Pedestrian Motion Detection using YOLO v3 Algorithm

Sethuselvi M[1], Dr.S.Kayalvizhi[2]

[1]*M.E, Dept of Computer Science and Engineering, Easwari Engineering College, Chennai, India*

[2]*Dept of Computer Science and Engineering, Easwari Engineering College, Chennai, India*

Abstract **- Vehicular accidents claim millions around the world every year most of which that are caused due to human-error. The Autonomous Vehicles are believed to bring solution to the problem by automating the use of vehicles. The autonomous vehicles have a very important feature called pedestrian detection to detect pedestrians on the road and stop in the appropriate places. Many multi-person trackers have been proposed for real-time detection of pedestrians but due to inaccurate predictions and long training time they have not been efficient. In this paper pedestrians are detected using yolo v3 algorithm on the COCO dataset which consists of sequence of images. The images are trained and the output is identified with the help of convolutional neural networks in yolo architecture.**

Index Terms **- PMD, YOLO, RCNN, CNN, IOU, NMS.**

I.INTRODUCTION

The advent of Machine Learning has brought about significant changes in the way technology is viewed. Many industries today are heavily relying upon Human-Computer Interaction by automating tedious jobs that are error prone when manually handled. Amongst many industries, the automobile industry is the most to benefit from automation as vehicular accidents claim millions around the world every year and are caused due to human errors. Autonomous Vehicles are believed to solve this problem by automating the use of vehicles where Machine Learning comes into play with less human interference.

The Automobile industry makes use of a Computer Vision technique called Object Detection, which locates existing objects. According to Computer Vision, humans and animals are considered as objects. Various applications of Object Detection include intelligent video surveillance for detecting suspicious activities such as terrorism, robbery etc. Various Machine Learning algorithms have been developed to achieve object detection.

Autonomous Vehicles adapt the Object Detection technique to detect the pedestrians moving along the streets and crossing of roads. Detection of pedestrians is crucial for any vehicle moving on the road to avoid terrible accidents due to carelessness. These accidents are avoided by stopping of the moving vehicles when it detects pedestrians.

In real time, pedestrians are captured using cameras and detection is performed by drawing a bounding box around them a probability. Although, a major problem faced by pedestrian detectors is occlusion in which a person may not be entirely visible due to presence of opaque objects in the scene leading to inaccurate detections. Many car manufacturers (e.g., Tesla, Volvo, Ford) are extensively working vehicle automation to improve the safety systems.

A. Pedestrian Motion Detection:

Pedestrian Motion Detection is widely used in Autonomous Vehicles to identify the pedestrians on the road and stop the vehicle when necessary. In automatically driven vehicles, pedestrian detectors play an important role as humans are error prone to making mistakes while driving due to various reasons (driving under the influence of alcohol) because of which the safety of the driver and the pedestrian are under great risk. The PMD ensures to provide reliable detection of pedestrians on road by cumulating the data from various environments to analyse different situations using which a pedestrian can be identified. The accumulation of images is extremely helpful in building an accurate and fast detector. Various Machine Learning models are then applied to distinguish a pedestrian among various other objects such as tree, handbags, cycle etc. that could be present in the scene. A detection is made using a bounding box of rectangular in shape and also the percentage probability. PMD also makes sure that a pedestrian is identified despite any occlusions by an object, appearance changes or rapid camera movement to build an occlusion aware pedestrian motion detector.

B. Characteristics of proposed pedestrian detector:
The dataset consists of image sequences that are divided into training and testing sets in unconstrained environments filmed with static and moving cameras. Various Deep Learning algorithms (based on Convolutional Neural Network) are used that have been pre-trained already, i.e. they come with pre-trained weights that has already been trained with another dataset. A confidence threshold is set so that the final detections are a result of satisfying the threshold. The detections are in the form of rectangular bounding boxes with top-left and bottom-right co-ordinates. Additionally, the output also consists of the class label, the percentage that indicates how accurately the object has been identified and rectangular box co-ordinates. The accuracy of the model is determined based on the number of detections using the statistical accuracy formula that depends upon the number of true positives, true negatives, false positives and false negatives. Approximate number of people in the image is calculated along with the number of detected persons which is used to calculate accuracy of the pre-trained model.

A general pedestrian motion detector follows the steps below:
1. Loading and resizing of the image.
2. Loading the pre-trained model trained using the COCO dataset.
3. Testing the image on the pre-trained model.
4. The detections in the image are shown along with the number of detections, Class label, percentage probabilities.

C. Significance of YOLO Algorithm:
Object detection which uses R-CNN family of techniques primarily use regions to localize the objects within the image. The network does not look at the entire image, only at the parts of the images which have a higher chance of containing an object. The YOLO framework takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is that it incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation.

II.YOLO v3 ALGORITHM

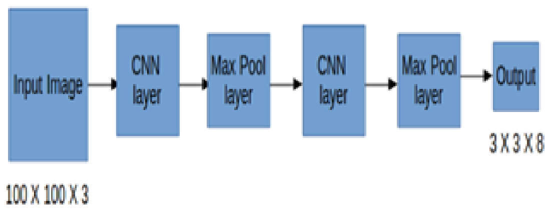


Fig.1. Architecture of YOLOv3

A. Architecture of YOLO v3 Algorithm:
To train the model both forward propagation is used. During the testing phase, image is passed to the model and run forward propagation until we get an output y. Here 3 X 3 grid taken, but generally in real-world scenarios grids which are larger than 19x19 is used. Even if an object spans out to more than one grid, it will only be assigned to a single grid in which its mid-point is located. It can reduce the chances of multiple objects appearing in the same grid cell by increasing the more number of grids.

B. Workflow of YOLO Algorithm
- Reading input image
- Loading YOLO v3 Network
- Getting Bounding Boxes
- Non-maximum Suppression
- Drawing Bounding Boxes with Labels

Reading input image: The input image from the dataset is read and the spatial dimension of the image is obtained. The image is then resized to a smaller dimension to avoid longer training time and excess memory usage as this can be a hinder to the performance of the object detection which is then fed into the appropriate model. The datasets used have annotations of objects by hand labelling bounding boxes on each object visible in the image which is then converted into a CSV file to find the IOU of the objects which helps in estimating the probability of the object detected during testing.
Loading YOLO v3 Network: The models that have been drawn comparisons of in this project are YOLO v3, RetinaNet, Mask RCNN. Each of these models use a backbone network of multiple layers to extract the feature map from the input image upon which the rest of the network is based on. YOLO v3 uses darknet-53

architecture which has 53 layers supplemented with an additional 53 layers summing up to 106 layers overall. The pre-trained model of the darknet-53 on COCO dataset is used here which contains a configuration file and weights file. The configuration file contains the details of the layers used in the architecture and the weights file contain the corresponding weights obtained after training on the COCO dataset.

Getting Bounding Boxes: After the model is loaded the class ids and the confidence scores of the pedestrians who are detected in the image are obtained. The confidence scores of the pedestrians are estimated based upon the IOU value which outputs intersection probability of the bounding boxes labelled in the dataset to the bounding box of the detected pedestrian. A threshold is maintained for finding the detected pedestrians whose confidence scores are higher than the threshold value.

IOU= Area of intersection/Area of union

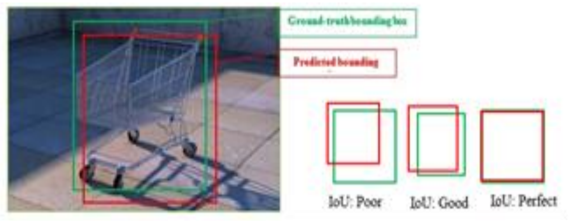


Fig.2.IOU Detection of an Object

In YOLO v3 the bounding box co-ordinates are obtained by splitting up the image into an S x S grid where for each object present in the image one grid cell is responsible for predicting it which is the grid cell in which the object falls into, that is the center of the object. The bounding box predictions have 4 components which is (centerX, centerY, width, height) where centerX and centerY are the co-ordinates of the center position of the box. The center co-ordinates are then used to find the top-left and bottom-right co-ordinates of the bounding box using the formula:

x = centerX - (width / 2)

y = (centerY - (height / 2)

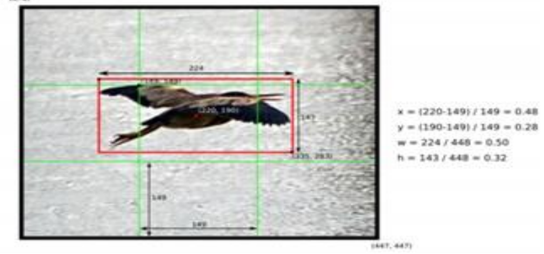


Fig.3.Calculating the Bounding Box coordinates of an Object

Non-Max Suppression: After the bounding box co-ordinates are applied to the detected pedestrian the week and the overlapping bounding boxes are eliminated by applying non-maxima suppression. Non-maxima suppression is first used to detect the overlapping bounding boxes and then these are removed by finding all the predicted bounding boxes that have a detection probability that is less than a given non-maxima suppression threshold to obtain the best bounding box.

In YOLO v3 the non-maxima suppression is applied to eliminate the bounding boxes which are overlapping in the grids, and this is done by choosing the IOU value of the detected pedestrian.

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

- Select the box with highest objectiveness score.
- Then, compare the overlap (intersection over union) of this box with other boxes.
- Remove the bounding boxes with overlap (intersection over union) >50%
- Then, move to the next highest objectiveness score.
- Finally, repeat steps 2-4

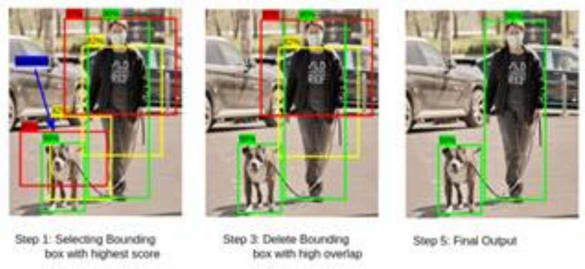

Step 1: Selecting Bounding box with highest score
Step 3: Delete Bounding box with high overlap
Step 5: Final Output

Fig.4.Detection of pedestrian using yolo v3 after removing IOU and NMS

Drawing Bounding Boxes with Labels: The output of the detections is identified as a rectangular bounding box with class labels and appropriate detection score with the number of people who are detected in an input image.

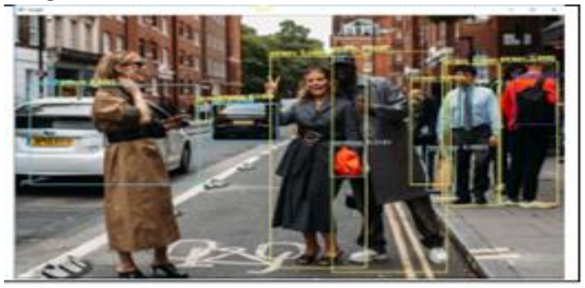
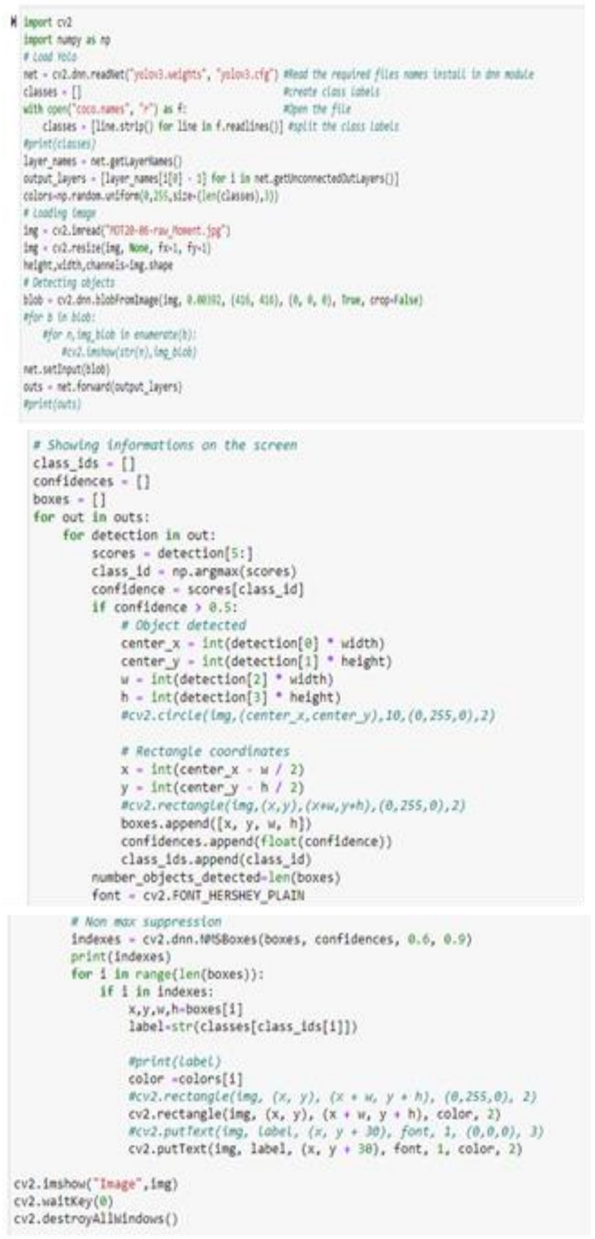


Fig.5Detection of Multiple pedestrians using yolo v3

III.IMPLEMENTATION

This section explains about the implementation details Dataset used-COCO dataset Image used-MOT20-06 raw image



```
import cv2
import numpy as np
# Load Yolo
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg") #Read the required files names install in dnn module
classes = []                                           #create class labels
with open("coco.names", "r") as f:                     #Open the file
    classes = [line.strip() for line in f.readlines()] #split the class labels
#print(classes)
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors=np.random.uniform(0,255,size=(len(classes),3))
# Loading image
img = cv2.imread("MOT20-06-raw_Moment.jpg")
img = cv2.resize(img, None, fx=1, fy=1)
height,width,channels=img.shape
# Detecting objects
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
#for b in blob:
    #for n,img_blob in enumerate(b):
        #cv2.imshow(str(n),img_blob)
net.setInput(blob)
outs = net.forward(output_layers)
#print(outs)
```

```
# 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)
            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 coordinates
            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])
            confidences.append(float(confidence))
            class_ids.append(class_id)
        number_objects_detected=len(boxes)
        font = cv2.FONT_HERSHEY_PLAIN
```

```
        # Non max suppression
        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.6, 0.9)
        print(indexes)
        for i in range(len(boxes)):
            if i in indexes:
                x,y,w,h=boxes[i]
                label=str(classes[class_ids[i]])

                #print(label)
                color =colors[i]
                #cv2.rectangle(img, (x, y), (x + w, y + h), (0,255,0), 2)
                cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                #cv2.putText(img, label, (x, y + 30), font, 1, (0,0,0), 3)
                cv2.putText(img, label, (x, y + 30), font, 1, color, 2)

cv2.imshow("Image",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig.6.Implementation of yolo v3

The above figures show the implementation of yolo algorithm in python.

IV.STIMULATION OF RESULT

Following shows the result of image of the detections of some persons in the given image.

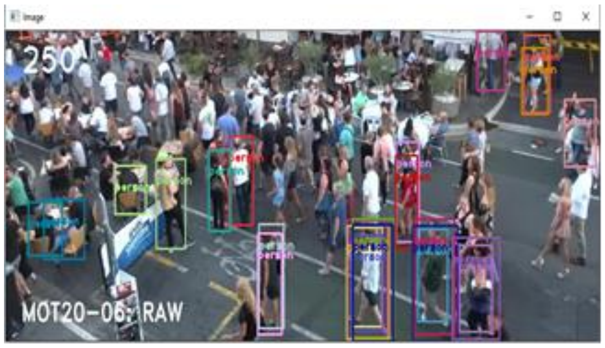


Fig.6.Output of yolo v3 in MOT20-06 image dataset

V.CONCLUSION AND FUTURE WORK

YOLO object detection is able to detect only few pedestrians. The main challenge is detection of persons in a crowd. Hence these algorithms are in research as when image dataset improves in future. Various algorithms are also being proposed to increase number of detections of persons for best accurate results.

VI.ABBREVIATIONS

PMD – Pedestrian Motion Detector
YOLO – You Only Look Once
RCNN – Region Based Convolutional Neural Network
CNN – Convolutional Neural Network
IOU – Intersection over Union
NMS-Non-Max Suppression

VII.ACKNOWLEDGEMENT

I would like to thank Dr. S Kayalvizhi mam for guiding me to write this paper. I would also like to show my gratitude towards Easwari Engineering College for giving an opportunity to write this paper and providing such a friendly environment for doing research and writing this paper.

REFERENCE

[1] Survey of Pedestrain Detection with Occlusion Chen Ning,Li Menglu,Yuan Hao,Su Xueping,Li Yunhong Complex &Intelligent Systems, Springer Volume 7,pages .577- 587(2021)
[2] Weighted boxes fusion: Ensembling boxes from different object detection models Roman Solovyev Institute for Design Problems in

Microelectronics of Russian Academy of Sciences https://arxiv.org/abs/1910.13302

[3] YOLO v3-Tiny: Object Detection and Recognition using one stage improved model, Pranav Adarsh, Pratibha Rathi, Manoj Kumar. 2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS)

[4] YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications Chethan Kumar B, Punitha R, Mohana Proceedings of the Third International Conference on Smart Systems and Inventive Technology (ICSSIT 2020)