1.1. BACKGROUND

1.1.1 DEEP LEARNING

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

1. Deep learning requires large amounts of **labeled data**. For example, driverless car development requires millions of images and thousands of hours of video.
2. Deep learning requires substantial **computing power**. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep

   learning network from weeks to hours or less.

Most deep learning methods use **neural network** architectures, which is why deep learning models are often referred to as **deep neural networks**. The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150. Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction. One of the most popular types of deep neural networks is known as convolutional neural networks **(CNN** or **ConvNet)**. A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not pretrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

1.1.2 OBJECT DETECTION

Object detection is a computer vision technique that works to identify and locate objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene. Object detection is commonly confused with image recognition, so before we proceed, it's important that we clarify the distinctions between them. Image recognition assigns a label to an image. A picture of a dog receives the label "dog". A picture of two dogs, still receives the label "dog". Object detection, on the other hand, draws a box around each dog and labels the box "dog". The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition.

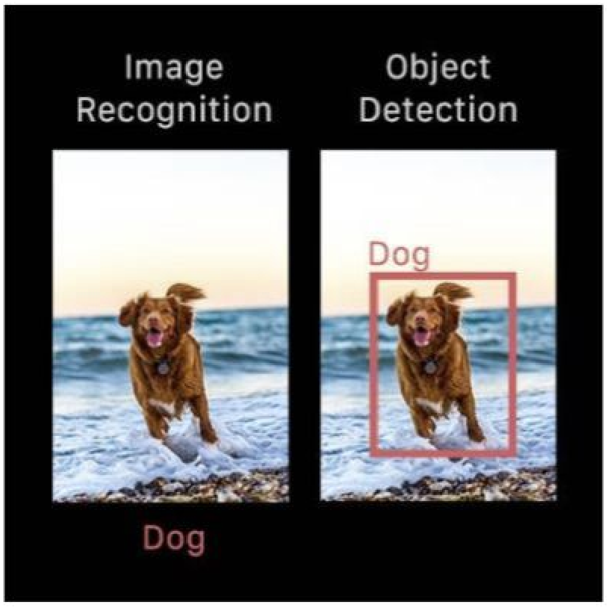

Fig 1.1 : Image recognition v/s object detection

Object detection can be broken down into **machine learning-based approaches** and **deep learning-based approaches**. In more traditional ML-based approaches, computer vision techniques are used to look at various features of an image, such as the colour histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into a regression model that predicts the location of the object along with its label. On the other hand, deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, in which features don't need to be defined and extracted separately.

An intelligent deep learning based system is implemented to detect the suspicious or missing cars using features from a set of CCTV videos captured from different locations. No human effort will be required to manually verify each and every car from the CCTV videos.

It can be used by the Police Department for carrying out their investigation fast without human intervention . It can be used in cases where there is a requirement for tracking the location of a car when it enters the city traffic, with the help of CCTVs fixed at different locations in the city.The given model can perform well, when compared to other tracking systems available.

4.3. ALGORITHMS / TECHNOLOGIES USED

4.3.1 YOLOv3

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images.

YOLO is a Convolutional Neural Network (CNN) for doing object detection. CNN's are classifier-based systems that can process input images as structured arrays of data and identify patterns between them. YOLO has the advantage of being much faster than other networks and still maintains accuracy.

It allows the model to look at the whole image at test time so its predictions are informed by the global context in the image. YOLO and other convolutional neural network algorithms "score" regions based on their similarities to predefined classes. High-scoring regions are noted as positive detections of whatever class they most closely identify with. For example, in a live feed of traffic, YOLO can be used to detect different kinds of vehicles depending on which regions of the video score highly in comparison to predefined classes of vehicles. The YOLOv3 algorithm first separates an image into a grid. Each grid cell predicts some number of boundary boxes (sometimes referred to as anchor boxes) around objects that score highly with the aforementioned predefined classes. Each boundary box has a respective confidence score of how accurate it assumes that prediction should be, and detects only one object per bounding box. The boundary boxes are generated by clustering the dimensions of the ground truth boxes from the original dataset to find the most common shapes and sizes.

Other comparable algorithms that can carry out the same objective are R-CNN (Region-based Convolutional Neural Networks made in 2015) and Fast R-CNN (R-CNN improvement developed in 2017) and Mask R-CNN. However, unlike systems like R-CNN and Fast R-CNN, YOLO is trained to do classification and bounding box regression at the same time.
YOLOv3 is fast and accurate in terms of mean average precision (mAP) and intersection over union (IOU) values as well. It runs significantly faster than other detection methods with comparable performance (hence the name – You only look once). Moreover, you can

easily trade-off between speed and accuracy simply by changing the size of the model, and no retraining required.

The YOLOv3 uses independent logistic classifiers and binary cross-entropy loss for the class predictions during training. These edits make it possible to use complex datasets such as Microsoft's Open Images Dataset (OID) for YOLOv3 model training. OID contains dozens of overlapping labels, such as "man" and "person" for images in the dataset.

YOLOv3 uses a multilabel approach which allows classes to be more specific and be multiple for individual bounding boxes. Meanwhile, YOLOv2 used a softmax, which is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. Using a softmax makes it so that each bounding box can only belong to one class, which is sometimes not the case; especially with datasets like OID.

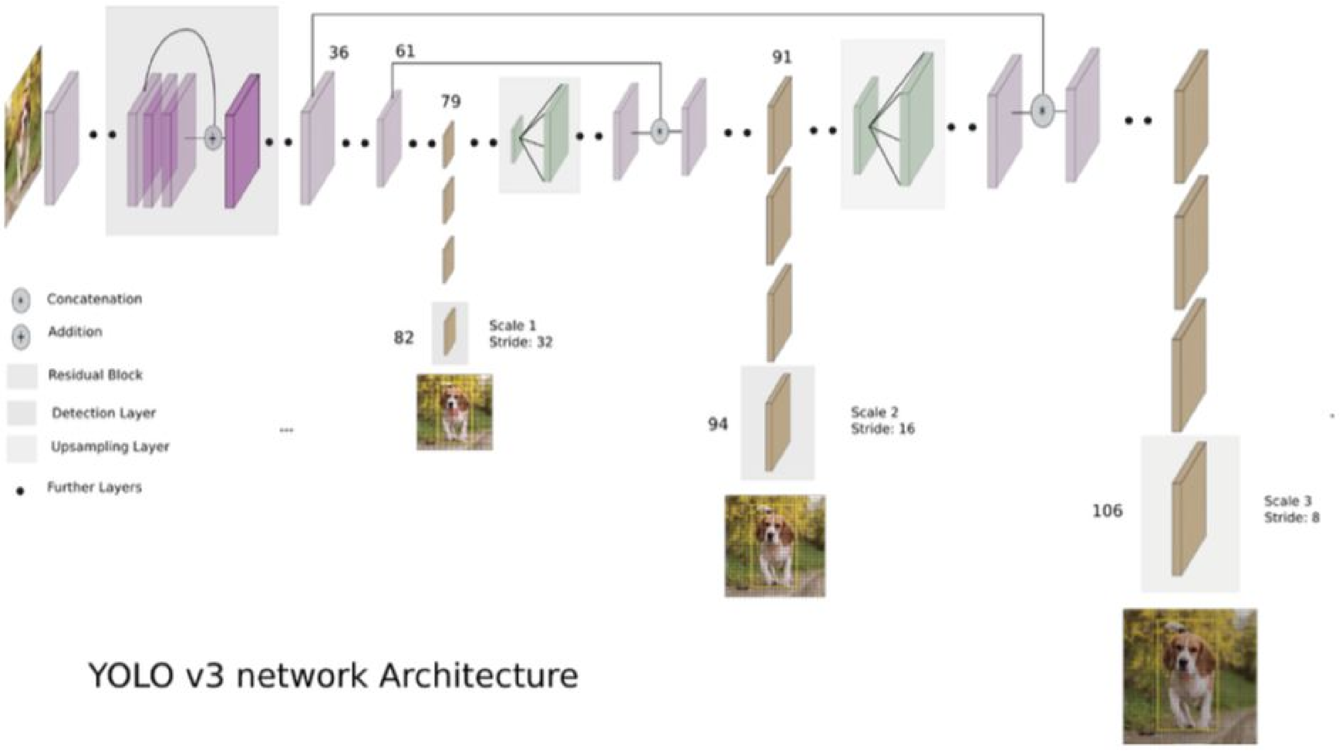

Fig 4.5: YOLO architecture

4.3.2 NON-MAXIMUM SUPPRESSION (NMS)

The purpose of non-max suppression is to select the best bounding box for an object and reject or "suppress" all other bounding boxes. The NMS takes two things into account

1. The objectiveness score is given by the model
2. The overlap or IOU of the bounding boxes.

The object is detected with the bounding boxes, the model returns an objectiveness score. This score denotes how certain the model is, that the desired object is present in this bounding box. The non-max suppression will first select the bounding box with the highest objectiveness score. And then remove all the other boxes with high overlap. This process runs iteratively until there is no more reduction of boxes.

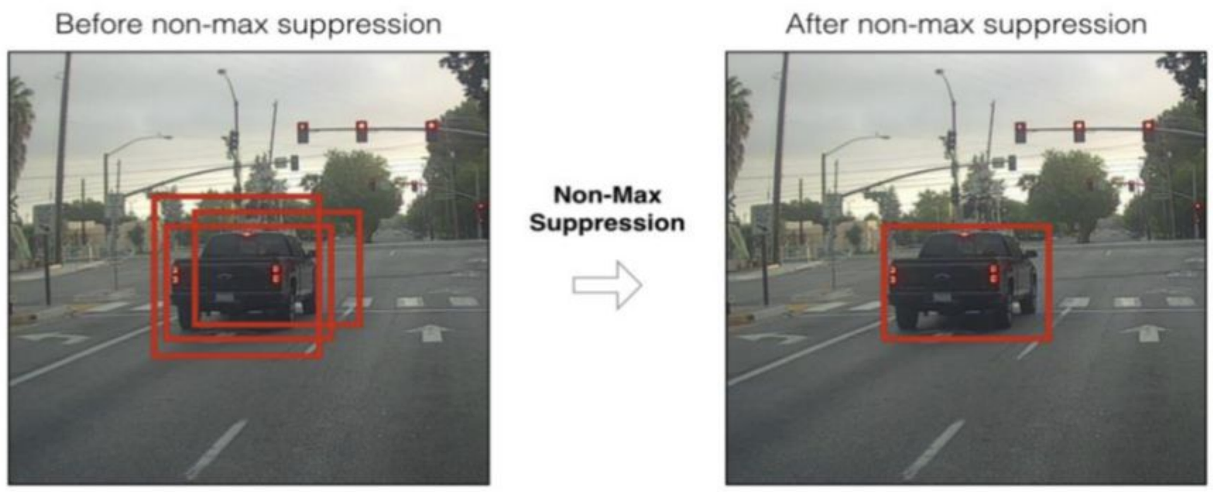

Fig 4.6: Non-max Suppression

The following is the process of selecting the best bounding box using NMS-

Step 1: Select the box with highest objectiveness score

Step 2: Then, compare the overlap (intersection over union) of this box with other boxes

Step 3: Remove the bounding boxes with overlap (intersection over union) >50%

Step 4: Then, move to the next highest objectiveness score

Step 5: Finally, repeat steps 2-4

APPENDIX I - CODE SNIPPETS

A. LOGIN PAGE

Description: This code snippet is to verify the user credentials to ensure secure login.

```
<?php
session_start();
include('connection.php');
$myusername=$_POST['username'];
$mypassword=$_POST['password'];

	$sqlss=mysqli_query($con,"SELECT * FROM reg WHERE
username='$myusername' and password='$mypassword'");
	//echo "SELECT * FROM reg WHERE
username='$myusername' and password='$mypassword'";
	$cc=mysqli_num_rows($sqlss);

	if($cc==1)
	{

	header("location:index1.php");

	}
	else{

		header("location:index.php?a=1");

	}

?>
```

B.SIGN UP PAGE

Description: This code snippet is to SIGN UP in to the system

```
<?php
 error_reporting(0);
 ?>



 <html>
 <head>
  <meta name="viewport" content="width=device-width"
 content="initial-scale=1">
  <link href="css/bootstrap.min.css" rel="stylesheet">
  <link href="css/log.css" rel="stylesheet">
 </head>
 <body>
   <div id="p3">
   <div class="container" id="id1">
   <div class="row-justify-content-center">
   <div class="col-sm-5" style="margin-top:10px
 !important;">
      <h1><center>AUTOTRACK</center></h1><br>
      <form action="insert.php" method="POST"
 enctype="multipart/form-data">
      <div id="t1">
        Name : <input type="text" name="name"  required
 id="ip1" ><br>
        Email : <input type="email" name="email"  required
 id="ip1" ><br>
     User Name : <input type="text" name="username"
 required id="ip1" ><br>
       Password : <input type="password" name="password"
 required id="ip1" ><br>
```

```
     </div>

     <div class="text-center"><input type="Submit"
name="Submit" value="Submit" style="height: 40px; width:
250px" id="p1"></div><br>

   </form>

  <form>

     <div class="text-center"><input type="Submit"
value="Login" formaction="index.php"style="height: 40px;
width: 250px" id="p2"></div><br>

    </form>

   </div>

   </div>

   </div>

   </div>

   </div>

</body>

</html>
```

CSS

```
body {

      font: bold 11px/1.5em Verdana;

      }


h1 {

      font-family:Verdana, Arial, Helvetica, sans-serif;

      font-size:16px;

      font-weight:bold;

      margin:0;

      padding:0;

      }
```

```
hr {
      border:none;
      border-top:1px solid #CCCCCC;
      height:1px;
      margin-bottom:25px;
      }

#header {
             width: 98%;
             height: 100px;
             margin-left: 15px;
             margin-right: 25px;
             margin-top: 20px;
             margin-bottom:10px;
             background: #121212;
             font-family:monospace;
             font-size:32px;
             padding-top: 55px;
             border-bottom-left-radius: 20px;
             border-bottom-right-radius: 20px;
             border-top-left-radius: 20px;
             border-top-right-radius: 20px;
             font-style: italic;
         }
#container {
             width: 1000px;
             height: auto;
             margin-left: 155px;
             margin-right: 100px;
             margin-top: 25px;
```

```
            border-colour: #121212;
            border-width: 1px;
            border-style: solid;
            border-top-left-radius: 20px;
            border-top-right-radius: 20px;
            border-bottom-left-radius: 20px;
            border-bottom-right-radius: 20px;
        }
.contents{
            padding-top: 40px;
            padding-left: 40px;
            padding-right: 20px;
            padding-bottom: 40px;
            font-family: monospace;
            font-size: 14px;
        }


/*input[type="file"]{
          width: 240px;
          height: 38px;
          font-family:Arial, "Helvetica", sans-serif;
          font-size:12px;
          font-weight:bold;
          colour:#000;
          text-decoration:none;
          text-transform:uppercase;
          text-align:center;
          text-shadow:1px 1px 0px #37a69b;
          padding-top:6px;
              padding-bottom:3px;
          margin-left: 10px;
          position:relative;
          cursor:pointer;
          border: none;
          border-top-left-radius: 5px;
          border-top-right-radius: 5px;
          border-bottom-right-radius: 5px;
          border-bottom-left-radius:5px;
        }*/
input[type="submit"],[type="reset"]{
  width:240px;
  height:35px;
  display:block;
  font-family:Arial, "Helvetica", sans-serif;
  font-size:12px;
  font-weight:bold;
  colour:#fff;
  text-decoration:none;
  text-transform:uppercase;
  text-align:center;
  text-shadow:1px 1px 0px #37a69b;
  padding-top:6px;
  position:relative;
  cursor:pointer;
  border: none;
  background-colour: #121212;

  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
  border-bottom-right-radius: 5px;
  border-bottom-left-radius:5px;
```

```
}
input[type="submit"]:active {
  top:3px;
  box-shadow: inset 0px 1px 0px #2ab7ec, 0px 2px 0px 0px
#31524d, 0px 5px 3px #999;
}
input[type="text"],select, textarea, [type="number"],
[type="password"], [type="file"]{
  border: 1px solid #78AB46;
  border-radius: 4px;
  box-shadow: 0 1px #fff;
  box-sizing: border-box;
  colour: #696969;
  height: 39px;
  //margin: 31px 0 0 29px;
  padding-left: 37px;
  transition: box-shadow 0.3s;
  width: 240px;
}
input[type="text"]:focus {
  box-shadow: 0 0 4px 1px rgba(55, 166, 155, 0.3);
  outline: 0;
}


input[type="password"]:focus {
  box-shadow: 0 0 4px 1px rgba(55, 166, 155, 0.3);
  outline: 0;
}
#tabs {
      float:left;
      width:100%;
```

```
        //background:#EFF4FA;
          background: #FFFFFF;
        font-size:93%;
        line-height:normal;
        border-bottom:1px solid #121212;
        }

    #tabs ul {
        margin:0;
        padding:10px 10px 0 50px;
        list-style:none;
        }

    #tabs li {
        display:inline;
        margin:0;
        padding:0;
        }

    #tabs a {
        float:left;
        background:url("tableft.gif") no-repeat left top;
        margin:0;
        padding:0 0 0 5px;
        text-decoration:none;
        }

    #tabs a span {
        float:left;
        display:block;
        background:url("tabright.gif") no-repeat right top;
```

```
      padding:5px 15px 4px 6px;
      colour:#FFF;
      }

/* Commented Backslash Hack hides rule from IE5-Mac \*/
#tabs a span {float:none;}

/* End IE5-Mac hack */
#tabs a:hover span {
      colour:#FFF;
      }

#tabs a:hover {
      background-position:0% -42px;
      }

#tabs a:hover span {
      background-position:100% -42px;
      }

#c{
              display:inline-block;
        display: inline;
          margin-right:10px;
      }
```

C. INPUT PAGE

Description: This code snippet is to give inputs like model, colour and license in to the system.

```
<?php
```

```
$target_path = "uploads/";


$target_path = $target_path . basename(
$_FILES['uploadedfile']['name']);

$myFile ="input.txt";

$fh = fopen($myFile,'w') or die("can't open file");

fwrite($fh,$target_path);

fclose($fh);


$data=$_POST['model'].",".$_POST['colour'];

$myFile ="colour.txt";

$fh = fopen($myFile,'w') or die("can't open file");

fwrite($fh,$data);

fclose($fh);



if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'],
$target_path)) {

$python=`python car_model_detectionn.py`;

header("location:index1.php");


} else{

    echo "There was an error uploading the file, please try
again!";

}

?>
```

D. UPLOAD AND DETECTION PAGE

Description: This code snippet is to upload the video file and to detect the vehicle in system.

MODEL DETECTION

Description: This code is to detect the model of the vehicle from the input video file.

```
import glob
import random


# Load Yolo
net = cv2.dnn.readNet("yolov3_custom_last.weights",
"yolov3_custom.cfg")

# Name custom object
classes = []
with open("classes.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in
net.getUnconnectedOutLayers()]
colours = np.random.uniform(0, 255, size=(len(classes),
3))
# Images path
#images_path =
glob.glob(r"F:\aumento_test\apple\apple_testing\*.png")

# Insert here the path of your images
#random.shuffle(images_path)

cap = cv2.VideoCapture('1_1.mp4')

# loop through all the images
#for img_path in images_path:
while(cap.isOpened()):
    # Loading image
    #img = cv2.imread(img_path)
    ret, frame = cap.read()
    if (ret!=True):
        break
    img=frame
    img = cv2.resize(img, None, fx=0.4, fy=0.4)
    height, width, channels = img.shape
    # Detecting objects
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416,
416), (0, 0, 0), True, crop=False)
    print("First Blob: {}".format(blob.shape))
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Showing informations on the screen
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            #class_id = np.argmax(scores)
# sort the probabilities (in descending) order, grab the
index of the
# top predicted label, and draw it on the input image
            class_id = np.argsort(scores)[::-1][0]
            confidence = scores[class_id]
            if confidence > 0.5:
                # Object detected
                print(class_id)
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5,
0.4)
    print(indexes)
    #font = cv2.FONT_HERSHEY_PLAIN
    font = cv2.FONT_ITALIC
    for i in range(len(boxes)):
        if i in indexes:
                x, y, w, h = boxes[i]
                label = str(classes[class_ids[i]])
                print(label)
                colour1 = colours[class_ids[i]]
                colour=(255,0,0)
                cv2.rectangle(img, (x, y), (x + w, y + h),
    colour, 2)
                cv2.putText(img, label, (x, y + 30), font,
    0.5, colour, 2)


        cv2.imshow("Image", img)
        #key = cv2.waitKey(0)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cv2.destroyAllWindows()
cap.release()
```

COLOUR DETECTION

Description: This code snippet is for detecting the colour and model of the vehicle.

```
    import cv2
    import numpy as np
    import pandas as pd
    import glob
    import random
    # Load Yolo
    net = cv2.dnn.readNet("yolov3_custom_last.weights",
    "yolov3_custom.cfg")
    classes = []
    with open("classes.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in
    net.getUnconnectedOutLayers()]
    colours = np.random.uniform(0, 255, size=(len(classes),
    3))
    cap = cv2.VideoCapture('4.mp4')
    #save video
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
```

```
size1 = (frame_width, frame_height)
result =
cv2.VideoWriter('out.avi',cv2.VideoWriter_fourcc(*'MJPG'
),10, size1)
# Images path
#images_path =
glob.glob(r"F:\aumento_test\bavan_sir_mail\car_colour\*.
png")
#random.shuffle(images_path)

#for img_path in images_path:
# Loading image
while True:
    #img = cv2.imread(img_path)
    ret, frame = cap.read()
    if (ret!=True):
        break
    img=frame
# Insert here the path of your images
# Loading image
#img = cv2.imread("5.png")
    img = cv2.resize(img, None, fx=0.4, fy=0.4)
    height, width, channels = img.shape

# Detecting objects
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416,
416), (0, 0, 0), True, crop=False)

    net.setInput(blob)
    outs = net.forward(output_layers)

# Showing informations on the screen
    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 > 0.5:
            # Object detected
                center_x = int(detection[0] * width)
```