectangle in each cars
    for (x,y,w,h) in cars:
        cv2.rectangle(frames, (x,y), (x+w,y+h), (0,0,255), 2)

```

Figure 5:opencv based code

After running the program, the output is a video in which the vehicles are detected and tracked. The results of this detection:

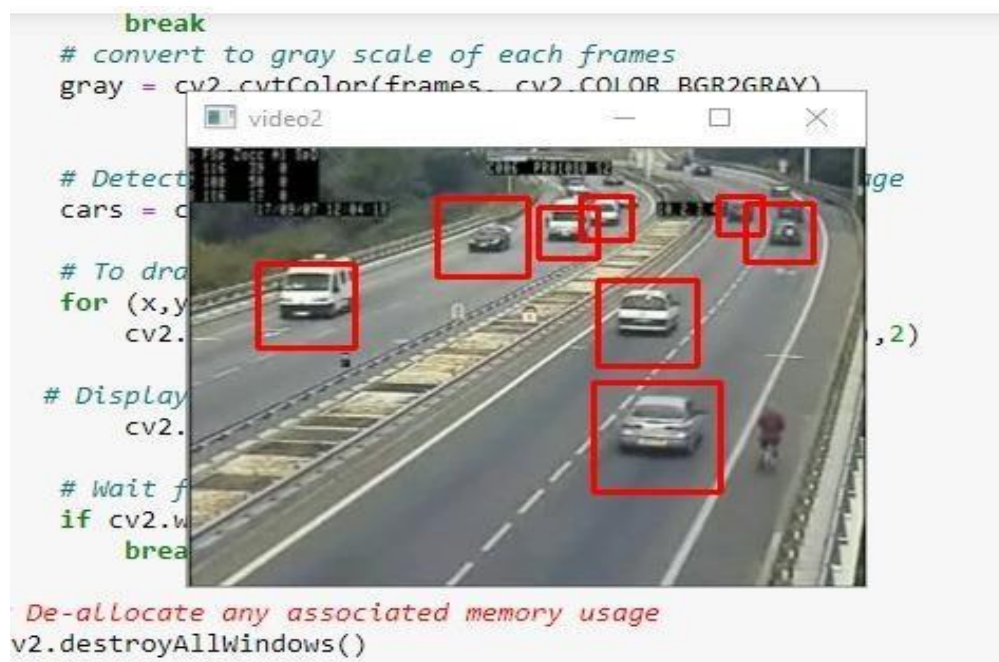


Figure 6: result of detection



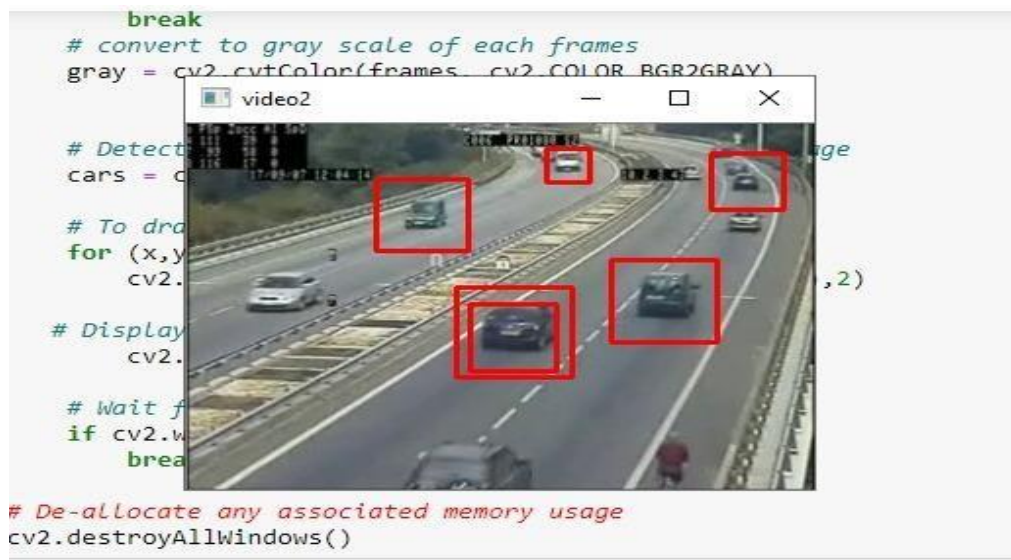


Figure 7: results of detection 2

This code was presented by Afzal Ansari , Source: (Ansari, n.d.)

## 2. Object detection using Yolo v3

### a. Yolo in practice

The second method is to use YOLO v3, Yolov3 is one of the most popular deep learning algorithms, it's fast, accurate and different.

It uses a different approach, which involves detection in a single pass. This “You only look once” algorithm pictures it in the sense that it only needs one forward propagation pass through the network to make predictions.

The procedure is as follows:

First: We need to download three files containing the configuration, the pre-trained weight file and the classes from the COCO dataset, and the classes from the COCO dataset.



Figure 8: Folders and files to download

**Second :** install: OpenCV and Numpy

**Third :** Run the code.

```
In [1]: import cv2
```

```
In [2]: import numpy as np
import time
```

```
In [5]: #Load YOLO
net = cv2.dnn.readNet("Desktop/yolov3.weights","Desktop/yolov3.cfg") # Original yolov3
#net = cv2.dnn.readNet("yolov3-tiny.weights","yolov3-tiny.cfg") #Tiny Yolo
classes = []
with open("Desktop/coco.names","r") as f:
    classes = [line.strip() for line in f.readlines()]
```

```
In [6]: print(classes)
```

```
['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop
sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpa
ck', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball
glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana',
'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'sofa', 'pottedplant', 'be
d', 'diningtable', 'toilet', 'tvmonitor', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaste
r', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']
```

```
In [7]: layer_names = net.getLayerNames()
outputlayers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

```
In [7]: layer_names = net.getLayerNames()
outputlayers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

```
In [8]: colors= np.random.uniform(0,255,size=(len(classes),3))
```

```
In [20]: cap=cv2.VideoCapture("Downloads/cars.mp4") #0 for 1st webcam
font = cv2.FONT_HERSHEY_PLAIN
starting_time= time.time()
frame_id = 0

while True:
    _,frame= cap.read() #
    frame_id+=1

    height,width,channels = frame.shape
    #detecting objects
    blob = cv2.dnn.blobFromImage(frame,0.00392,(320,320),(0,0,0),True,crop=False) #reduce 416 to 320

    net.setInput(blob)
    outs = net.forward(outputlayers)
    #print(outs[1])

    #Showing info on screen/ get confidence score of algorithm in detecting an object in blob
    class_ids=[]
    confidences=[]
    boxes=[]
```

```

#Showing info on screen/ get confidence score of algorithm in detecting an object in blob
class_ids=[]
confidences=[]
boxes=[]
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.3:
            #object detected
            center_x= int(detection[0]*width)
            center_y= int(detection[1]*height)
            w = int(detection[2]*width)
            h = int(detection[3]*height)

            #cv2.circle(img,(center_x,center_y),10,(0,255,0),2)
            #rectangle co-ordinaters
            x=int(center_x - w/2)
            y=int(center_y - h/2)
            #cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

            boxes.append([x,y,w,h]) #put all rectangle areas
            confidences.append(float(confidence)) #how confidence was that object detected and show that percentage
            class_ids.append(class_id) #name of the object tha was detected

indexes = cv2.dnn.NMSBoxes(boxes,confidences,0.4,0.6)

```

```

indexes = cv2.dnn.NMSBoxes(boxes,confidences,0.4,0.6)

for i in range(len(boxes)):
    if i in indexes:
        x,y,w,h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence= confidences[i]
        color = colors[class_ids[i]]
        cv2.rectangle(frame,(x,y),(x+w,y+h),color,2)
        cv2.putText(frame,label+" "+str(round(confidence,2)),(x,y+30),font,1,(255,255,255),2)

elapsed_time = time.time() - starting_time
fps=frame_id/elapsed_time
cv2.putText(frame,"FPS:"+str(round(fps,2)),(10,50),font,2,(0,0,0),1)

cv2.imshow("Image",frame)
key = cv2.waitKey(1) #wait 1ms the loop will start again and we will process the next frame

if key == 27: #esc key stops the process
    break;

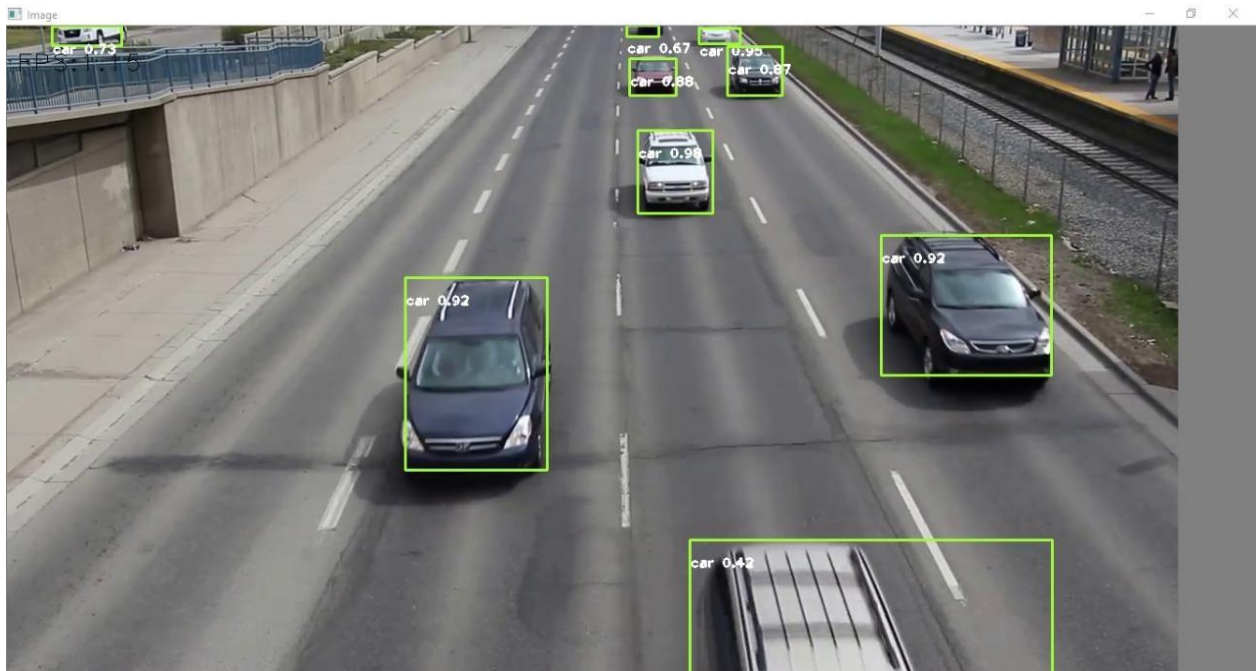
cap.release()
cv2.destroyAllWindows()

```

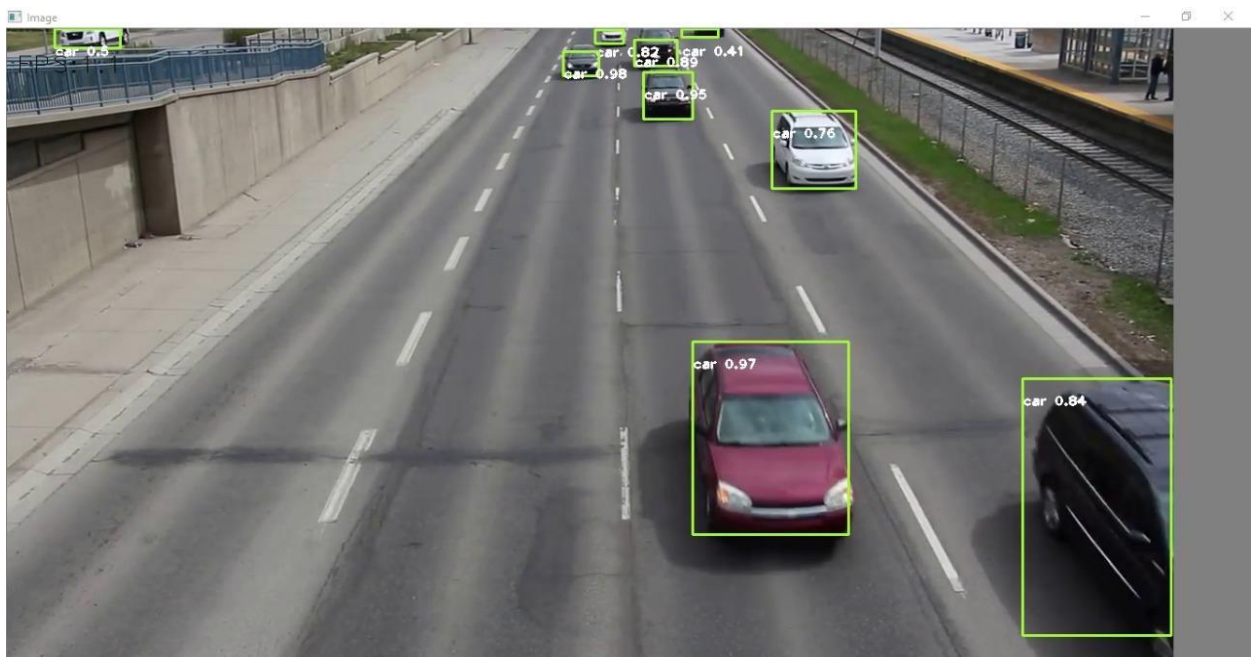
Figure 9: yolo based code



After running the program, the output is a video showing the vehicles.



*Figure 10: results of yolo detection*



*Figure 11: results of detection 2*

On remarque des deux illustrations qui sont des outputs de notre algorithme, où les objets mobiles sont bien détectés ainsi que suivies à l'aide des BOXES.

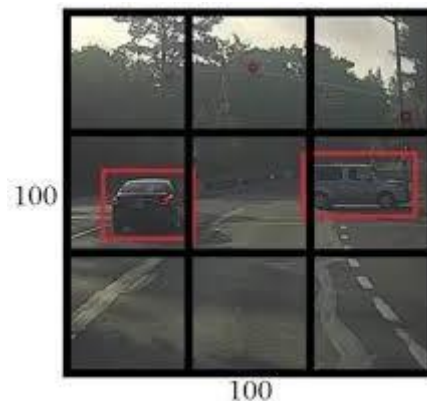
## b. Yolo in theory

In this section, we try to explain the YOLOv3 algorithm as we have understood it in the article. It stands for Detection, Classification and Localization (Bounding box).



*Figure 12: bounding box*

Yolo divides the input image into a grid  $S * S$ . Each grid predicts a single object.



*Figure 13: The detection grid*

The system predicts bounding boxes using groups of dimensions as anchor boxes. Anchor boxes are hand-selected boxes with different aspect ratios (for two-dimensional boxes) designed to match the relative ratios of the object classes to be detected.

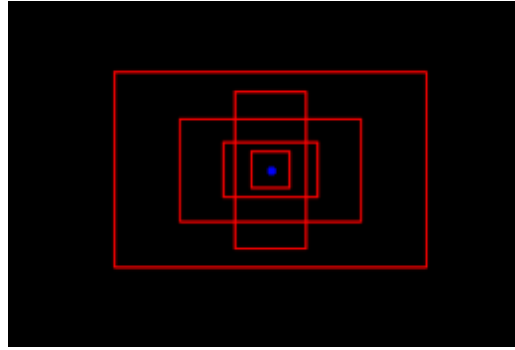


Figure 14: anchor boxes

The network predicts 4 coordinates for each bounding box  $t_x$ ,  $t_y$ ,  $t_w$ ,  $t_h$ .

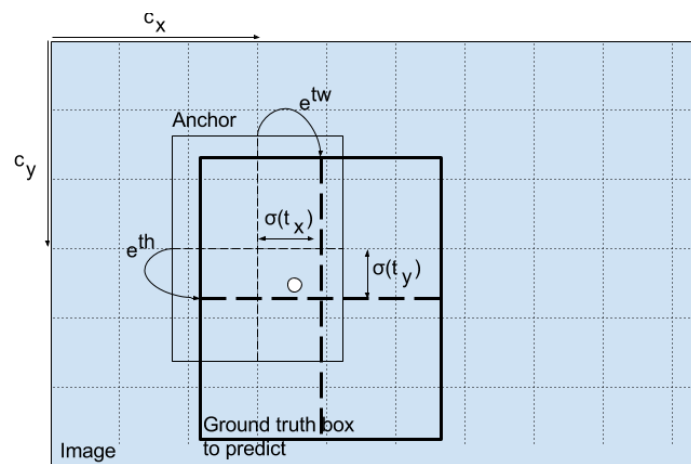


Figure 15: Boxes contact details

A sigmoid function is used to ensure that the coordinates do not exceed (1,1).

- Each box predicts the classes that the enclosing box may contain, using multi-label classification.
- The resulting output is

$$\mathbf{y} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Anchor box 1  
 Pedestrian  
 Anchor box 2  
 Car

Figure 16: results of classification M-label

The system uses k-means clustering to determine the bounding box priors. We arbitrarily chose 9 clusters and 3 scales, then distributed the clusters evenly across the scales. On the COCO dataset, the 9 clusters were:  $(10 \times 13)$ ,  $(16 \times 30)$ ,  $(33 \times 23)$ ,  $(30 \times 61)$ ,  $(62 \times 45)$ ,  $(59 \times 119)$ ,  $(116 \times 90)$ ,  $(156 \times 198)$ ,  $(373 \times 326)$ .

The feature extractor comprises 53 convolutional layers.

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

Figure 17: convolutional layers

### 3. Comparison of different algorithms

**mAP (mean accuracy) for object detection**

**AP (average precision)** is a popular metric for measuring the precision of object detectors such as Faster R-CNN, SSD, YOLO etc.

Average precision calculates the average precision value for a recall value between 0 and 1.

Sounds complicated, but it's quite simple, as an example shows. But first, let's take a brief look at precision.

### a. Precision et Recall

**Precision:** Measures the accuracy of your predictions, i.e. the percentage of your predictions that are correct.

**Recall:** Measures the extent to which you find all the positive cases. For example, we can find 80% of possible positive cases in our K best predictions.

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

*TP* = True positive

*TN* = True negative

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

*FP* = False positive

*FN* = False negative

For example, in the cancer screening test :

$$Precision = \frac{TP}{\text{total positive results}}$$

$$Recall = \frac{TP}{\text{total cancer cases}}$$

### b. IoU (Intersection over union)

The IoU measures the overlap between two boundaries. We use it to measure the degree of overlap between the predicted boundary and the ground truth (the object's actual boundary). In some datasets, we predefine an IoU threshold (e.g. 0.5) to determine whether the prediction is a true positive or a false positive.

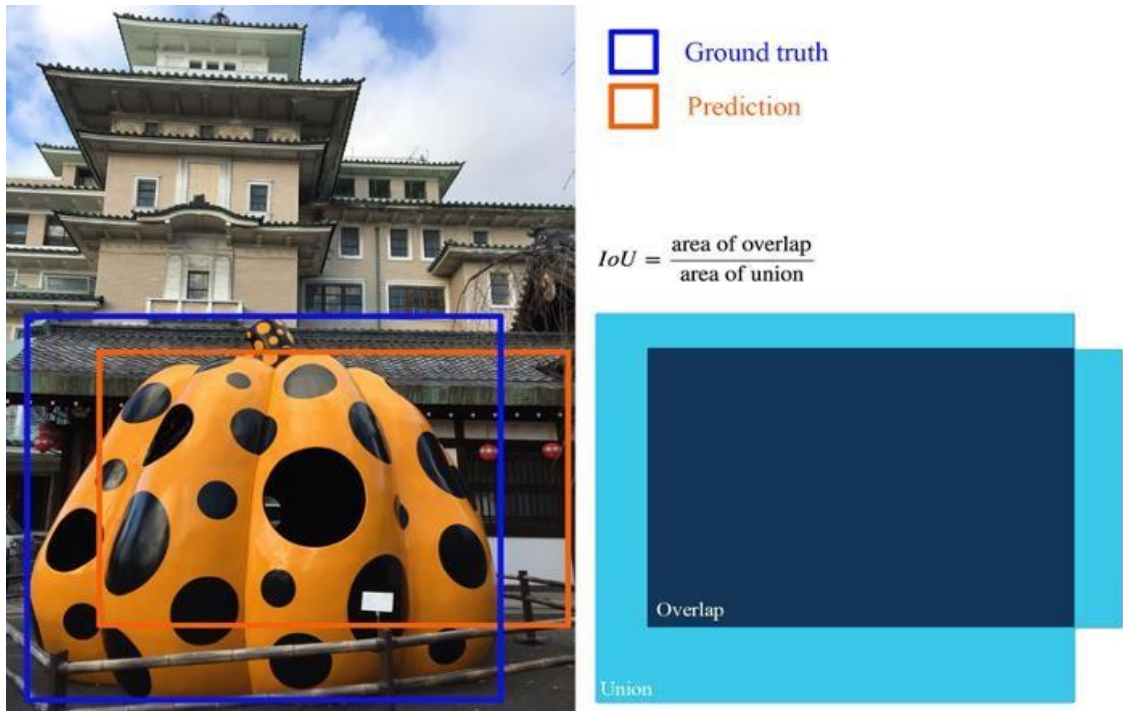


Figure 18: notion of IoU

### c. AP (Average Precision)

We'll create an oversimplified example to demonstrate the calculation of average precision. In this example, the data set contains just 5 apples. We collect all predictions made for apples in all images and rank them in descending order according to the level of confidence predicted. The second column indicates whether the prediction is correct or not. In this example, the prediction is correct if  $IoU \geq 0.5$ .

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Figure 19: Recall / Precision

Let's take the rank 3 line and first show how precision and recall are calculated.

Precision is the proportion of TP =  $2/3 = 0.67$ .

Recall is the proportion of TPs among possible positives =  $2/5 = 0.4$ .

Recall values increase as we move down the prediction ranking..

#### d. Comparing yolo and the alternatives

YOLOv3 is extremely fast and precise. In mAP measured at 0.5 IOU, YOLOv3 is on a par with Focal Loss but about 4x faster. What's more, you can easily arbitrate between speed and precision by simply changing the model size, without having to re-train.

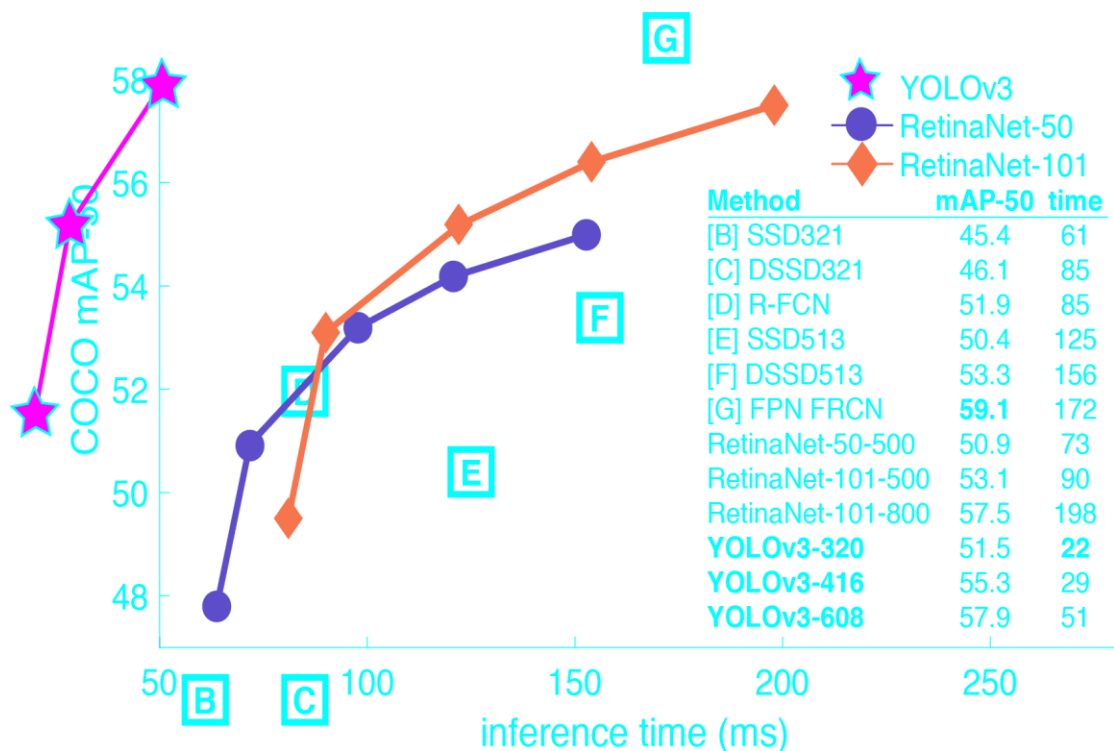


Figure 20: comparison of models

Our model has several advantages over classifier-based systems. It examines the entire image at test time, so its predictions are informed by the overall context of the image. It also makes predictions with a single network evaluation, unlike systems like R-CNN, which require thousands for a single image. This makes it extremely fast, over 1000 times faster than R-CNN and 100 times faster than fast R-CNN.



# Conclusion

In short, our project had two essential parts: a theoretical part, where we were obliged to carry out several bibliographical searches to familiarize ourselves with our subject and also to look for all the old solutions to our problem, study them and then move on to the practical part, where we implemented several artificial intelligence techniques, compared them and thus concluded which was the most optimal solution for solving our problem.

Now that the detection and tracking aspects of our work have been completed, we'll move on to the last part of our work, which is prediction.