A
Major Project
On

SOCIAL DISTANCE PREDICTOR USING DEEP LEARNING

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING
By

K.Ankith Sai Reddy (177R1A0522)
Ch. Dilip Goud (177R1A0507)
V. Prashanth (177R1A0552)

Under the Guidance of
D. Vigneswar Rao
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CMR TECHNICAL CAMPUS
UGC AUTONOMOUS
Accredited by NAAC, NBA, Permanently Affiliated by JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGC ACT. 1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.
2017-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**SOCIAL DISTANCE PREDICTOR**" being submitted by **K.Ankith Sai Reddy (177R1A0522), V. Prashanth (177R1A0552), Ch. Dilip Goud(177R1A0507),** in partial fulfillments of the requirements for the award of the degree of B.Tech in Computer Science and Engineering of the Jawaharlal Nehru Technological University Hyderabad, during the year 2020-2021. It is certified that they have completed the project satisfactorily.

DR. M. VARAPRASAD RAO **INTERNAL GUIDE**	DR. A. RAJI REDDY **DIRECTOR**
DR. K. SRUJAN RAJU **HOD**	**EXTERNAL EXAMINER**

Submitted for Viva-voce Examination held on _______________

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We take this opportunity to express my profound gratitude and deep regard to my guide: **D. Vigneswar Rao**, Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) Coordinators: **Mr. J. Narasimha Rao, Mr. B. P. Deepak Kumar, Mr. K. Murali, Dr. Suwarna Gothane and Mr. B. Ramji** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to the Head of the Department **Dr. K. Srujan Raju** for providing excellent infrastructure and a nice atmosphere for completing this project successfully. We are obliged to our Director **Dr. A. Raji Reddy** for being cooperative throughout the course of this project. We would like to express our sincere gratitude to our Chairman **Sri. Ch. Gopal Reddy** for his encouragement throughout the course of this project.

The guidance and support received from all the members of **CMR TECHNICAL CAMPUS** who contributed and who are contributing to this project, was vital for the success of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement without which this assignment would not be possible. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project.

K.Ankith Sai Reddy (177R1A0522)
V. Prashanth (177R1A0552)
Ch.Dilip Goud (177R1A0507)

ABSTRACT

The objective of the model is to make the detection of social distance between people who are roaming on the roads and the footpaths or anywhere outdoors or in the society easy and more reliable. We know that when a naked human eye monitors the, accuracy is considerably low and also a human eye cannot monitor more than one group at a time. Therefore this system is introduced to make the process easy, accurate and consistent in monitoring people at all times by using 24/7 surveillance cameras.

This system is using the YoloV3 (You Only Look Once) which is the fastest object detection algorithm and contains several classes of objects for object detection. This algorithm is not only used for object detection but also used in finding the position of the object and tracking the movement of the object from one position to another. This feature will be used in finding the distance of the objects whose coordinates will further help in finding out the distance between each object in real time and determine whether they are following social distancing or not.

We also have a feature in YoloV3 to train the algorithm using custom datasets, since we are only using the algorithm for human subjects, we are going to train the data set based on human’s classification. We train the algorithm using COCO. COCO is a large-scale object detection, segmentation and captioning data set and has several other features which help us train the algorithm for the detection of objects.

In this system the prediction of the distance between objects in real time is taken place by using methods like Euclidean distance and centroid between the coordinates of the objects. The coordinates are given by the YoloV3 algorithm.

The system is designed to make the monitoring of the social distance easy, accurate and consistent at all times without using any means of manpower and by simply providing 24/7 surveillance cameras. This reduces human cost and at the same time increases the ability to monitor more people at a time maintaining the accuracy. All this combined makes this system more effective, efficient and accurate.

LIST OF FIGURES

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROJECT SCOPE

The project titled as “Social Distance Predictor” is a Machine Learning based application which trains itself and improves every time on new scenarios. This project aims to design a social distance predictor which verifies the distance between two or more people is safe and is according to social distance norms provided by the government body. This Project helps monitor social distance between people in real time and help admins warn the people at public places. This system is more efficient, effective and accurate. At these hard times of covid, this helps in reducing immense man power for monitoring people and also drastically increases the accuracy to help people maintain social distance.

1.2 PROJECT PURPOSE

Covid - 19 is a deadly and a very dangerous disease which is causing chaos in every country and every place possible in this world. There have been strict rules created by the national body and world organisations to help prevent this deadly virus from spreading into communities. One of which is to maintain Social Distance, people must maintain a distance of 1.5 Meters between each other in order to maintain social distance. We know that in public places where there is a huge crowd it is difficult to monitor people who maintain social distance especially in areas where there is dense population. In order to do so we have created this system which is based on machine learning model where it measures the social distance between two or more people in real time with the help of CCTV Cameras. The live footage from the CCTV cameras will go to the system where the system processes the social distance between two people and sends the output video to the admin and thus the admin could announce and take preliminary precautions.

1.3 PROJECT FEATURES

Our proposed system aims to design a Social distance predictor based on computer vision that can adapt to different crowd situations. This system uses YOLO object detection algorithms which is trained on the COCO dataset which is classified on human subjects. Therefore YOLO, according to its trained dataset detects the human subjects and gives the centroid coordinates of the human subjects and our program takes the coordinates of the centroid and measures the social distance between two or more centroids by calculating the Euclidean distance between them. Thus the system shows the classified human subjects in boxes and the color is given based on social distance, that is the human subjects those who maintain the social distance are classified in green boxes and those who don’t in red boxes thus indicating the admin of who and who is not maintaining the social distance regulations.

2. SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role to clean the data-bases on an everyday basis so that every day we have a clean database of 0 leaves approved and no leaves from the previous day are present in the present day.

2.1 PROBLEM DEFINITION

The problem of the existing system is, in these tough times of covid - 19 social distancing is a very important precaution to be taken by the public especially in the public places therefore the crowd will only follow the norms if they are always been warned and monitored. The traditional system initially was to keep the guards at the public place or the police men to maintain and keep a watch on the public of whether they are maintaining social distance and wearing masks, now there is a huge crowd everywhere and we need a large force which consists of more personnel to monitor and warn the crowd. Now also a human can only monitor a few people who are in his field of vision, and chances are not all people are warned by the personnel. Also the accuracy has been compromised since human beings cannot exactly measure the distance between two people. Therefore all these limitations have been solved in the proposed system. We have developed a system which would solve accuracy, require less personnel and is effective and efficient and uses computer vision and machine learning object detection which helps us detect the human subjects and thus helps us find out social distance between two or more people using algorithmic formulas.

2.2 EXISTING SYSTEM

In the existing system only we require more police personnel and more manpower in order to monitor people who are not maintaining social distance. Now there are many ways which the government bodies have defined to monitor social distance between people.

- **Police personnel -** The police personnel are taking care in each area checking people who and who are not maintaining social distancing in public places and even in the vehicles it has been mandatory to maintain social distancing that is only the driver and one passenger sitting diagonal to the driver is allowed to travel that too in an urgent situation.
- **Lockdowns -** most of the public places have been locked down for a few days and only the most essential shops have been open to take groceries and they have also been limited to be open until a certain time.
- **Foot Marks -** Now if someone has travelled in a public transport these days they might have noticed that there are footmarks and cross marks on the seats

indicating that people must follow social distance and only stand there where there are footmarks and only sit there where there is no cross mark on the seat.

- **CCTV Surveillance:** Nowadays in public places there is 24/7 surveillance running and people are being manually pointed out by the admin who is sitting in the control center of the cameras.

LIMITATIONS:

- More man power is required
- Time consuming for the person who monitors the people
- Needs to follow a lot of additional precautions
- Need to maintain a record of how many people are not maintaining social distancing
- Less accurate in real time
- Less efficient due to lack of monitoring of huge numbers of people in the crowded places and could only focus on those who are in the field of vision of the personnel monitoring the social distance maintenance.

2.3 PROPOSED SYSTEM

The aim of the proposed system is to solve all the problems and loopholes of the existing system. This system has been developed based on computer vision and machine learning which is a supervised trained model. The proposed system using the CCTV footage using that as the input video applying computer vision on it and detects the people by the algorithm known as YOLO which stands for you only look once, and returns the coordinates of the centroids of the detected human subjects. With the help of those coordinates we measure Euclidean distance and check whether the people are maintaining social distance or not. Here we require very few people, the person who is maintaining the system properly and the admin who monitors the CCTV surveillance.

The admin who monitors the CCTV surveillance is shown the output processed video. The ones who maintain social distance that is 1.5 Meters are shown in green boxes and the ones who don’t in red boxes. This model is more accurate and can monitor more people who are not maintaining social distance at a time. The object detection algorithm we have used is fast and accurate which has been trained to classify human subjects.

ADVANTAGES

- Ensures accuracy and efficiency.
- Requires less manpower.
- Reduces the overhead of maintaining shifts of people during day and night.
- Depends on CCTV surveillance which is 24/7/

- Minimum time needed for the various processing.
- Greater efficiency.
- Over-all better and easier service.
- User friendliness.
- Also gives the count of people who are not maintaining social distance therefore no need to keep a track of them.

2.4 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.

Three key considerations involved in the feasibility analysis are

➢Economic Feasibility

➢Technical Feasibility

➢Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. The following are some of the important financial questions asked during preliminary investigation:

• The costs conduct a full system investigation.

• The cost of the hardware and software.

• The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. As the cameras are preinstalled at signals in many major cities, timers are available and surveillance centres are also present which means these centres can be used for running the algorithm to calculate signal time. Hence, the system can be developed because of its economic feasibility.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 SOCIAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is socially feasible.

2.5 SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a document that captures a complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

2.5.1 USER REQUIREMENTS

User requirements, often referred to as user needs, describe what the user does with the system, such as what activities that users must be able to perform. User requirements are generally documented in a User Requirements Document (URD) using narrative text. User requirements are generally signed off by the user and used as the primary input for creating system requirements.

2.5.2 SOFTWARE REQUIREMENT

- Python 3.0 or above installed
- Windows 10 or macOS X or Ubuntu
- Tensorflow and openCV

2.5.3 HARDWARE REQUIREMENT

- Core i3 4th gen or more
- 4GB RAM or more
- SSD storage preferred
- GPU preferred to train the model

- Camera Feed for input

2.6 TECHNOLOGIES USED IN DEVELOPMENT

- **Language:** python 3.0
- **IDE:** Visual Studio Code or jupyter notebook
- **Backend:** Artificial Intelligence and Machine Learning
- **Major libraries:** Tensor flow and openCV
- **Algorithm:** YOLO v3
- **Dataset:** COCO dataset based on human subjects
- **Deployment:** Git and github

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

Initially before understanding the project architecture it is important to understand the architecture of the YOLO algorithm. YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. We assign one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

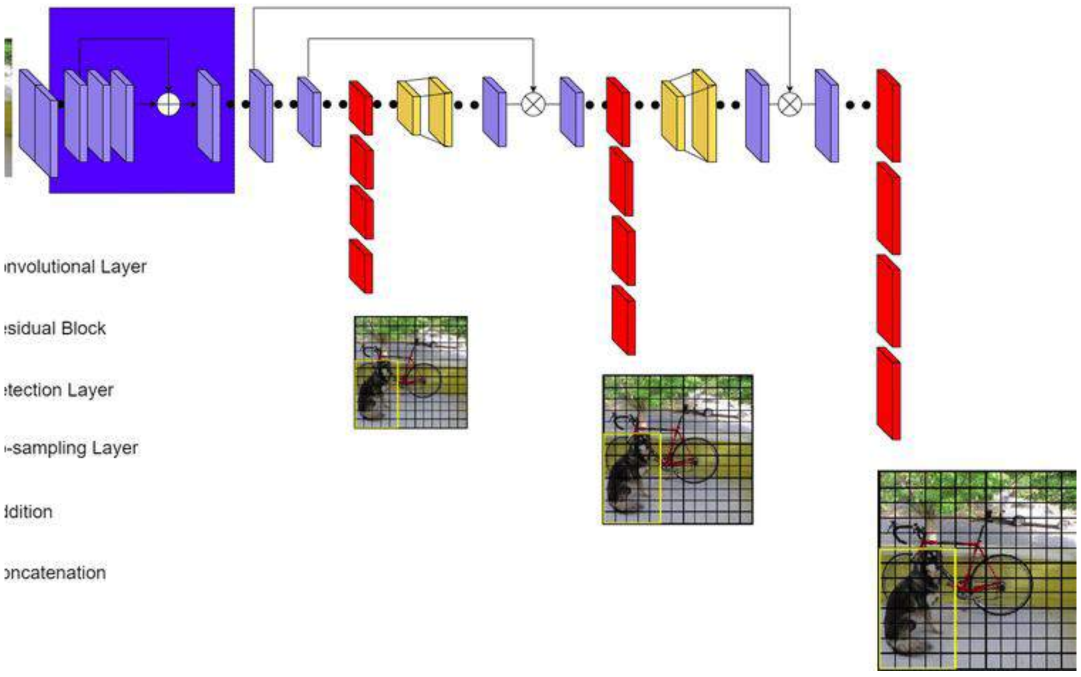


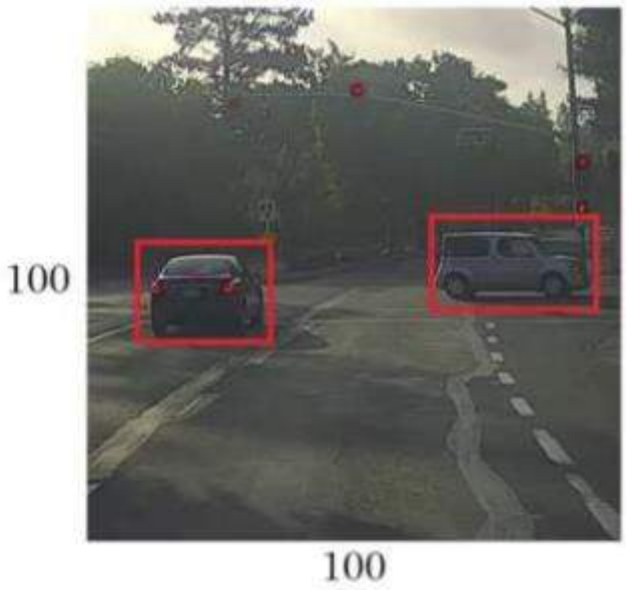
Fig. 3.1.1 Architecture of YOLO

The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It 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 its superb speed – it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation. This is one of the best algorithms for object detection and has shown a comparatively similar performance to the R-CNN algorithms.

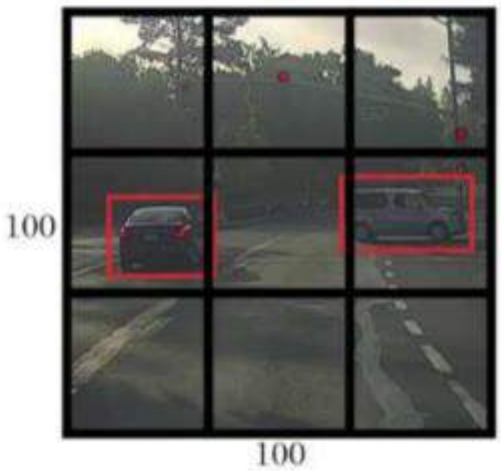
How Does Yolo Framework work?

Now that we have grasp on why YOLO is such a useful framework, let’s jump into how it actually works. In this section, I have mentioned the steps followed by YOLO for detecting objects in a given image.

- YOLO first takes an input image:



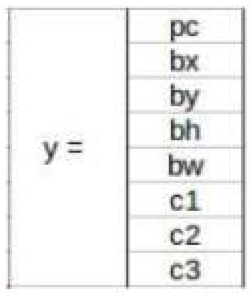
- The framework then divides the input image into grids (say a 3 X 3 grid):



- Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects (if any are found, of course).

Pretty straightforward, isn't it? Let's break down each step to get a more granular understanding of what we just learned.

We need to pass the labelled data to the model in order to train it. Suppose we have divided the image into a grid of size 3 X 3 and there are a total of 3 classes which we want the objects to be classified into. Let's say the classes are Pedestrian, Car, and Motorcycle respectively. So, for each grid cell, the label y will be an eight-dimensional vector:



	pc
	bx
	by
	bh
y =	bw
	c1
	c2
	c3

- pc defines whether an object is present in the grid or not (it is the probability)
- bx, by, bh, bw specify the bounding box if there is an object
- c1, c2, c3 represent the classes. So, if the object is a car, c2 will be 1 and c1 & c3 will be 0, and so on.

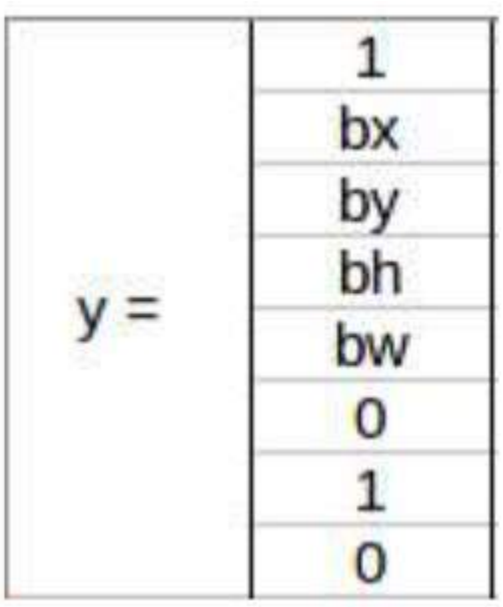
Let's say we select the first grid from the above example:

Since there is no object in this grid, pc will be zero and the y label for this grid will be:

$$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

Here, ‘?’ means that it doesn’t matter what bx, by, bh, bw, c1, c2, and c3 contain as there is no object in the grid. Let’s take another grid in which we have a car (c2 = 1):

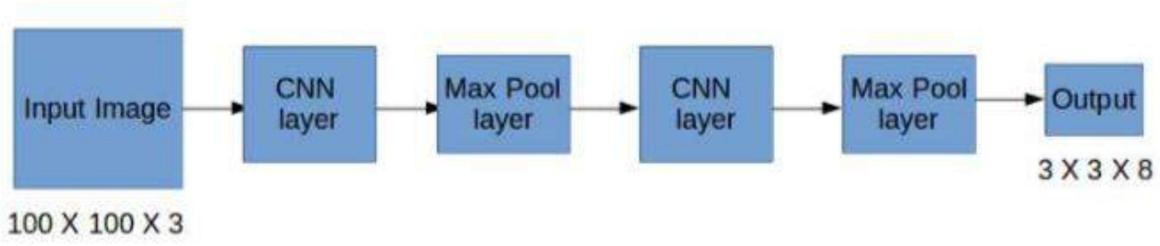
Before we write the y label for this grid, it’s important to first understand how YOLO decides whether there actually is an object in the grid. In the above image, there are two objects (two cars), so YOLO will take the mid-point of these two objects and these objects will be assigned to the grid which contains the mid-point of these objects. The y label for the centre left grid with the car will be:



y =	1
	bx
	by
	bh
	bw
	0
	1
	0

Since there is an object in this grid, pc will be equal to 1. bx, by, bh, bw will be calculated relative to the particular grid cell we are dealing with. Since car is the second class, c2 = 1 and c1 and c3 = 0. So, for each of the 9 grids, we will have an eight-dimensional output vector. This output will have a shape of 3 X 3 X 8.

So now we have an input image and its corresponding target vector. Using the above example (input image – 100 X 100 X 3, output – 3 X 3 X 8), our model will be trained as follows:



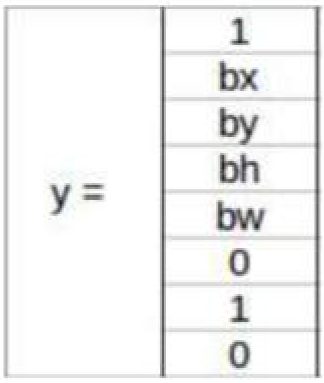
We will run both forward and backward propagation to train our model. During the testing phase, we pass an image to the model and run forward propagation until we get an output y. In order to keep things simple, I have explained this using a 3 X 3 grid here, but generally in real-world scenarios we take larger grids (perhaps 19 X 19).

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. We can reduce the chances of multiple objects appearing in the same grid cell by increasing the more number of grids (19 X 19, for example).

How to Encode Bounding boxes:

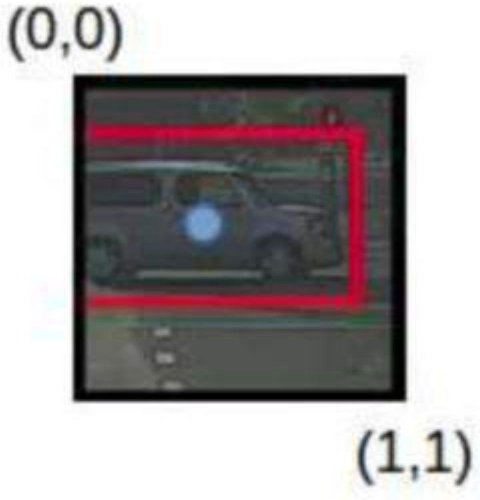
As I mentioned earlier, bx, by, bh, and bw are calculated relative to the grid cell we are dealing with. Let's understand this concept with an example. Consider the center-right grid which contains a car:

So, bx, by, bh, and bw will be calculated relative to this grid only. The y label for this grid will be:

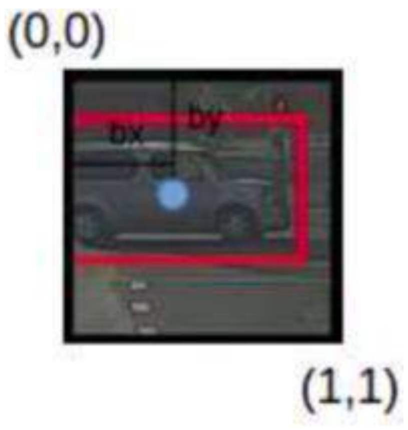


y =	1
	bx
	by
	bh
	bw
	0
	1
	0

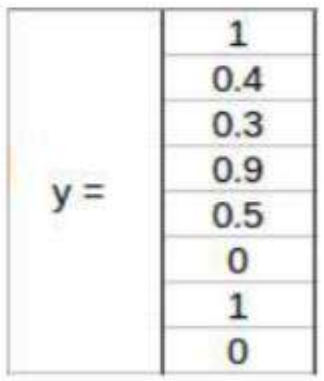
pc = 1 since there is an object in this grid and since it is a car, c2 = 1. Now, let's see how to decide bx, by, bh, and bw. In YOLO, the coordinates assigned to all the grids are:



bx, by are the x and y coordinates of the midpoint of the object with respect to this grid. In this case, it will be (around) bx = 0.4 and by = 0.3:



bh is the ratio of the height of the bounding box (red box in the above example) to the height of the corresponding grid cell, which in our case is around 0.9. So, bh = 0.9. bw is the ratio of the width of the bounding box to the width of the grid cell. So, bw= 0.5 (approximately). The y label for this grid will be:



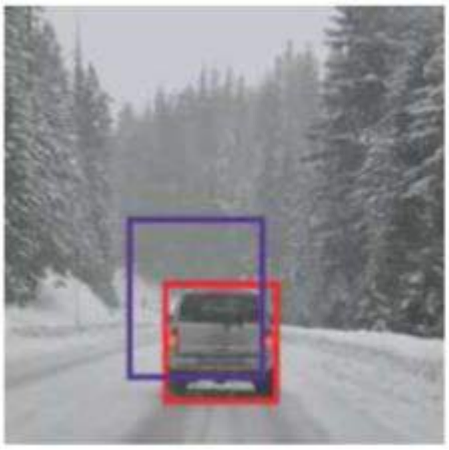
y =	1
	0.4
	0.3
	0.9
	0.5
	0
	1
	0

Notice here that bx and by will always range between 0 and 1 as the midpoint will always lie within the grid. Whereas bh and bw can be more than 1 in case the dimensions of the bounding box are more than the dimension of the grid.

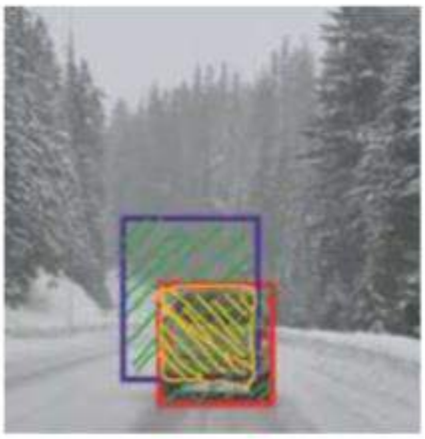
In the next section, we will look at more ideas that can potentially help us in making this algorithm's performance even better.

INTERSECTION OVER UNION AND NON-MAX SUPPRESSION

Here's some food for thought – how can we decide whether the predicted bounding box is giving us a good outcome (or a bad one)? This is where Intersection over Union comes into the picture. It calculates the intersection over union of the actual bounding box and the predicted bonding box. Consider the actual and predicted bounding boxes for a car as shown below:



Here, the red box is the actual bounding box and the blue box is the predicted one. How can we decide whether it is a good prediction or not? IoU, or Intersection over Union, will calculate the area of the intersection over union of these two boxes. That area will be:

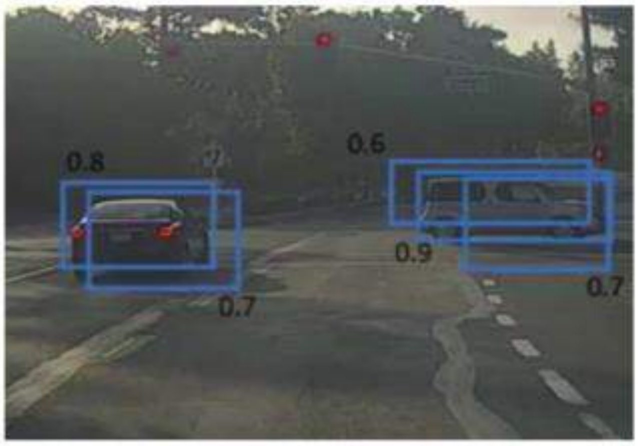


IoU = Area of the intersection / Area of the union, i.e. IoU = Area of yellow box / Area of green box

If IoU is greater than 0.5, we can say that the prediction is good enough. 0.5 is an arbitrary threshold we have taken here, but it can be changed according to your specific problem. Intuitively, the more you increase the threshold, the better the predictions become.

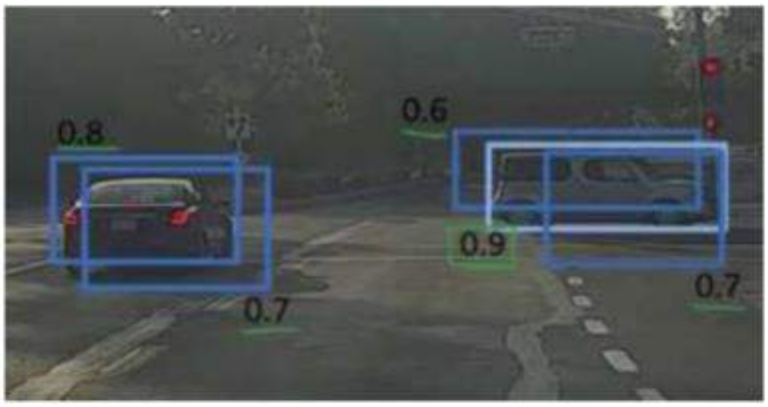
There is one more technique that can improve the output of YOLO significantly – Non-Max Suppression.

One of the most common problems with object detection algorithms is that rather than detecting an object just once, they might detect it multiple times. Consider the below image:

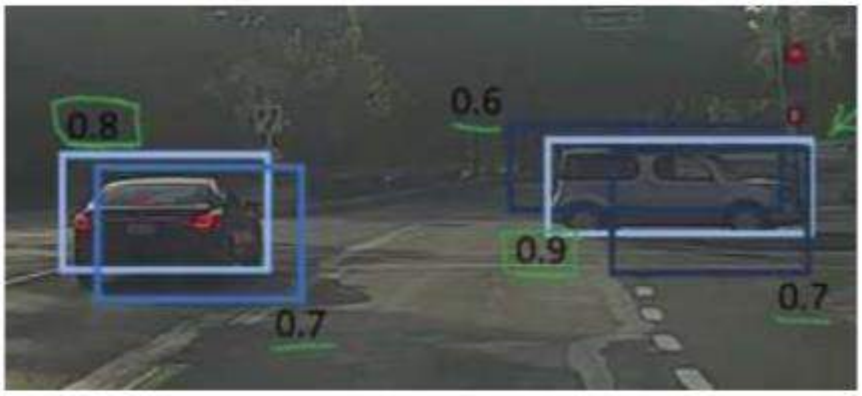


Here, the cars are identified more than once. The Non-Max Suppression technique cleans up this up so that we get only a single detection per object. Let’s see how this approach works.

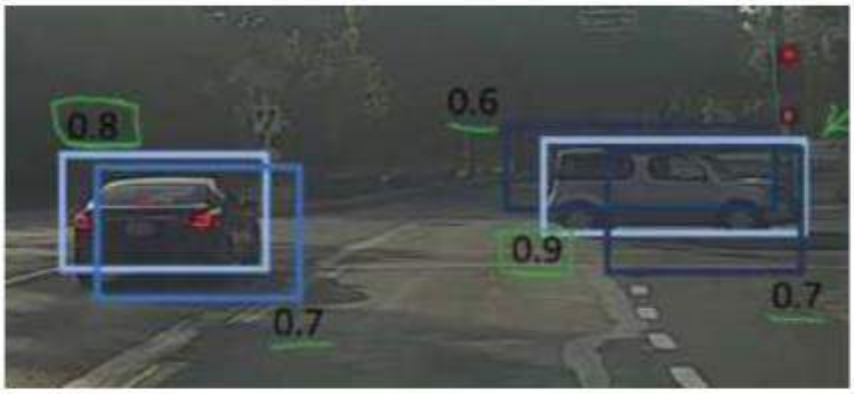
1. It first looks at the probabilities associated with each detection and takes the largest one. In the above image, 0.9 is the highest probability, so the box with 0.9 probability will be selected first:



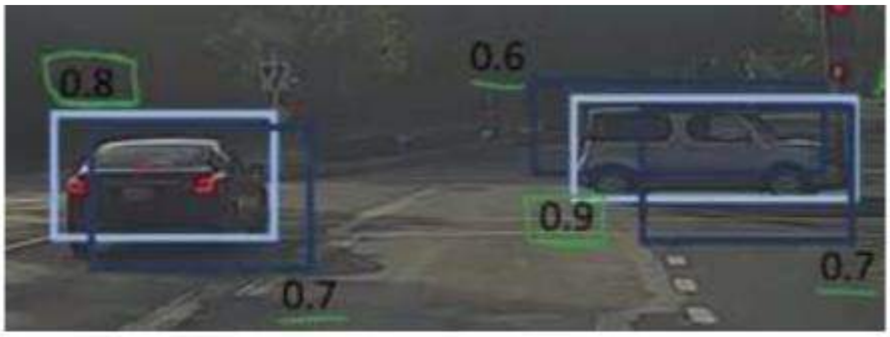
2. Now, it looks at all the other boxes in the image. The boxes which have high IoU with the current box are suppressed. So, the boxes with 0.6 and 0.7 probabilities will be suppressed in our example:



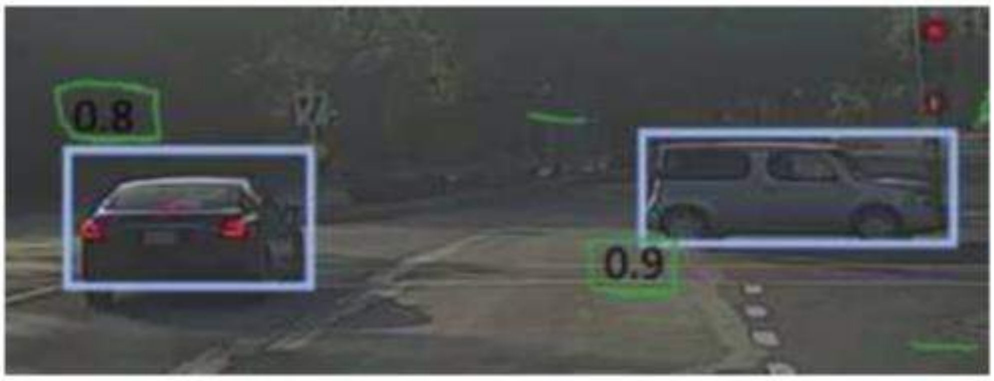
3. After the boxes have been suppressed, it selects the next box from all the boxes with the highest probability, which is 0.8 in our case:



4. Again, it will look at the IoU of this box with the remaining boxes and compress the boxes with a high IoU:



5. We repeat these steps until all the boxes have either been selected or compressed and we get the final bounding boxes:



This is what Non-Max Suppression is all about. We are taking the boxes with maximum probability and suppressing the close-by boxes with non-max probabilities. Let's quickly summarize the points which we've seen in this section about the Non-Max suppression algorithm:

1. Discard all the boxes having probabilities less than or equal to a pre-defined threshold (say, 0.5)

2. For the remaining boxes:

   1. Pick the box with the highest probability and take that as the output prediction

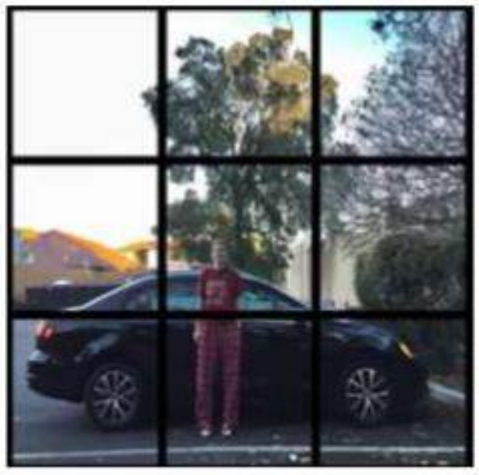
   2. Discard any other box which has IoU greater than the threshold with the output box from the above step

3. Repeat step 2 until all the boxes are either taken as the output prediction or discarded

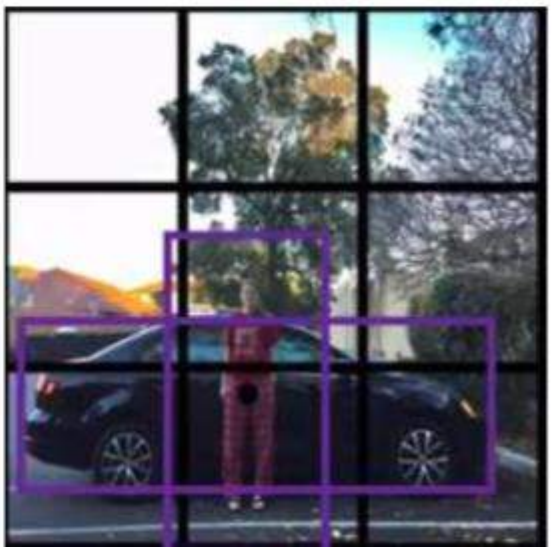
There is another method we can use to improve the perform of a YOLO algorithm – let’s check it out.

ANCHOR BOXES

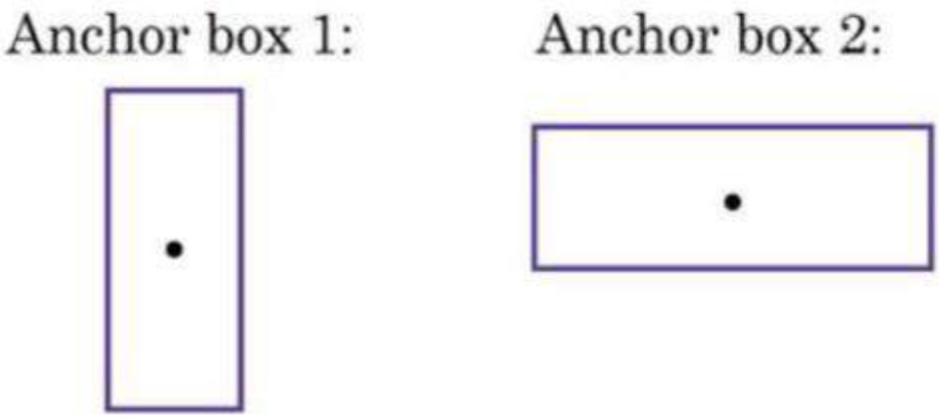
We have seen that each grid can only identify one object. But what if there are multiple objects in a single grid? That can so often be the case in reality. And that leads us to the concept of anchor boxes. Consider the following image, divided into a 3 X 3 grid:



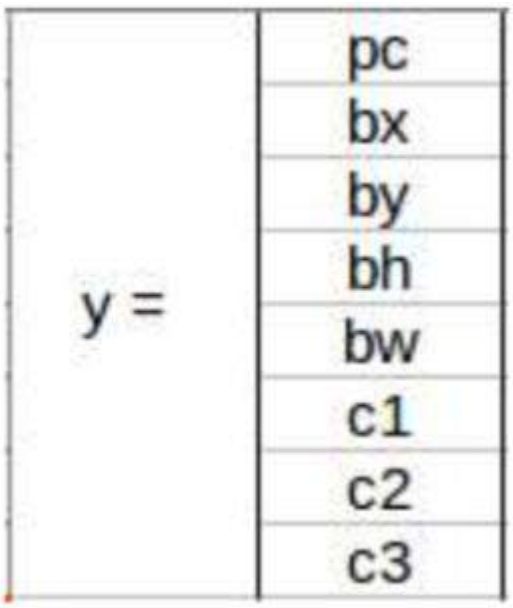
Remember how we assigned an object to a grid? We took the midpoint of the object and based on its location, assigned the object to the corresponding grid. In the above example, the midpoint of both the objects lies in the same grid. This is how the actual bounding boxes for the objects will be:



We will only be getting one of the two boxes, either for the car or for the person. But if we use anchor boxes, we might be able to output both boxes! How do we go about doing this? First, we pre-define two different shapes called anchor boxes or anchor box shapes. Now, for each grid, instead of having one output, we will have two outputs. We can always increase the number of anchor boxes as well. I have taken two here to make the concept easy to understand:

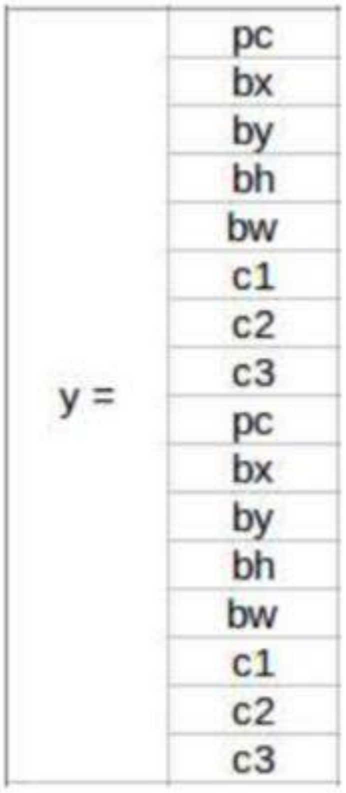


This is how the y label for YOLO without anchor boxes looks like:



y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

What do you think the y label will be if we have 2 anchor boxes? I want you to take a moment to ponder this before reading further. Got it? The y label will be:



y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3
	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

the first 8 rows belong to anchor box 1 and the remaining 8 belongs to anchor box 2. The objects are assigned to the anchor boxes based on the similarity of the bounding boxes and the anchor box shape. Since the shape of anchor box 1 is similar to the bounding box for the person, the latter will be assigned to anchor box 1 and the car will be assigned to anchor box 2. The output in this case, instead of 3 X 3 X 8 (using a 3 X 3 grid and 3 classes), will be 3 X

3 X 16 (since we are using 2 anchors). So, for each grid, we can detect two or more objects based on the number of anchors. Let’s combine all the ideas we have covered so far and integrate them into the YOLO framework.

3.1.1 COCO Dataset

COCO (Common Object in Context) is a large image dataset designed for object detection, segmentation, person key points detection, stuff segmentation, and caption generation. This package provides Matlab, Python, and Lua APIs that assists in loading, parsing, and visualizing the annotations in COCO. For our system we used the COCO dataset to classify for the human subjects.

3.1.2 Project Architecture (System):

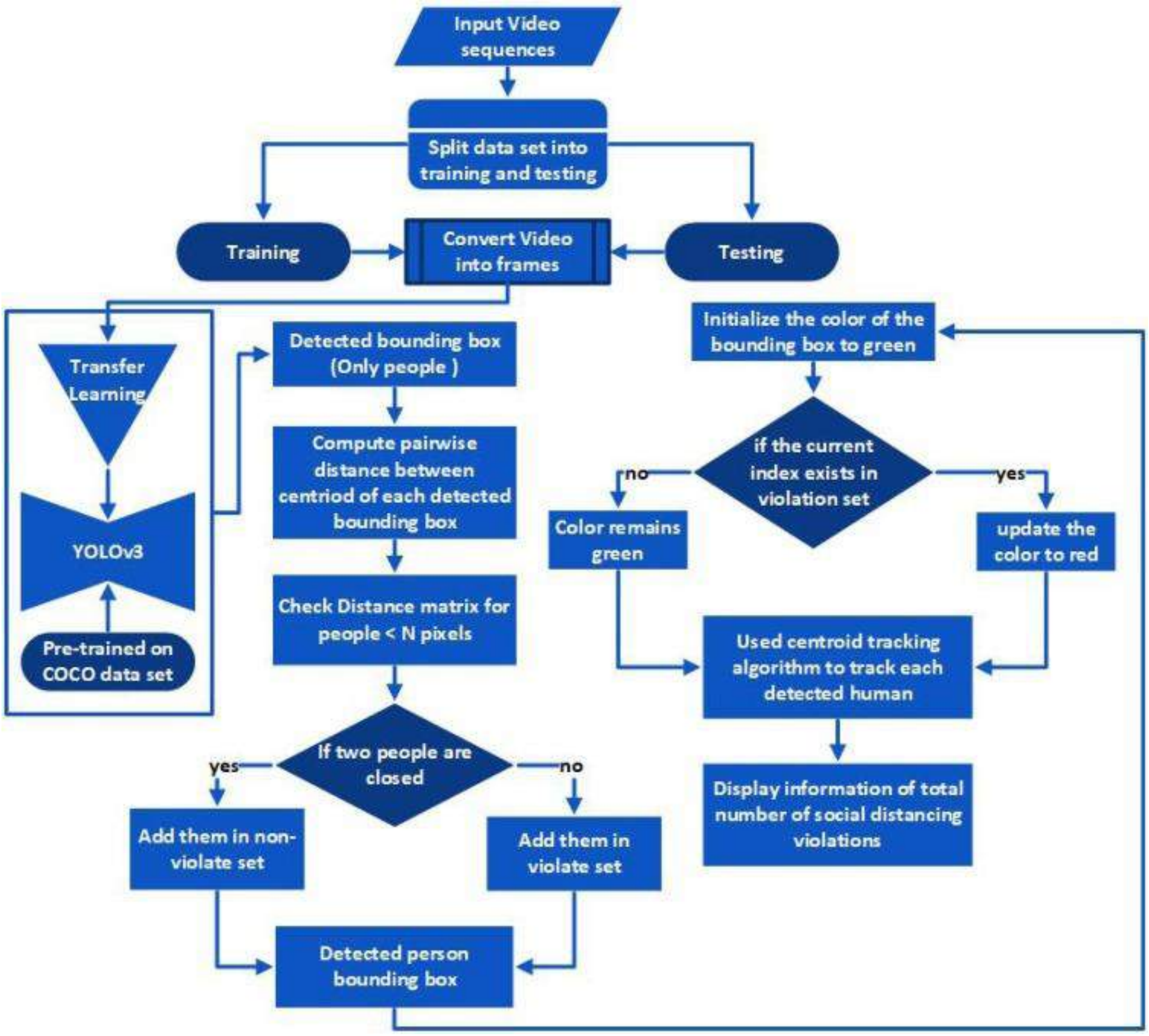

Input Video sequences
Split data set into training and testing
Training
Convert Video into frames
Testing
Transfer Learning
YOLOv3
Pre-trained on COCO data set
Detected bounding box (Only people)
Compute pairwise distance between centriod of each detected bounding box
Check Distance matrix for people < N pixels
If two people are closed
yes
no
Add them in non-violate set
Add them in violate set
Detected person bounding box
Initialize the color of the bounding box to green
if the current index exists in violation set
no
yes
Color remains green
update the color to red
Used centroid tracking algorithm to track each detected human
Display information of total number of social distancing violations

Fig 3.1.2.1 Project Architecture

The recorded overhead data sets (input video converted to frames) are split into training and testing sets. A deep learning-based detection paradigm is used to detect individuals in sequences.

There are a variety of object detection models available, due to the best performance we are using YOLO V3 algorithm. The model used single-stage network architecture to estimate the bounding boxes and class probabilities. The model was originally trained on the COCO (Common objects in context) data set. For overhead view person detection, transfer learning is implemented to enhance the detection model's efficiency, and a new layer of overhead training is added with the existing architecture.

After detection, the bounding box information, mainly centroid information, is used to compute each bounding box centroid distance. We used Euclidean distance and calculated the distance between each detected bounding box of peoples. Following computing centroid distance, a predefined threshold is used to check either the distance among any two bounding box centroids is less than the configured number of pixels or not. If two people are close to each other and the distance value violates the minimum social distance threshold. The bounding box information is stored in a violation set, and the color of the bounding box is updated/changed to red. A centroid tracking algorithm is adopted for tracking so that it helps in tracking of those people who violate/breach the social distancing threshold. At the output, the model displays the information about the total number of social distancing violations along with detected people bounding boxes and centroids.

In this work, YOLOv3 is used for human detection as it improves predictive accuracy, particularly for small-scale objects. The main advantage is that it has adjusted network structure for multi-scale object detection. Furthermore, for object classification, it uses various independent logistic rather than softmax. The model's overall architecture is presented in Fig. 3.1.2.2 ; it can be seen that feature learning is performed using the convolutional layers, also called Residual Blocks. The blocks are made up of many convolutional layers and skip connections. The model's unique characteristic is that it performs detection at three separate scales, as depicted in Fig. 3.1.2.2 .The convolutional layers with a given stride are practiced to downsample the feature map and transfer invariant-sized features (Redmon & Farhadi, 2018). Three feature maps, as shown in Fig. 3.1.2.2, are utilized for object detection.

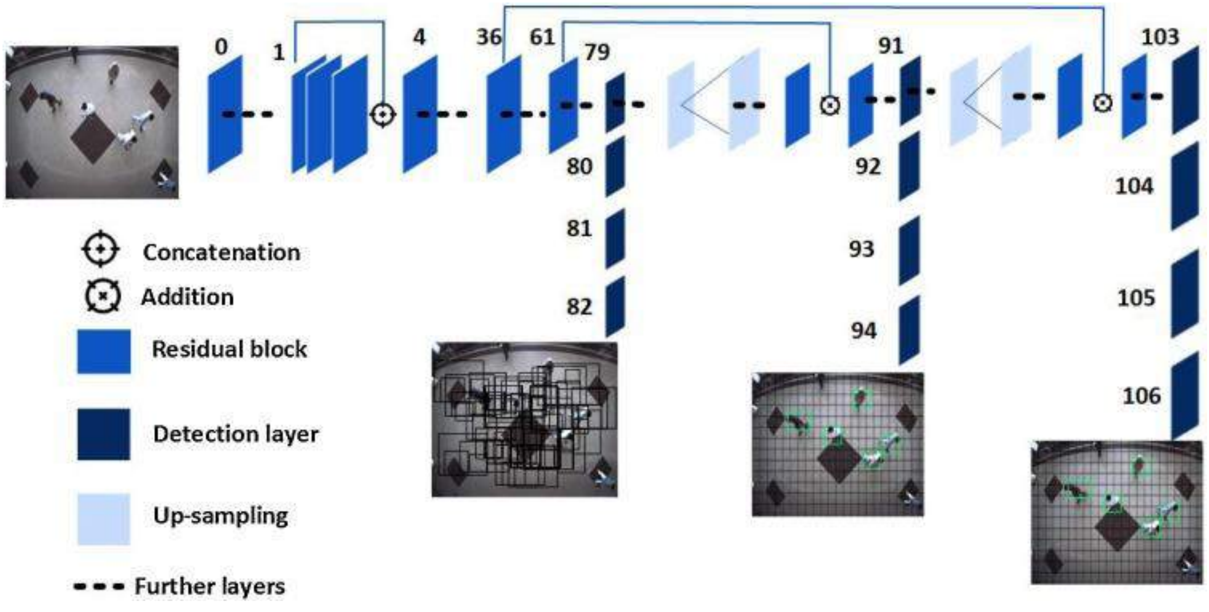


Fig. 3.1.2.2 Yolo Implemented for Human Detection

The architecture shown in Fig. 3.1.2.2 is trained using an overhead data set. For that purpose, a transfer learning approach is adopted, that enhance the efficiency of the model. With transfer learning, the model is additionally trained without dropping the valuable information of the existing model. Further, the additional overhead data set trained layer is appended with the existing architecture. In this way, the model takes advantage of the pre-trained and newly trained information, and both detection results are further deliver better and faster detection results.

The architecture shown in Fig. 3.1.2.2 used a single-stage network for the entire input image to predict the bounding box and class probability of detected objects. For feature extraction, the architecture utilizes convolution layers, and for class prediction, fully connected layers are used. During human identification, as seen in Fig. 3.1.2.2, the input frame is divided into a region of S×S, also called grid cells. These cells are related to bounding box estimation and class probabilities. It predicts the probability of whether the center of the person bounding box is in the grid cell or not:

$$\mathbf{Conf(p)=Pr(p)\times IOU(pred,actual)}$$

Pr(p) indicates that whether the person present is in the detected bounding box or not. The value of Pr(p) is 1 for yes and 0 for not. IoU(pred,actual) determines the Intersection Over Union of the actual and predicted bounding box. It is defined as (Redmon & Farhadi, 2018):

$$\mathbf{IoU(pred,actual)=areaBoxT\cap BoxP/BoxT\cup BoxP}$$

where the ground truth box (actual) manually labeled in the training data set represented with BoxT, and the predicted bounding box is displayed as BoxP. area presents the area of intersection. An acceptable area is predicted and decided for each detected person in the input frame. The confidence value is applied after prediction to achieve the optimal bounding box. For each predicted bounding box, h,w,x,y are estimated, where bounding box coordinates are defined by x,y, and width and height are determined by w,h. The model produces the following predicted bounding box values as seen in Fig. 3.1.2.3 and Eq. (3) (Redmon & Farhadi, 2018);

$$\mathbf{bx=\sigma(tx)+cx}$$

$$\mathbf{by=\sigma(ty)+cy}$$

$$\mathbf{bw=pwetw}$$

$$\mathbf{bh=pwhth}$$

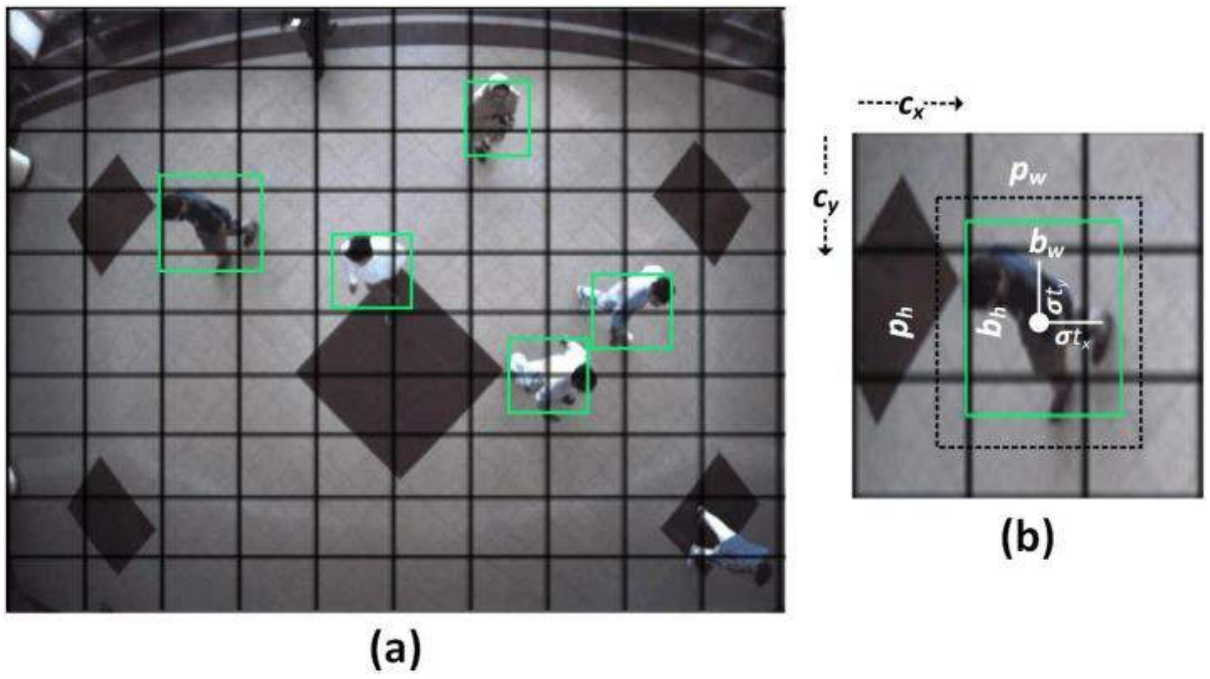


Fig. 3.1.2.3 Detection Coordinates of person bounding boxes

In Eq. (3), bx,by,bw,bh are predicted coordinate bounding boxes, where the coordinates' center is represented as x,y and width and height with w,h. tw,th,tx,ty, defined the network output and cx,cy are used to correspond the top-left coordinates of the grid cell as shown in Fig. 3.1.2.3, while the pw and ph are width and height of anchors.

A threshold value is defined that process the high confidence values and discards the low confidence values. Using non-maximal suppression, the final location parameters are derived for the detected bounding box. At last, loss function is calculated, for detected bounding box (Redmon & Farhadi, 2018). The given loss function is the sum of three functions, i.e., regression, classification, and confidence.

After detecting people in video frames, in the next step, the centroid of each detected person bounding boxes shown as green boxes are used for distance calculation, as shown in Fig. 3.1.2.4 (b). The detected bounding box coordinates (x,y) are used to compute the bounding box's centroid. Fig. 3.1.2.4(c) demonstrates accepting a set of bounding box coordinates and computing the centroid. After computing, centroid, a unique ID is assigned to each detected bounding box. In the next step, we measure the distance between each detected centroid using Euclidean distance. For every subsequent frame in the video stream, we firstly compute bounding box centroids shown in Fig. 3.1.2.4 (c); and then calculate the distance (highlighted with red lines) between each pair of detected bounding box centroids, Fig. 3.1.2.4 (d). The information of each centroid is stored in the form of a list. Based on distance values, a threshold is defined to check if any two people are less than N pixels apart or not. If the distance violates the minimum social distance set or two people are too close, then the information is added into the violation set. The bounding box color is initialized as green. The information is checked in the violation set; if the current index exists in the violation set, the color is updated to red. Furthermore, the centroid tracking algorithm is used to track the detected people in the video sequence. The tracking algorithm also helps to keep track of people who are violating the social distance threshold. At the output, the model displays information about the total number of social distancing violations.

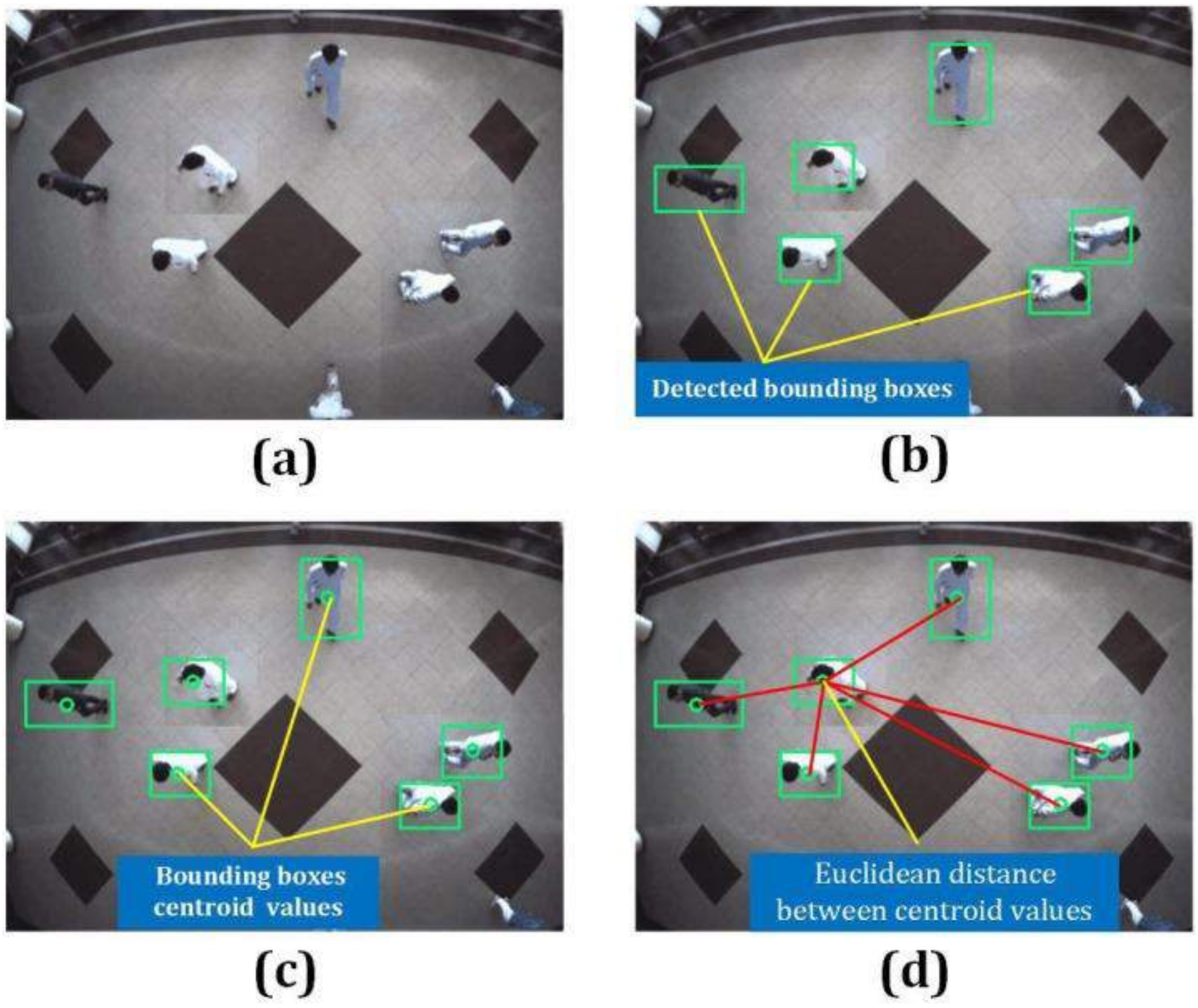

Detected bounding boxes
(a)
(b)
Bounding boxes centroid values
Euclidean distance between centroid values
(c)
(d)

Fig. 3.1.2.4 Social Distance Prediction

3.2 USE CASE DIAGRAM

Use case diagram describes who are the actors and what all work each of the actors do using the system. In our system there are three actors since most of the work is done by the system itself. One is the System which evaluates the social distance in real time, the other one is the CCTV which gives the input footage to the system and the last one is admin who checks whether they are following the social distance or not.

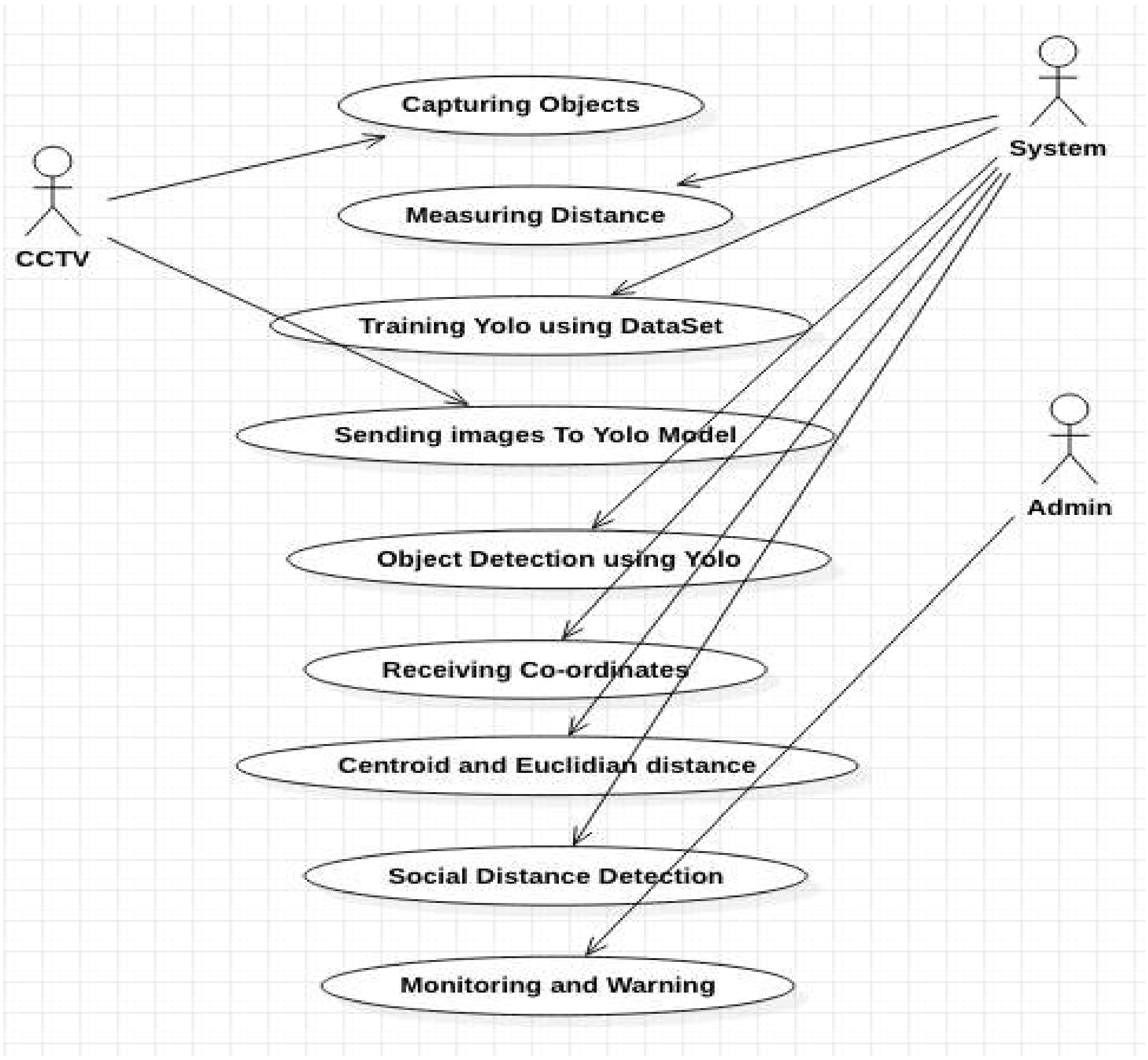

System
CCTV
Admin
Capturing Objects
Measuring Distance
Training Yolo using DataSet
Sending images To Yolo Model
Object Detection using Yolo
Receiving Co-ordinates
Centroid and Euclidian distance
Social Distance Detection
Monitoring and Warning

Fig. 3.2 Use Case Diagram

3.3 CLASS DIAGRAM

Class diagram shows the functions and methods performed by each of the actors, in the case of our system it shows what all functions and methods have been executed by each of the actors.

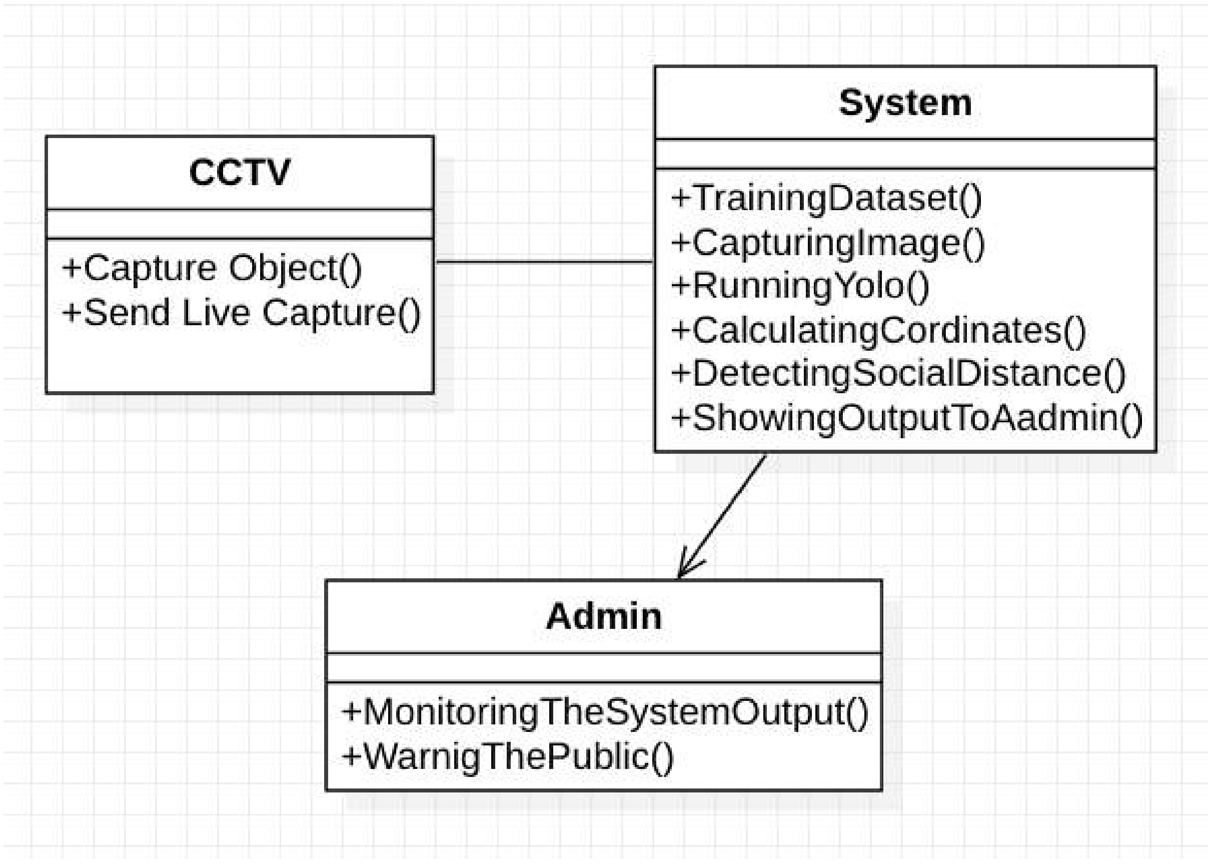


Fig. 3.3 Class Diagram

3.4 Sequence Diagram

As we have seen in views/modules, we will see how the entire system works step by step and action by action from one actor to another in the following sequence diagram. Sequence diagram will show all the activities in the sequence of which the system is executed and how one activity passes on to another.

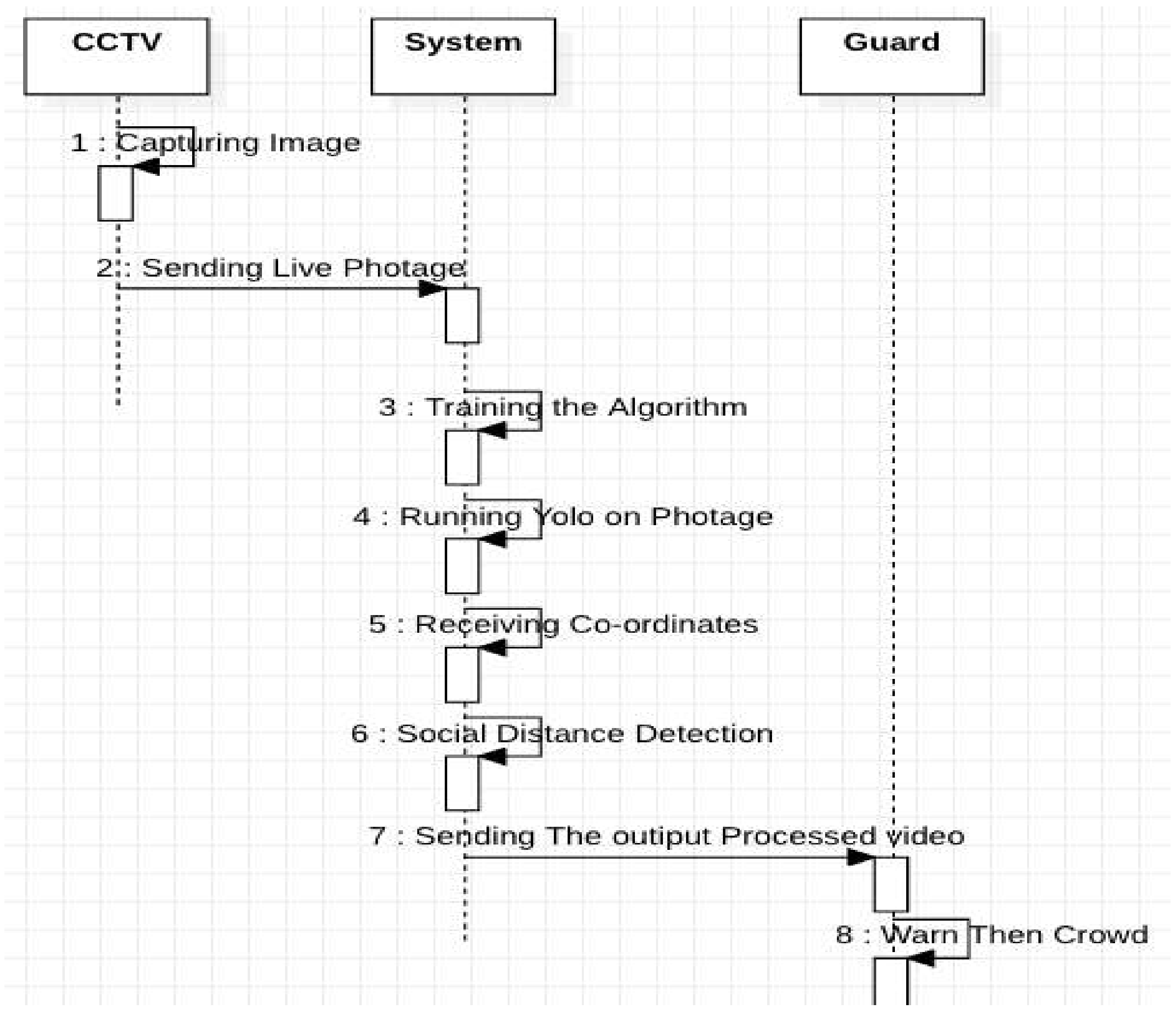

CCTV
System
Guard
1 : Capturing Image
2 : Sending Live Photage
3 : Training the Algorithm
4 : Running Yolo on Photage
5 : Receiving Co-ordinates
6 : Social Distance Detection
7 : Sending The outiput Processed video
8 : Warn Then Crowd

Fig. 3.4 Sequence diagram

3.5 Activity Diagram

It describes the flow of the activity states.

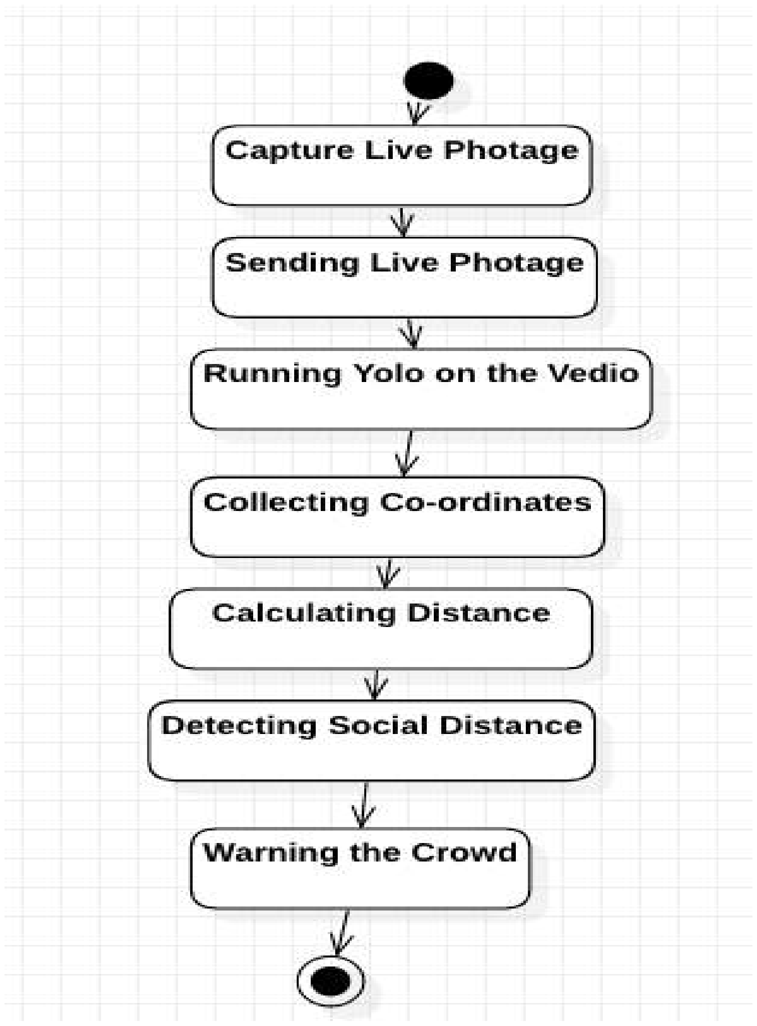


Fig. 3.5 Activity Diagram

4. IMPLEMENTATION

4.1 Social Distance Detector.py

```
# USAGE
# python social_distance_detector.py --input pedestrians.mp4
# python social_distance_detector.py --input pedestrians.mp4 --output output.avi

# import the necessary packages
from TheLazyCoder import social_distancing_config as config
from TheLazyCoder.detection import detect_people
from scipy.spatial import distance as dist
import numpy as np
import argparse
import imutils
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="",
	help="path to (optional) input video file")
ap.add_argument("-o", "--output", type=str, default="",
	help="path to (optional) output video file")
ap.add_argument("-d", "--display", type=int, default=1,
	help="whether or not output frame should be displayed")
args = vars(ap.parse_args())

# load the COCO class labels our YOLO model was trained on
labelsPath = os.path.sep.join([config.MODEL_PATH, "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")

# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join([config.MODEL_PATH, "yolov3.weights"])
configPath = os.path.sep.join([config.MODEL_PATH, "yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# check if we are going to use GPU
if config.USE_GPU:
        # set CUDA as the preferable backend and target
        print("[INFO] setting preferable backend and target to CUDA...")
        net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
        net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

# determine only the *output* layer names that we need from YOLO
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# initialize the video stream and pointer to output video file
print("[INFO] accessing video stream...")
vs = cv2.VideoCapture(args["input"] if args["input"] else 0)
writer = None

# loop over the frames from the video stream
while True:
        # read the next frame from the file
        (grabbed, frame) = vs.read()

        # if the frame was not grabbed, then we have reached the end
        # of the stream
        if not grabbed:
                break

        # resize the frame and then detect people (and only people) in it
        frame = imutils.resize(frame, width=700)
        results = detect_people(frame, net, ln,
                personIdx=LABELS.index("person"))

        # initialize the set of indexes that violate the minimum social
        # distance
        violate = set()

        # ensure there are *at least* two people detections (required in
        # order to compute our pairwise distance maps)
        if len(results) >= 2:
                # extract all centroids from the results and compute the
                # Euclidean distances between all pairs of the centroids
                centroids = np.array([r[2] for r in results])
                D = dist.cdist(centroids, centroids, metric="euclidean")

                # loop over the upper triangular of the distance matrix
                for i in range(0, D.shape[0]):
```

```
                for j in range(i + 1, D.shape[1]):
                        # check to see if the distance between any two
                        # centroid pairs is less than the configured number
                        # of pixels
                        if D[i, j] < config.MIN_DISTANCE:
                                # update our violation set with the indexes of
                                # the centroid pairs
                                violate.add(i)
                                violate.add(j)

# loop over the results
for (i, (prob, bbox, centroid)) in enumerate(results):
        # extract the bounding box and centroid coordinates, then
        # initialize the color of the annotation
        (startX, startY, endX, endY) = bbox
        (cX, cY) = centroid
        color = (0, 255, 0)

        # if the index pair exists within the violation set, then
        # update the color
        if i in violate:
                color = (0, 0, 255)

        # draw (1) a bounding box around the person and (2) the
        # centroid coordinates of the person,
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
        cv2.circle(frame, (cX, cY), 5, color, 1)

# draw the total number of social distancing violations on the
# output frame
text = "Social Distancing Violations: {}".format(len(violate))
cv2.putText(frame, text, (10, frame.shape[0] - 25),
        cv2.FONT_HERSHEY_SIMPLEX, 0.85, (0, 0, 255), 3)

# check to see if the output frame should be displayed to our
# screen
if args["display"] > 0:
        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
```

```
				break

		# if an output video file path has been supplied and the video
		# writer has not been initialized, do so now
		if args["output"] != "" and writer is None:
			# initialize our video writer
			fourcc = cv2.VideoWriter_fourcc(*"MJPG")
			writer = cv2.VideoWriter(args["output"], fourcc, 25,
				(frame.shape[1], frame.shape[0]), True)

		# if the video writer is not None, write the frame to the output
		# video file
		if writer is not None:
			writer.write(frame)
```

4.2 Requirement.txt:

Note: This is the file which consists of the packages and frameworks that we have used in the current project, before executing the project we first run a command to install these requirements on the command line or terminal.

```
imutils
numpy
opencv-python
Scipy
```

4.3 Detection.py:

```
# import the necessary packages
from .social_distancing_config import NMS_THRESH
from .social_distancing_config import MIN_CONF
import numpy as np
import cv2

def detect_people(frame, net, ln, personIdx=0):
	# grab the dimensions of the frame and  initialize the list of
	# results
	(H, W) = frame.shape[:2]
	results = []

	# construct a blob from the input frame and then perform a forward
	# pass of the YOLO object detector, giving us our bounding boxes
# and associated probabilities
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
net.setInput(blob)
layerOutputs = net.forward(ln)

# initialize our lists of detected bounding boxes, centroids, and
# confidences, respectively
boxes = []
centroids = []
confidences = []

# loop over each of the layer outputs
for output in layerOutputs:
        # loop over each of the detections
        for detection in output:
                # extract the class ID and confidence (i.e., probability)
                # of the current object detection
                scores = detection[5:]
                classID = np.argmax(scores)
                confidence = scores[classID]

                # filter detections by (1) ensuring that the object
                # detected was a person and (2) that the minimum
                # confidence is met
                if classID == personIdx and confidence > MIN_CONF:
                        # scale the bounding box coordinates back relative to
                        # the size of the image, keeping in mind that YOLO
                        # actually returns the center (x, y)-coordinates of
                        # the bounding box followed by the boxes' width and
                        # height
                        box = detection[0:4] * np.array([W, H, W, H])
                        (centerX, centerY, width, height) = box.astype("int")

                        # use the center (x, y)-coordinates to derive the top
                        # and and left corner of the bounding box
                        x = int(centerX - (width / 2))
                        y = int(centerY - (height / 2))

                        # update our list of bounding box coordinates,
                        # centroids, and confidences
                        boxes.append([x, y, int(width), int(height)])
                        centroids.append((centerX, centerY))
                        confidences.append(float(confidence))

    # apply non-maxima suppression to suppress weak, overlapping
    # bounding boxes
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF, NMS_THRESH)

    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indexes we are keeping
        for i in idxs.flatten():
            # extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

            # update our results list to consist of the person
            # prediction probability, bounding box coordinates,
            # and the centroid
            r = (confidences[i], (x, y, x + w, y + h), centroids[i])
            results.append(r)

    # return the list of results
    return results
```

4.4 social_distancing_config.py:

```
# base path to YOLO directory
MODEL_PATH = "yolo-coco"

# initialize minimum probability to filter weak detections along with
# the threshold when applying non-maxima suppression
MIN_CONF = 0.3
NMS_THRESH = 0.3

# boolean indicating if NVIDIA CUDA GPU should be used
USE_GPU = False

# define the minimum safe distance (in pixels) that two people can be
# from each other
MIN_DISTANCE = 50
```

5. SCREEN SHOTS

5.1 INTRODUCTION:

The below are the images that shows the input sample video and the output sample video processed by the system, here the input sample video is divided into frames and those frames are been given as the input and the system is been acted upon those frames and detect the social distance between people in each frame and classifies people in violation set as red and those who are not in violation set in green.

5.2 Input Sample Frames:

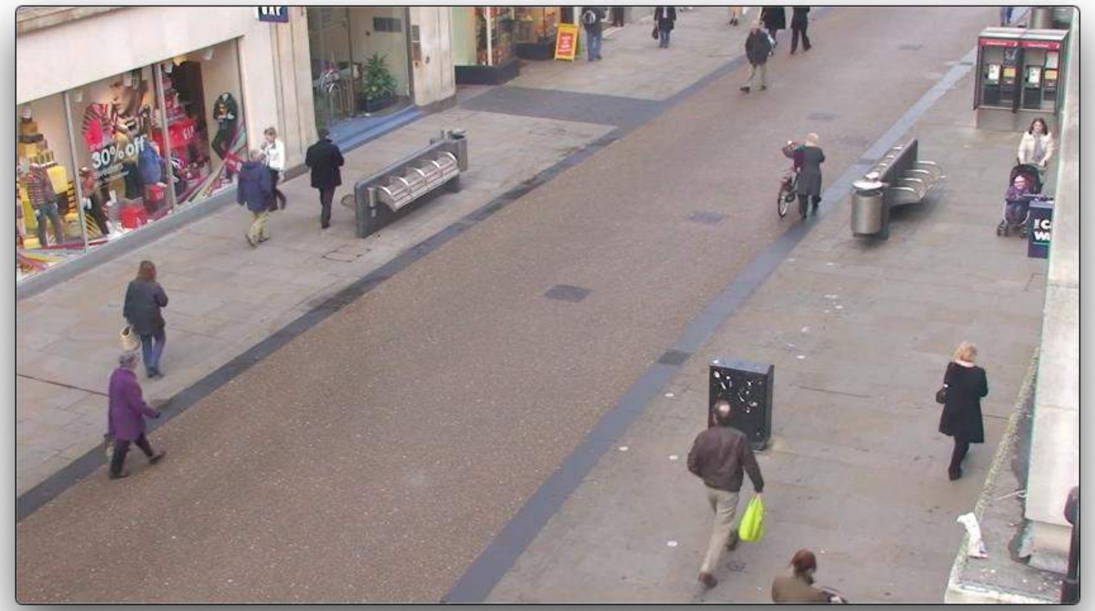


Fig. 5.2.1 input sample frame

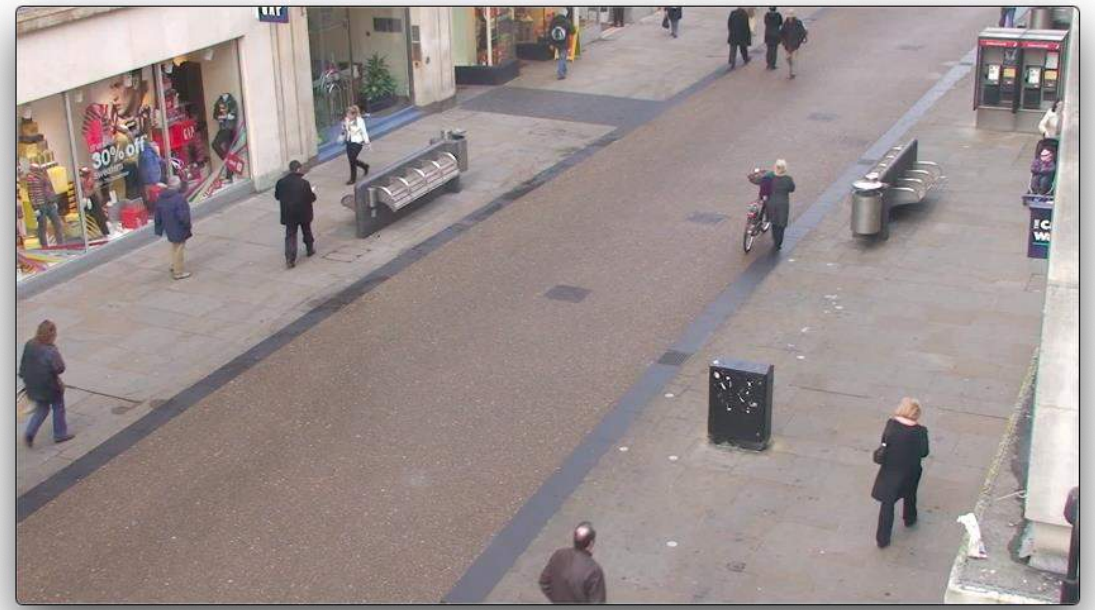


Fig. 5.2.2 input sample frame

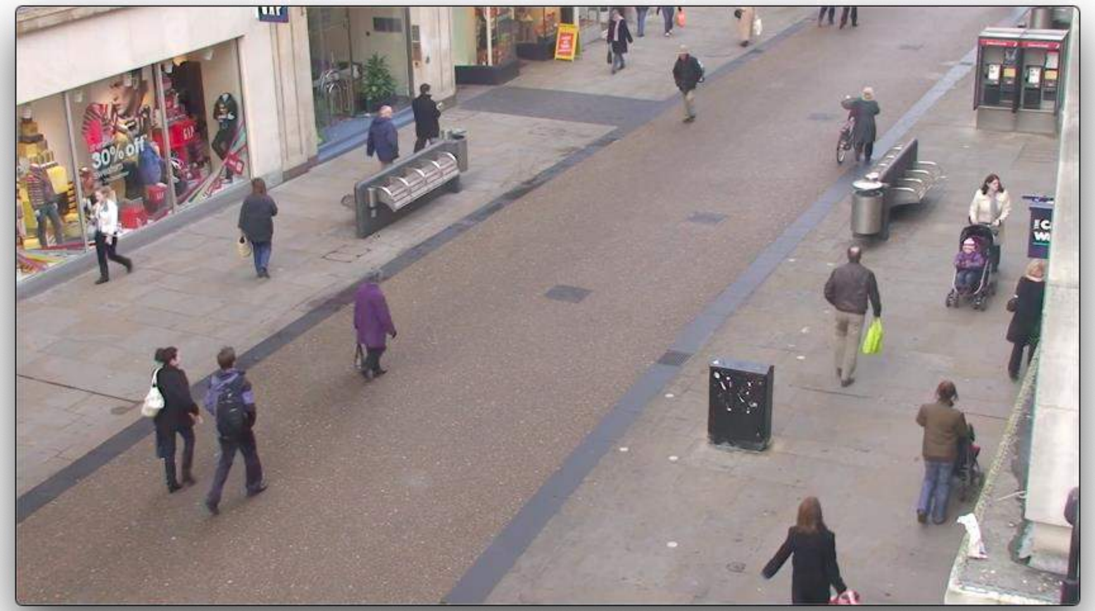


Fig. 5.2.3 input sample frame

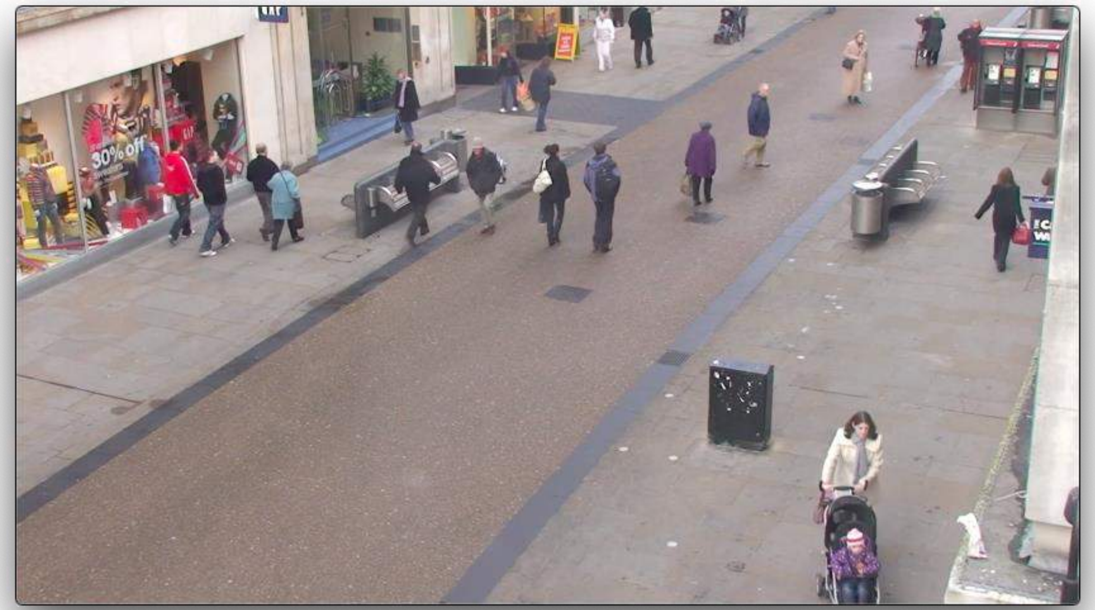


Fig. 5.2.4 input sample frame

5.3 Output Processed Sample Frames:

In these output processed sample frames the system processes the and runs yolo on each of the input frame and classifies the human subjects, now here it measures the Euclidean distance between each of the classified subject’s centroid and verifies whether they are under social distance or not, the subjects which tend to follow social distance are in green boxes and the ones who don’t are in the red boxes.

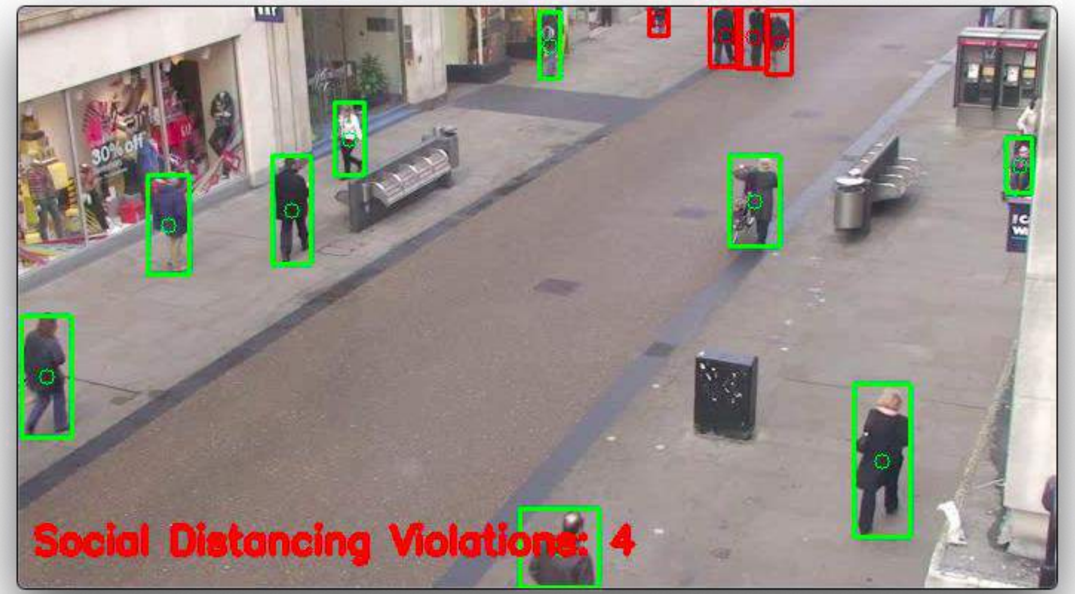


Fig. 5.3.1 output processed sample frame

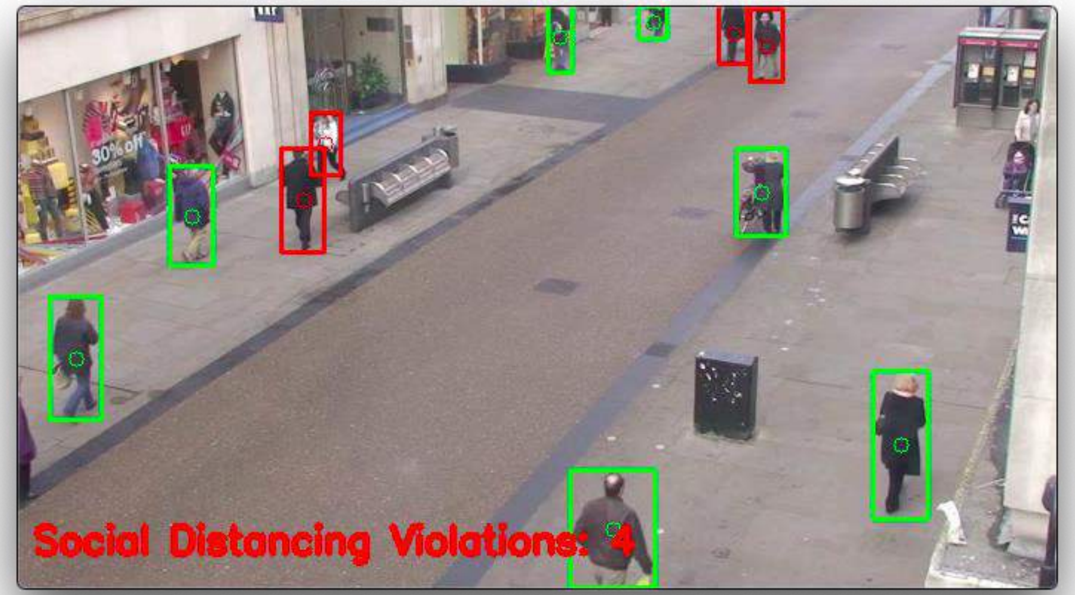


Fig. 5.3.2 output processed sample frame

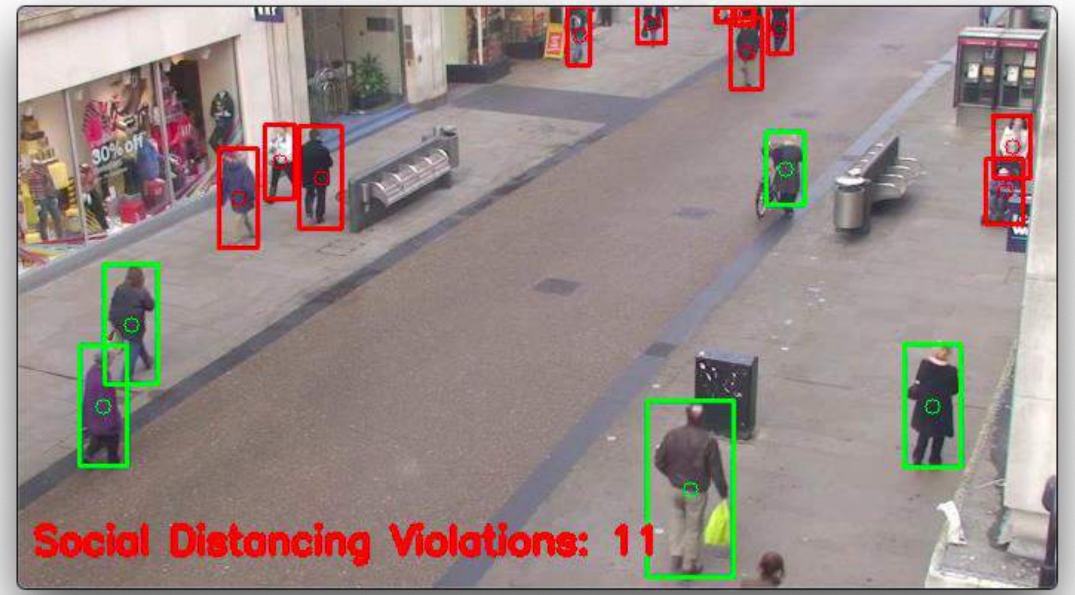


Fig. 5.3.3 output processed sample frame

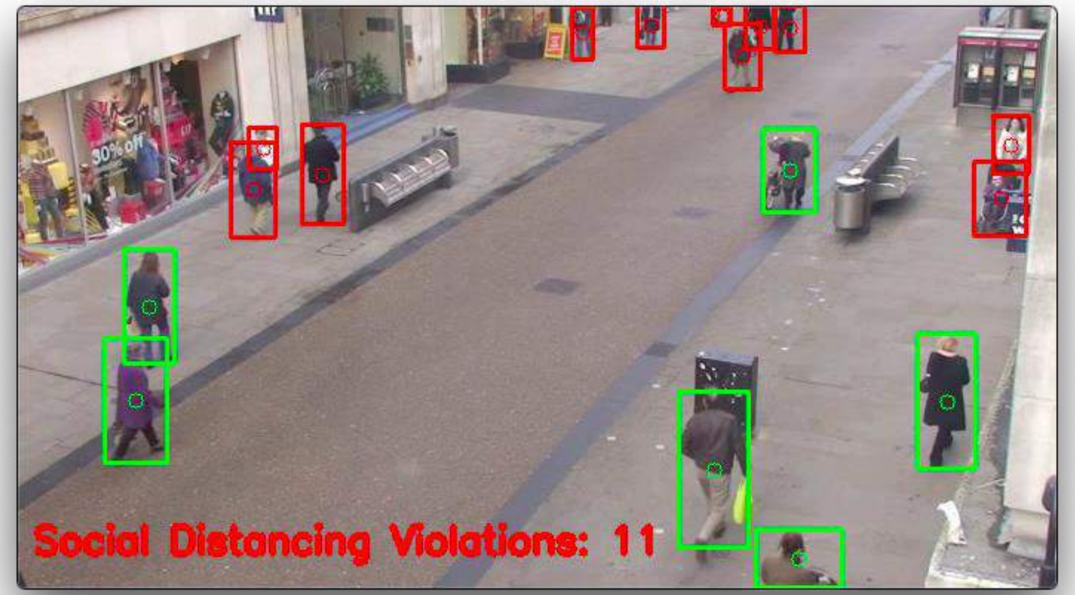


Fig. 5.3.4 output processed sample frame

6.TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that
although the components were individually satisfied, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.

Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes.

6.3 TEST CASES

Test Case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application. The test cases are displayed in the Screenshots chapter. The input frames of the input sample video given by the CCTV are the sample test cases.

7. CONCLUSION

7.1 PROJECT CONCLUSION

In this work, a deep learning-based social distance monitoring framework is presented using an overhead perspective. The pre-trained YOLOv3 paradigm is used for human detection. As a person's appearance, visibility, scale, size, shape, and pose vary significantly from an overhead view, the transfer learning method is adopted to improve the pre-trained model's performance. The model is trained on an overhead data set, and the newly trained layer is appended with the existing model. To the best of our knowledge, this work is the first attempt that utilized transfer learning for a deep learning-based detection paradigm, used for overhead perspective social distance monitoring. The detection model gives bounding box information, containing centroid coordinates information. Using the Euclidean distance, the pairwise centroid distances between detected bounding boxes are measured. To check social distance violations between people, an approximation of physical distance to the pixel is used, and a threshold is defined. A violation threshold is used to check if the distance value violates the minimum social distance set or not. Furthermore, a centroid tracking algorithm is used for tracking people in the scene. Experimental results indicated that the framework efficiently identifies people walking too close and violates social distancing; also, the transfer learning methodology increases the detection model's overall efficiency and accuracy. For a pre-trained model without transfer learning, the model achieves detection accuracy of 92% and 95% with transfer learning. The tracking accuracy of the model is 95%.

7.2 FUTURE ENHANCEMENTS

The work may be improved in the future for different indoor and outdoor environments. Different detection and tracking algorithms might be used to help track the person or people who are violating or breaching the social distancing threshold.

8. BIBLIOGRAPHY

8.1 REFERENCES

1. Adlhoch, C. (2020). https://www.ecdc.europa.eu/sites/default/files/documents/covid-19-social-distancing-measuresg-guide-second-update.pdf.
2. Adolph C., Amano K., Bang-Jensen B., Fullman N., Wilkerson J. medRxiv. 2020 [Google Scholar]
3. Ahmad M., Ahmed I., Ullah K., Khan I., Adnan A. 2018 9th IEEE annual ubiquitous computing, electronics mobile communication conference (UEMCON) 2018. pp. 746–752. [CrossRef] [Google Scholar]
4. Ahmad M., Ahmed I., Ullah K., Khan I., Khattak A., Adnan A. International Journal of Advanced Computer Science and Applications. 2019;10 doi: 10.14569/IJACSA.2019.0100367. [CrossRef] [Google Scholar]
5. Ahmad M., Ahmed I., Khan F.A., Qayum F., Aljuaid H. International Journal of Distributed Sensor Networks. 2020;16 1550147720934738. [Google Scholar]
6. Ahmed I., Adnan A. 2017. Cluster computing; pp. 1–22. [Google Scholar]
7. Ahmed I., Ahmad A., Piccialli F., Sangaiah A.K., Jeon G. IEEE Internet of Things Journal. 2018;5:1598–1605. [Google Scholar]
8. Ahmed I., Ahmad M., Nawaz M., Haseeb K., Khan S., Jeon G. Computer Communications. 2019;147:188–197. [Google Scholar]
9. Ahmed I., Din S., Jeon G., Piccialli F. IEEE Internet of Things Journal. 2019 [Google Scholar]
10. Ahmed I., Ahmad M., Adnan A., Ahmad A., Khan M. International Journal of Machine Learning and Cybernetics. 2019:1–12. [Google Scholar]
11. Ainslie K.E., Walters C.E., Fu H., Bhatia S., Wang H., Xi X. Wellcome Open Research. 2020;5 [Google Scholar]
12. B. News (2020). Online. https://www.bbc.co.uk/news/world-asia-china51217455, (Accessed 23 January 2020).
13. Brunetti A., Buongiorno D., Trotta G.F., Bevilacqua V. Neurocomputing. 2018;300:17–33. [Google Scholar]
14. Chakraborty B.A. 2021. Springer. [Google Scholar]
15. Chakraborty C., Banerjee A., Garg L., Coelho Rodrigues J.J.P. Series Studies in Big Data. 2021;80:98–136. doi: 10.1007/978-981-15-8097-0. [CrossRef] [Google Scholar]
16. Choi J.-W., Moon D., Yoo J.-H. ETRI Journal. 2015;37:551–561. [Google Scholar]
17. Ferguson N.M., Cummings D.A., Cauchemez S., Fraser C., Riley S., Meeyai A. Nature. 2005;437:209–214. [PubMed] [Google Scholar]
18. Girshick R. Proceedings of the IEEE international conference on computer vision. 2015. pp. 1440–1448. [Google Scholar]
19. Girshick R., Donahue J., Darrell T., Malik J. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014. pp. 580–587. [Google Scholar]

20. Online. https://www.health.harvard.edu/diseases-and-conditions/preventing-the-spread-of-the-coronavirus (Accessed 18 August 2020).
21. Harvey A., LaPlace J. 2019. Megapixels: Origins, ethics, and privacy implications of publicly available face recognition image datasets. [Google Scholar]
22. Iqbal M.S., Ahmad I., Bin L., Khan S., Rodrigues J.J. 2020. Transactions on Emerging Telecommunications Technologies; p. e4017. [Google Scholar]
23. Krizhevsky A., Sutskever I., Hinton G.E. Advances in neural information processing systems. 2012. pp. 1097–1105. [Google Scholar]
24. Lin T.-Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D. European conference on computer vision. Springer; 2014. pp. 740–755. [Google Scholar]
25. Migniot C., Ababsa F. Journal of Real-Time Image Processing. 2016;11:769–784. [Google Scholar]
26. N. H. C. of the Peoples Republic of China (2020). Online. http://en.nhc.gov.cn/2020-03/20/c78006.htm (Accessed 20 March 2020).
27. Nguyen C.T., Saputra Y.M., Van Huynh N., Nguyen N.-T., Khoa T.V., Tuan B.M. 2020. Enabling and emerging technologies for social distancing: a comprehensive survey and open problems.arXiv:2005.02816 [Google Scholar]
28. Patrick S.P., dos Santos R.S., de Souza L.B.M. 22nd International Conference on E-Health Networking, Applications and Services (IEEE Healthcom 2020); Shenzhen, China, December 12–15; 2020. [Google Scholar]
29. Pouw C.A., Toschi F., van Schadewijk F., Corbetta A. 2020. Monitoring physical distancing for crowd management: Real-time trajectory and group analysis.arXiv:2007.06962 [PMC free article] [PubMed] [Google Scholar]
30. Prem K., Liu Y., Russell T.W., Kucharski A.J., Eggo R.M., Davies N. The Lancet Public Health. 2020 [Google Scholar]
31. Punn N.S., Sonbhadra S.K., Agarwal S. medRxiv. 2020 [Google Scholar]
32. Punn N.S., Sonbhadra S.K., Agarwal S. 2020. Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques.arXiv:2005.01385 [Google Scholar]
33. Ramadass L., Arunachalam S., Sagayasree Z. International Journal of Pervasive Computing and Communications. 2020 [Google Scholar]
34. Redmon J., Divvala S., Girshick R., Farhadi A. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. pp. 779–788. [Google Scholar]
35. Redmon J., Farhadi A. 2018. YOLOv3: An incremental improvement.arXiv:1804.02767 [Google Scholar]
36. Ren S., He K., Girshick R., Sun J. Advances in neural information processing systems. 2015. pp. 91–99. [Google Scholar]
37. Robakowska M., Tyranska-Fobke A., Nowak J., Slezak D., Zuratynski P., Robakowski P. Disaster and Emergency Medicine Journal. 2017;2:129–134. [Google Scholar]
38. Sathyamoorthy A.J., Patel U., Savle Y.A., Paul M., Manocha D. 2020. COVID-robot: Monitoring social distancing constraints in crowded scenarios.arXiv:2008.06585 [Google Scholar]

39. Simonyan K., Zisserman A. 2014. Very deep convolutional networks for large-scale image recognition.arXiv:1409.1556 [Google Scholar]
40. Online. https://www.statista.com/chart/21198/effect-of-social-distancing-signer-lab/ (Accessed 18 August 2020).
41. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. pp. 1–9. [Google Scholar]
42. W. C. D. C. Dashboard (Online). https://covid19.who.int/ (Accessed 23 August 2020).
43. W.H. Organization (2020). https://www.who.int/emergencies/diseases/novel-corona-virus-2019 (Accessed 02 May 2020).
44. WHO (Online). https://www.who.int/dg/speeches/detail/2020 (Accessed 12 March 2020).
45. Yang D., Yurtsever E., Renganathan V., Redmill K.A., Özgüner Ü. 2020. A vision-based social distancing and critical density detection system for COVID-19.arXiv:2007.03578 [Google Scholar]
46. Yash Chaudhary D.G., Mehta M. 22nd international conference on E-health networking, applications and services (IEEE Healthcom 2020); Shenzhen, China, December 12–15, 2020; 2020. [Google Scholar]

Github link:

https://github.com/Ankith-Sai/Social-Distance-Prediction-

IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed Multi-disciplinary Journal

Certificate of Publication

K. Ankith Sai Reddy

CMR Technical Campus

TITLE OF PAPER

Social Distance Predictor Using Deep Learning

has been published in

IJARESM, Impact Factor: 7.429, Volume 9 Issue 6, June- 2021

Paper Id: IJARESM/June21

Date: 11-06-2021

CERTIFIED ISO 9001:2008 JOURNAL

Website: www.ijaresm.com
Email: editor.ijaresm@gmail.com

IJARESM Auth. Signatory INDIA

[illegible]

Authorized Signatory

IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed Multi-disciplinary Journal

Certificate of Publication

V. Prashanth

CMR Technical Campus

TITLE OF PAPER

Social Distance Predictor Using Deep Learning

has been published in

IJARESM, Impact Factor: 7.429, Volume 9 Issue 6, June- 2021

Paper Id: IJARESM/June21

Date: 11-06-2021

Website: www.ijaresm.com
Email: editor.ijaresm@gmail.com

IJARESM INDIA Auth. Signatory

[illegible]

Authorized Signatory

IJARESM

ISSN: 2455-6211, New Delhi, India
International Journal of All Research Education & Scientific Methods
An ISO & UGC Certified Peer-Reviewed Multi-disciplinary Journal

Certificate of Publication

Ch. Dilip Goud

CMR Technical Campus

TITLE OF PAPER

Social Distance Predictor Using Deep Learning

has been published in

IJARESM, Impact Factor: 7.429, Volume 9 Issue 6, June- 2021

Paper Id: IJARESM/June21

Date: 11-06-2021

CERTIFIED ISO 9001:2008 JOURNAL

Website: www.ijaresm.com
Email: editor.ijaresm@gmail.com

IJARESM Auth. Signatory INDIA

[illegible]

Authorized Signatory

International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 9, Issue 6, June -2021, Impact Factor: 7.429, Available online at: www.ijaresm.com

Social Distance Predictor Using Deep Learning

D. Vigneshwar Rao [1], K. Ankith Sai Reddy[2], V. Prashanth[3], Ch. Dilip Goud[4]

[1]Assistant Professor, CMR Technical Campus
[2,3,4]CMR Technical Campus

--*****************--

ABSTRACT

The objective of the model is to make the detection of social distance between people who are roaming on the roads and the footpaths or anywhere outdoors or in the society easy and more reliable. We know that when a naked human eye monitors the, accuracy is considerably low and also a human eye cannot monitor more than one group at a time. Therefore this system is introduced to make the process easy, accurate and consistent in monitoring people at all times by using 24/7 surveillance cameras. This system is using the YoloV3 (You Only Look Once) which is the fastest object detection algorithm and contains several classes of objects for object detection. This algorithm is not only used for object detection but also used in finding the position of the object and tracking the movement of the object from one position to another. This feature will be used in finding the distance of the objects whose coordinates will further help in finding out the distance between each object in real time and determine whether they are following social distancing or not.

Keywords: Social Distance, Object Detection Algorithm, YOLO.

INTRODUCTION

The project titled as "Social Distance Predictor" is a Machine Learning based application which trains itself and improves every time on new scenarios. This project aims to design a social distance predictor which verifies the distance between two or more people is safe and is according to social distance norms provided by the government body. This Project helps monitor social distance between people in real time and help admins warn the people at public places. This system is more efficient, effective and accurate. At these hard times of covid, this helps in reducing immense man power for monitoring people and also drastically increases the accuracy to help people maintain social distance. Covid - 19 is a deadly and a very dangerous disease which is causing chaos in every country and every place possible in this world. There have been strict rules created by the national body and world organizations to help prevent this deadly virus from spreading into communities. One of which is to maintain Social Distance, people must maintain a distance of 1.5 Meters between each other in order to maintain social distance. We know that in public places where there is a huge crowd it is difficult to monitor people who maintain social distance especially in areas where there is a dense population. In order to do so we have created this system which is based on a machine learning model where it measures the social distance between two or more people in real time with the help of CCTV Cameras. The live footage from the CCTV cameras will go to the system where the system processes the social distance between two people and sends the output video to the admin and thus the admin could announce and take preliminary precautions.

Overview

The ongoing COVID-19 corona virus outbreak has caused a global disaster with its deadly spreading. Due to the absence of effective remedial agents and the shortage of immunizations against the virus, population vulnerability increases. In the current situation, as there are no vaccines available; therefore, social distancing is thought to be an adequate precaution (norm) against the spread of the pandemic virus. The risks of virus spread can be minimized by avoiding physical contact among people. The purpose of this work is, therefore, to provide a deep learning platform for social distance tracking using an overhead perspective. The framework uses the YOLOv3 object recognition paradigm to identify humans in video sequences. The transfer learning methodology is also implemented to increase the accuracy of the model.

MATHEMATICAL FORMULATION

$$\mathrm{Conf}(p)=\Pr(p)\times \mathrm{IOU}(pred,actual)$$

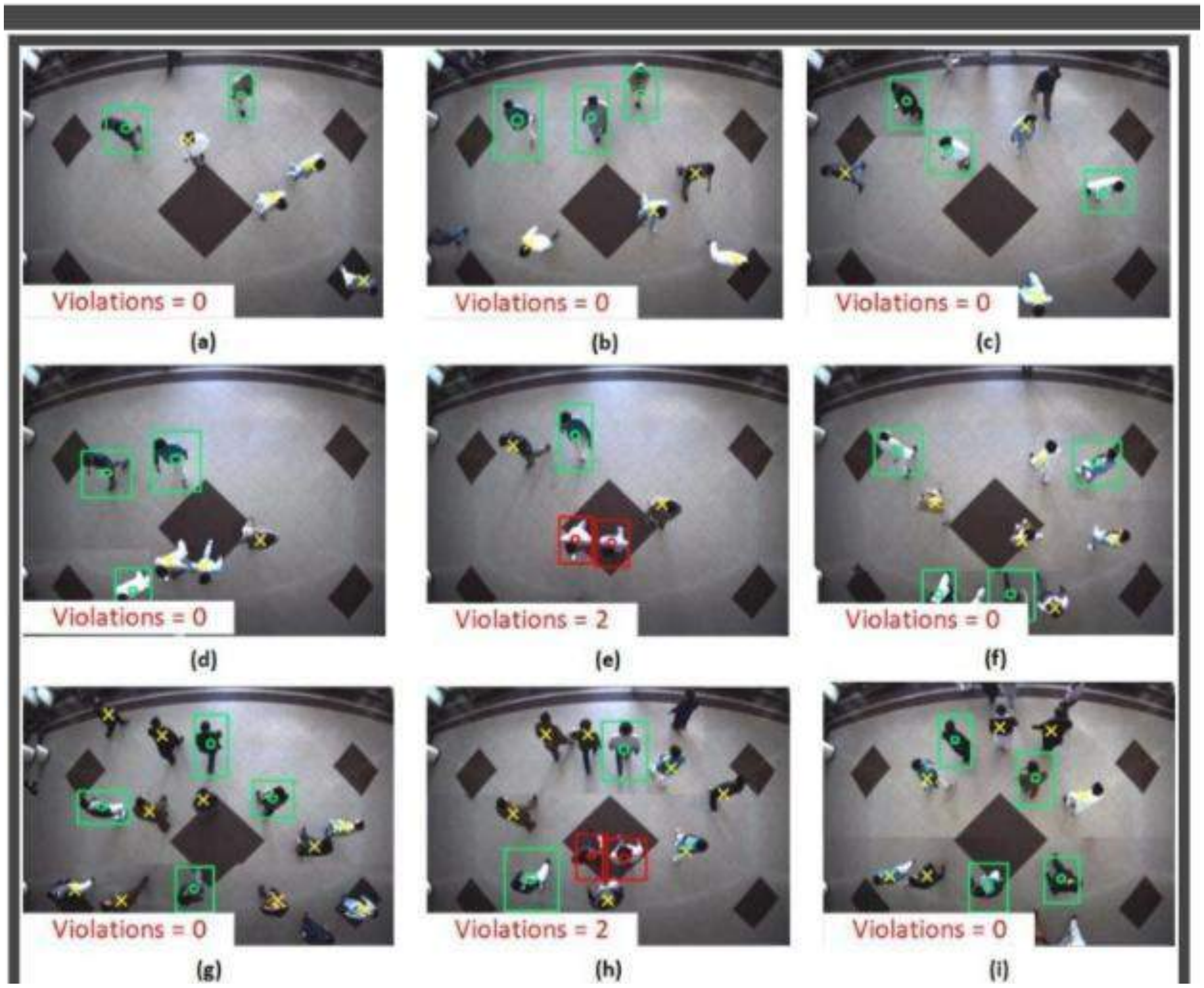
$$\mathrm{IoU}(pred,actual)=areaBoxT\cap BoxP/BoxT\cup BoxP.$$

Where;

- Pr(p) indicates whether the person present is in the detected bounding box or not.
- IoU(pred,actual) determines the Intersection Over Union of the actual and predicted bounding box.
- where the ground truth box (actual) manually labeled in the training data set is represented with BoxT, and the predicted bounding box is displayed as BoxP.

ANALYSIS

The testing results of the social distance framework using a pre-trained model (Redmon & Farhadi, 2018) has been visualized. The testing results are evaluated using different video sequences. The people in the video sequences are freely moving in the scenes; it can be seen from sample frames that the individual's visual appearance is not identical to the frontal or side view. The person's size is also varying at different locations. Since the model only considers human (person) class; therefore, only an object having an appearance like a human is detected by a pre-trained model. The pre-trained model delivers good results and detects various size person bounding boxes, as shown with green rectangles.



CONCLUSION

In this work, a deep learning-based social distance monitoring framework is presented using an overhead perspective. The pre-trained YOLOv3 paradigm is used for human detection. As a person's appearance, visibility, scale, size, shape, and pose vary significantly from an overhead view, the transfer learning method is adopted to improve the pre-trained model's performance. The model is trained on an overhead data set, and the newly trained layer is appended with the existing model. To the best of our knowledge, this work is the first attempt that utilized transfer learning for a deep learning-based detection paradigm, used for overhead perspective social distance monitoring. The detection model gives bounding box information, containing centroid coordinates information. Using the Euclidean distance, the pairwise centroid distances between detected bounding boxes are measured. To check social distance violations between people, an approximation of physical distance to the pixel is used, and a threshold is defined. A violation threshold is used to check if the distance value violates the minimum social distance set or not. Furthermore, a centroid tracking algorithm is used for tracking people in the scene. Experimental results indicated that the framework efficiently identifies people walking too close and violates social distancing; also, the transfer learning methodology increases the detection model's overall efficiency and accuracy. For a pre-trained model without transfer learning, the model achieves detection accuracy of 92% and 95% with transfer learning. The tracking accuracy of the model is 95%.

REFERENCES

[1] Adlhoch,C.(2020). https://www.ecdc.europa.eu/sites/default/files/documents/covid19-social-distancing-measuresg-guide-second-update.pdf.

[2] Adolph C., Amano K., Bang-Jensen B., Fullman N., Wilkerson J. medRxiv. 2020 [Google Scholar]

[3] Ahmad M., Ahmed I., Ullah K., Khan I., Adnan A. 2018 9th IEEE annual ubiquitous computing, electronics mobile communication conference (UEMCON) 2018. pp. 746–752. [CrossRef] [Google Scholar]

[4] Ahmad M., Ahmed I., Ullah K., Khan I., Khattak A., Adnan A. International Journal of Advanced Computer Science and Applications. 2019;10 doi: 10.14569/IJACSA.2019.0100367. [CrossRef] [Google Scholar]

[5] Ahmad M., Ahmed I., Khan F.A., Qayum F., Aljuaid H. International Journal of Distributed Sensor Networks. 2020;16 1550147720934738. [Google Scholar]

[6] Ahmed I., Adnan A. 2017. Cluster computing; pp. 1–22. [Google Scholar]

[7] Ahmed I., Ahmad A., Piccialli F., Sangaiah A.K., Jeon G. IEEE Internet of Things Journal. 2018;5:1598–1605. [Google Scholar]

[8] Ahmed I., Ahmad M., Nawaz M., Haseeb K., Khan S., Jeon G. Computer Communications. 2019;147:188–197. [Google Scholar]

[9] Ahmed I., Din S., Jeon G., Piccialli F. IEEE Internet of Things Journal. 2019 [Google Scholar]

[10] Ahmed I., Ahmad M., Adnan A., Ahmad A., Khan M. International Journal of Machine Learning and Cybernetics. 2019:1–12. [Google Scholar] 11. Ainslie K.E., Walters C.E., Fu H., Bhatia S., Wang H., Xi X. Well