Deep Learning Based Crowd Monitoring And Person Identification System

by

Mohammad Fahim Haque
18201141
Dipto Sarkar
18201182
Tawhid Al Muhaimin Choudhury
18201191
Samiul Hoque Rafi
18201178
Md Shajidur Rahim
18101535

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
September 2023

Declaration

It is hereby declared that

1. The Final thesis report submitted is our own original work while completing our Computer Science and Engineering degree at BRAC University.
2. The Final thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The Final thesis does not contain material that has been accepted or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Fahim

Mohammad Fahim Haque
18201141

Dipto

Dipto Sarkar
18201182

[illegible]

Tawhid Al Muhaimin Choudhury
18201191

[illegible]

Samiul Hoque Rafi
18201178

[illegible]

Md Shajidur Rahim
18101535

Approval

The Final thesis report titled "Deep Learning Based Crowd Monitoring And Person Identification System" was submitted by

1. Mohammad Fahim Haque(18201141)
2. Dipto Sarkar(18201182)
3. Tawhid Al Muhaimin Choudhury(18201191)
4. Samiul Hoque Rafi(18201178)
5. Md Shajidur Rahim(18101535)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on September 25, 2023.

Examining Committee:

Supervisor:
(Member)

Prof. Dr. Amitabha Chakrabarty, PhD
Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

In this paper, we propose a deep learning-based crowd monitoring and person identification system and a crowd-video dataset to address the challenges posed by the recent COVID-19 pandemic or future pandemics may occur, where maintaining social distance in public places is necessary. The system can be also implemented where crowd monitoring is necessary. For instance, public places like bank booths, airports, train stations, hospitals, tourist attractions, public transportation hubs, stadiums, arenas, etc. The system combines person tracking and social distance measurements to accurately detect individuals who may unintentionally violate the rules due to a lack of spatial awareness. To implement the system, a custom dataset was created to evaluate and tackle perspective correction and person-only detection issues. Three popular object detection models: YOLOv8, YOLOv7, and Faster R-CNN with and without the DeepSort tracking algorithm were used and a comparison of their performances is demonstrated. To build our system we took two different approaches. In the first approach, we used Faster R-CNN & YOLOv8 for person identification, and for tracking, we used the SORT tracking algorithm. In the second approach, we used YOLOv7 & YOLOv8 for person detection and DeepSort-based tracking algorithm which generates unique IDs and successfully tracks and re-identifies each person frame by frame using Kalman filter and Hungarian algorithm. The experimental results show that all models can accurately detect humans with 90% accuracy and estimate the distance between them. However, Faster R-CNN falls short in real-time human detection, whereas YOLOv8 outperforms YOLOv7 in terms of speed and detection accuracy. Still, YOLOv8 is relatively new and has less support for implementation. Thus, YOLOv7 is chosen for implementation in mobile, micro-controller-based, or IoT devices, as it offers better support for immediate implementation. The proposed system is efficient, accurate, and does not require human supervision. It includes a log system to track violations with frame rates and unique IDs. The system was tested using our custom dataset, and positive results were achieved, indicating its potential usefulness in crowd monitoring and social distance enforcement.

Keywords: COVID-19; Faster R-CNN; YOLOv8; YOLOv7; DeepSort; IoT; micro-controller

Acknowledgement

First and foremost, we would want to express our gratitude to Almighty Allah for enabling us to finish our thesis on schedule and without hindrance.

Having stated that, we want to thank Prof. Dr. Amitabha Chakrabarty(PhD), our esteemed lecturer and supervisor, for his steadfast encouragement and vigilant supervision, which enabled us to finish our assignment. We also want to express our gratitude to our devoted friends for standing by us during the tough times. Finally, without our parents' constant support, it might not be conceivable. We are very close to graduation thanks to their kind help and prayers.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$CGVPR$	Color to Gray Video Person Re-identification
CNN	Convolutional Neural Network
$COCO$	Common Objects In Context dataset
CV	Computer Vision
DL	Deep Learning
$FRCNN$	Faster Region-based Convolutional Neural Networks
GAP	Global Average Pooling
GMP	Global Max Pooling
HOG	Histogram of Oriented Gradients
HSV	Hue Saturation Value
$LSVM$	Lagrangian Support Vector Machine
PCB	Printed Circuit Board
$ROIF$	Region-of-Interest based Features
$SDPL$	Semicoupled Dictionary Pair Learning
WHO	World Health Organization
$YOLO$	You Only Look Once

Chapter 1

Introduction

1.1 Background

The term ”Technology” and its applications have changed significantly since the 20th century and have continued to change through time. Our world is increasingly dominated by technology. The development of human civilization has been significantly influenced by both technological and cultural advancements. Technology offers creative solutions to do tasks through a variety of clever and imaginative methods.

All of those technologies could not prevent COVID-19 and its devastating effects. Technology continues to play the biggest role whenever a situation calls for it. The most important preventive measure for controlling efforts to halt the development of COVID-19 is social distancing. The World Health Organization (WHO) recommends that people should keep a distance of at least one meter between each other to prevent the spread of this disease. And now technology enters the picture, bringing with it research on a variety of cutting-edge techniques for social isolation across a range of segments. Because coronavirus disease has had such a detrimental impact on the global economy, this research aims to lessen the disease's effects while reducing resource consumption. However, the approach of this research is unique as the system that will be used in this research can be customizable to be used in other similar situations where person detection and tracking is a must like bank booths, educational institutions, hospitals, and many others.

Therefore, in this research paper, we proposed an approach how to monitor and track social distance, person identification, and overall crowd monitoring system. We also developed a custom dataset for training and evaluating our system through deep learning. For this, we have used a deep sort tracking algorithm-based system using deep learning models like YOLOv8, YOLOv7, and FRCNN to perform a comparative analysis and present the best solution based on the current technological support to implement the system in an IoT device to perform in real-world scenarios.

1.2 Problem Statement

The COVID outbreak in this globe has had a variety of effects on the world, including economic ones. Worldwide, millions of people perished as a result of this disease's protracted uncontrollable devastation. The concept of social distancing has become more important for the implementation of many systems in all fields of endeavor, particularly in computer vision and deep learning. However, previous computer-vision-based systems have significant flaws. For instance, low light conditions are ignored, and methods for mapping pixel distance to geometric units are not well developed. Therefore, it is difficult to manually detect and maintain social distance in nations with dense populations, making it difficult to avert pandemic conditions.

To stop the transmission of the virus and safeguard public health, it is crucial to maintain social distancing and surveillance procedures. Manually monitoring these measures, however, is expensive, time-consuming, and error-prone. To detect, track, and evaluate human activities and behaviors in a variety of settings, including crowded locations, workplaces, educational institutions, and public transportation, it is necessary to develop an automated system that can use computer vision techniques.

Such a system can help decision- and policy-making processes by offering real-time feedback, alerts, and statistics to the public and the government. The main difficulties in creating such a system are: (1) how to accurately and robustly detect and recognize humans and their poses, gestures, facial expressions, and interactions in complex and dynamic environments; (2) how to measure and quantify the level of social distance and adherence to the surveillance rules; and (3) how to deal with the ethical, privacy, and security issues associated with the collection and processing of sensitive data. The purpose of this thesis is to address these issues by offering new computer vision techniques and algorithms that can track social isolation and surveillance utilizing a variety of cameras and sensors with our novel dataset. The expected outcomes of this thesis are (1) a thorough review of the literature on the state-of-the-art methods and applications of computer vision for social distancing and surveillance; (2) a novel framework that can integrate various data sources and perform multi-modal analysis of human activities and behaviors; (3) a set of experiments and evaluations that can show the efficacy and efficiency of the suggested methods and algorithms based on our custom dataset (4) a detailed analysis of the results.

Firstly, in a multiple or non-overlapping camera network, re-identification is very challenging. Because multiple cameras will overlap differentiating the surveillance can be a lengthy process. Additionally, face detection and reidentification from video surveillance is a significant obstacle. Because we have to crop images from video clips frame by frame and then detection and identification can be done. Finally, reidentification based on visual appearance is a huge task because of the many variations of visual appearance and different angles of the cameras. Also, detection and identification in an environment with pedestrians with many various appearances can be a heckle. Moreover, the angle of the cameras may vary a lot as different angles may cause different widths and heights of the persons, and distance measurements may vary according to that.

Sometimes the re-identification process becomes a challenging task as visual appearance can be very dynamic. To solve visual appearance variation which is caused by various backgrounds, illumination, and variation poses these kinds of challenges, we can use a method called ROIF which can make person re-identification more accurate in comparatively less time by combining textural and chromatic features. In this research, an efficient way for the ROIF method was shown and the implementation was done by HSV histogram and chromatic content[1].

In [2], sometimes the surveillance camera used to detect persons can be set to gray mode due to special cases like saving batteries or some other cases. In this case, for re-identification between true-color and grayscale mode pedestrian videos, we use a method called CGVPR. We can use the SDPL approach to re-identify persons in these situations to do it more efficiently.

In [3], Zhu points out that the DL-based person ReID method's main drawback is the lack of training sets. Very deep models require very big training sets to work effectively. To solve this problem we can use body shape similarity. A walking person can be identified faster if we include body shape similarity and parts location in our identification model. We can implement these features to enhance our ReID system's performance.

1.3 Research Objective

The purpose of this research is to create a program for crowd monitoring to enforce social distancing based on deep learning and computer vision and to populate a custom dataset to test the program. Computer vision techniques will be implemented to improve the accuracy of the program. The objectives are:

1. To create a system that can be implemented on IoT devices.
2. To create a system that can be customizable so that it can be used in other similar situations where social distancing and object tracking is a must(Bank booth, Hospitals, Institution, Factory, etc.)
3. To develop a custom dataset for training and evaluating the system.
4. To ensure social stability during times of pandemic such as COVID-19.
5. To identify the people who do not follow the rules of social distancing.
6. To reduce COVID-19 or other contagious disease infection rate.
7. Monitoring and tracking of the public space to ensure that only authorized individuals enter and exit it.
8. Raising people's awareness to help stop COVID-19, other contagious viruses, and flu from spreading.

1.4 Research contributions

In our paper, we managed to make a system where deep sort is used for tracking where we both track and generate unique IDs for persons that are detected using YOLOv8 and YOLOv7. We also performed Faster R-CNN to identify individuals and measure distance. Therefore, we specifically add the following contributions to this paper based on our research-

- We develop a program for crowd monitoring to enforce social distancing with deep learning and computer vision based on our comparative analysis which is customizable for other similar situations.
- We develop a custom dataset for training and evaluating the system and compare the results using performance metrics.
- We present a comparative analysis of the proposed system we designed to monitor and track social distancing based on three deep learning models: YOLOv8, YOLOv7, and R-CNN.
- We make our code public with the implementation processes documented for further development.
- We optimize the system code for computer vision-based IoT devices and similar projects.

Chapter 2

Overview of Social Distance Monitoring System

A social distance monitoring system may consist of several deep-learning models and tracking algorithms. The overall system varies between different approaches. A social distance monitoring system tracks and analyzes the space between people in public areas to make sure that social distancing rules are being followed. It uses video analysis algorithms and computer vision techniques to track and recognize people, evaluate their distances, and produce alerts or visual cues when people are not keeping a safe distance.

2.1 Person detection algorithms

The Viola-Jones object detection framework[4] and Histogram of Oriented Gradients (HOG) are two well-known machine learning methods for object detection. Face detection was the main reason for the introduction of the earlier framework. The three key steps of the method were the integral image, the Adaboost classifier, and the Classifier Cascade. But Robert K. McConnell of Wayland Research Inc. unveiled HOG in 1986. First, cells are used to split the picture into smaller sections. Each pixel in the cell is then assigned a HOG, and the sum of all the histograms is referred to as a descriptor. Typical classifiers feed these descriptions as features. SVM was used by Dalal and Triggs as a classifier to categorize an item using these attributes[5]. Figure 1 shows the evolution of several significant object identification techniques throughout time. Researchers focus on single-stage object detectors, particularly YOLO algorithms, because of the high model complexity and significant resource consumption of two-stage object detectors.

Deep learning is excelling in a variety of disciplines in addition to object recognition. To effectively handle the data relevant to healthcare, deep learning provides several models. Different sources, including X-ray, CT, and MRI, are now often used to check for potential viral infections since the Covid-19 epidemic began. The function and application of several deep models for COVID-19 infection detection are outlined in[7]. In terms of feature extraction from visual inputs, CNN has become one of the most well-liked deep learning methods. Using the Crow search

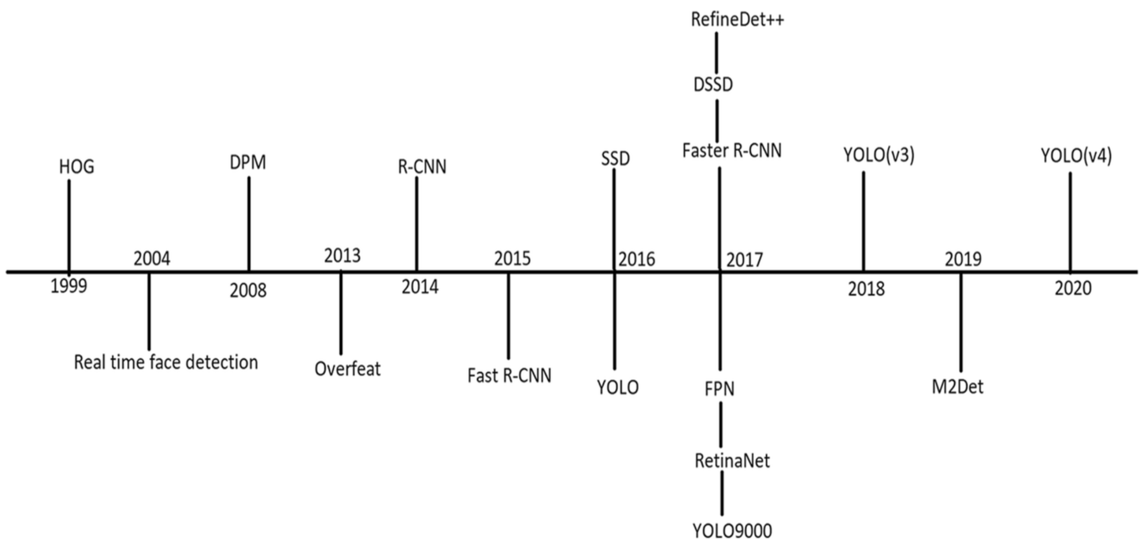


Figure 1: Year wise evolution of object detection algorithms[6]

algorithm (CSA) to find the ideal hyperparameters, for instance, the CNN-based hand identification system has achieved 100% training and testing accuracy[8].

Person identification is a very essential component of computer vision, particularly in applications like surveillance systems, IoT devices, crowd analysis, autonomous driving, and social distance measurement. These algorithms identify individuals within images or video frames. Here are some popular person detection algorithms:

2.1.1 R-CNN

A selective search-based approach was taken by et al. [9] which Ross Girshick introduced to tackle the problem of picking a large number of regions by extracting two thousand areas from images and dubbed the region proposals. So, now it has to deal with two thousand regions whereas previously it had to categorize a large number of areas in ”Object detection system overview” figure 2.

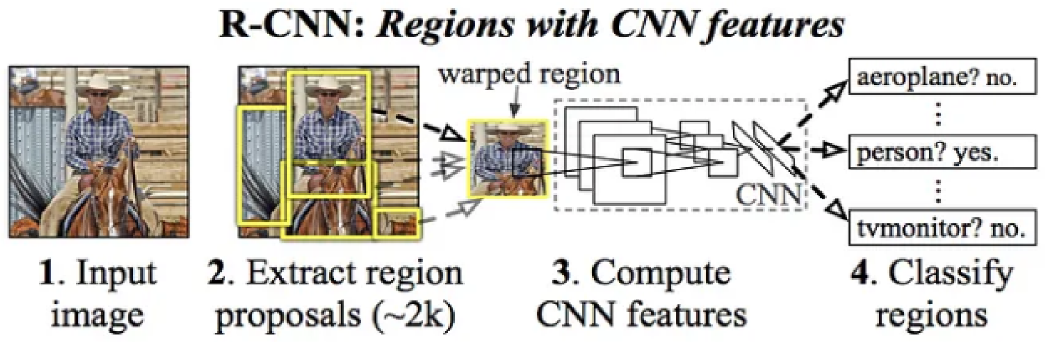


Figure 2: Object detection system overview(RCNN)

The CNN-based neural network gets the two thousand areas where suitable suggestions are disfigured into a square and also outputs a vector with 4096-dimensional qualities. The convolutional neural network in figure 2.1.1, with the function of a feature extractor and the output layer is made up of features taken from images and input into an SVM for the classification of the existence of the item in that regional proposition. The system is also capable of forecasting four offset values for further improvement of the bounding box accuracy and also determining whether an object is presented inside the region that is suggested. For example, if a suggested area occurs then the algorithm would be able to predict the person but the person's face might get sliced in that region proposal. Therefore, the offset values will help modify the proposed region's bounding box.

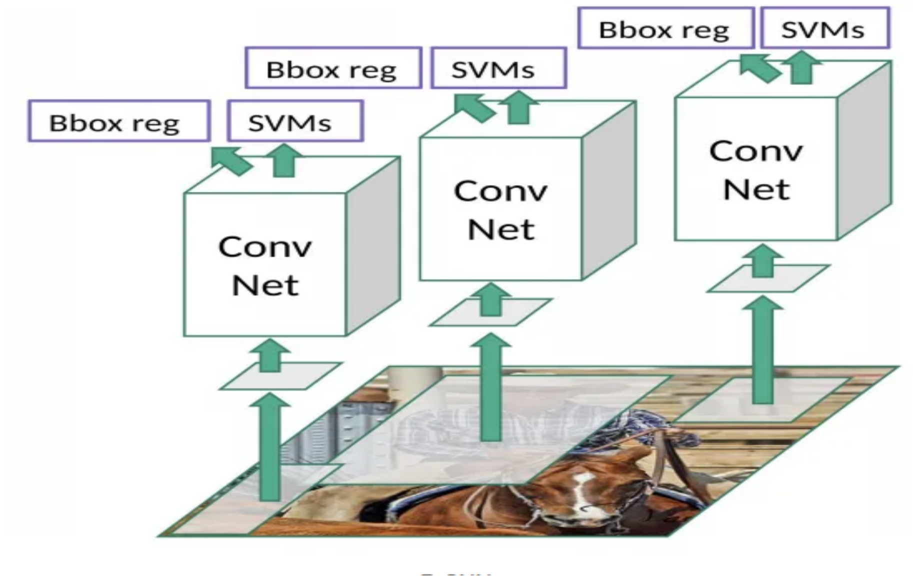


Figure 3: RCNN conv layers

2.1.2 Fast R-CNN

To tackle some drawbacks of the earlier paper et al. [9], the author introduced Fast R-CNN, which is a fast object detection method. Even though the technique has many improvements over the R-CNN algorithm, the basic structure is identical. Fast R-CNN creates a CNN feature map to input the image in a convolutional neural network. Fast R-CNN takes region-based ideas from this CNN feature map in contrast to R-CNN, which gets two thousand region proposals to the CNN one at a time. To maintain the size of these region suggestions and make them compatible with a fully linked layer, they are then warped into squares and input into a region of Interest pooling layer.

This configuration requires one forward pass through the CNN to build the convolutional feature map for the whole picture. Fast R-CNN outperforms R-CNN in terms of speed by removing the requirement to continually input several area suggestions through CNN. The approach is more effective and practical for real-time object identification applications since the feature map creation only has to be done once for the full image.

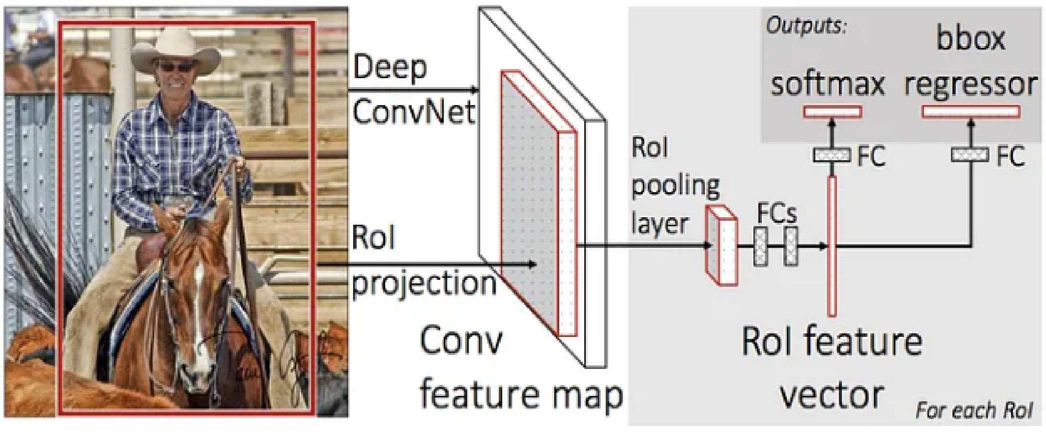


Figure 4: Fast R-CNN

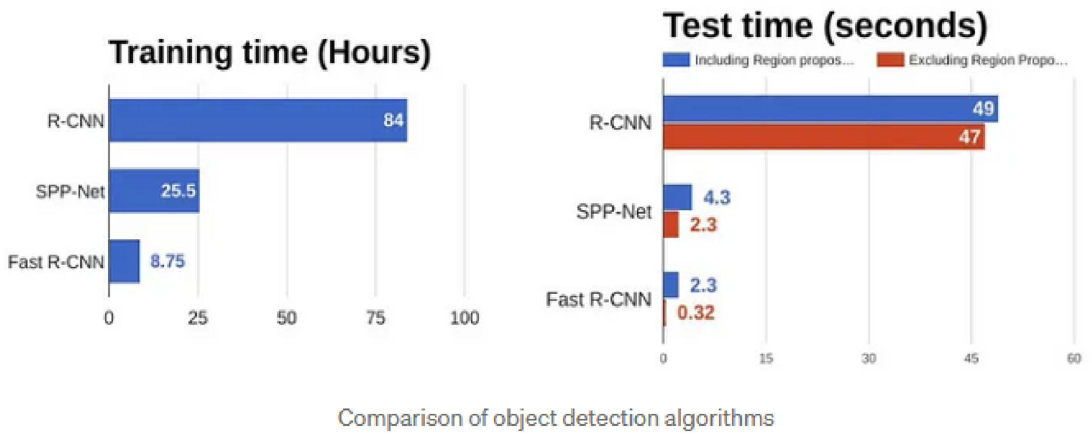


Figure 5: Training and test time comparison between object detection algorithms

2.1.3 Faster R-CNN

Selective searching is utilized by The R-CNN and fast R-CNN to generate region proposals for identifying objects. This approach has a drawback in that it requires too long and is slow, which reduces the network's overall performance. Shaoqing Ren developed a revolutionary recognition of objects technique to address this problem and do away with the necessity for selective search. Instead, this novel method enables the network to self-learn and offer regional ideas. The process of item identification becomes substantially more rapid and efficient as a result [10].

The image is input into a convolutional network, similar to Fast R-CNN, to generate a convolutional feature map. Instead of using a selective search technique on the feature map to produce the region suggestions, another network is used to anticipate the area ideas. The anticipated region suggestions are then reshaped using a RoI pooling layer to categorize the image within the proposed region and predict the offset values for the box boundaries.

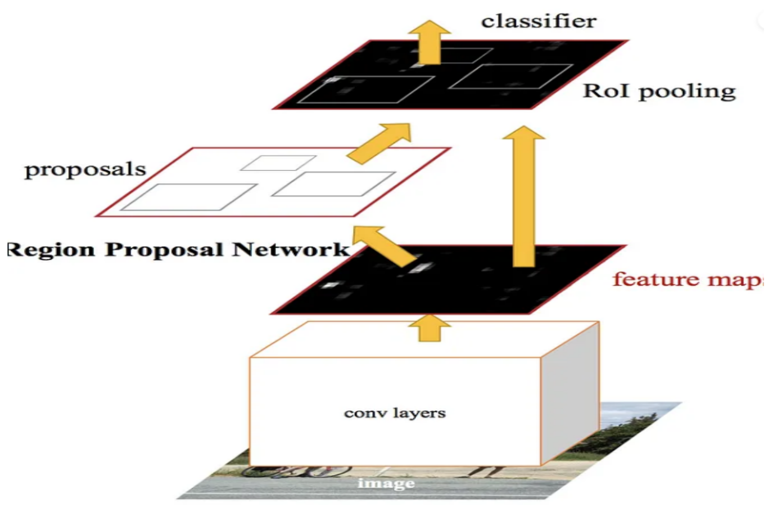


Figure 6: Faster R-CNN

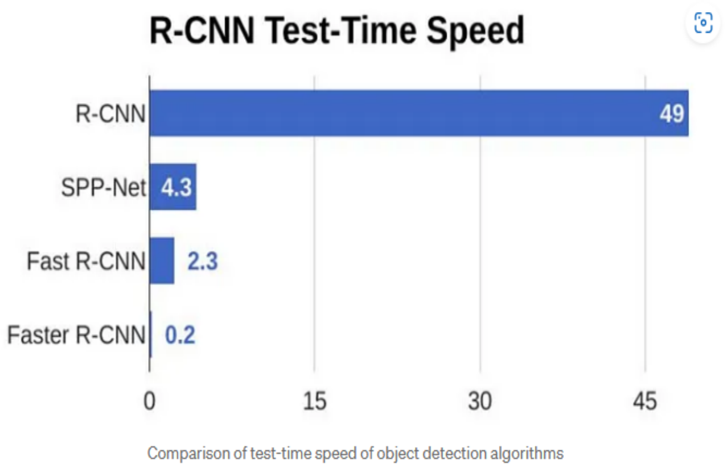


Figure 7: Test-time speed of object detection algorithms

2.1.4 YOLO

To locate the item within the picture, all of the earlier object detection methods used areas. The full picture is not viewed by the network. Rather, areas of the image with a high likelihood of containing the item. As opposed to the region-based methods mentioned above, YOLO, or You Only Look Once, is a very distinct object identification approach. One neural network in YOLO predicts the bounding boxes and the class probabilities for these boxes.

For YOLO to function, we first divide a picture into an SxS grid and then take m bounding boxes for each grid. This type of network generates a class probability and offset values for bounding every box. The item is located within the picture by selecting the bounding boxes with the class probability over a threshold value.

YOLO is considerably quicker than other object detection methods(45 frames per second). The YOLO algorithm's weakness is that it has trouble recognizing little

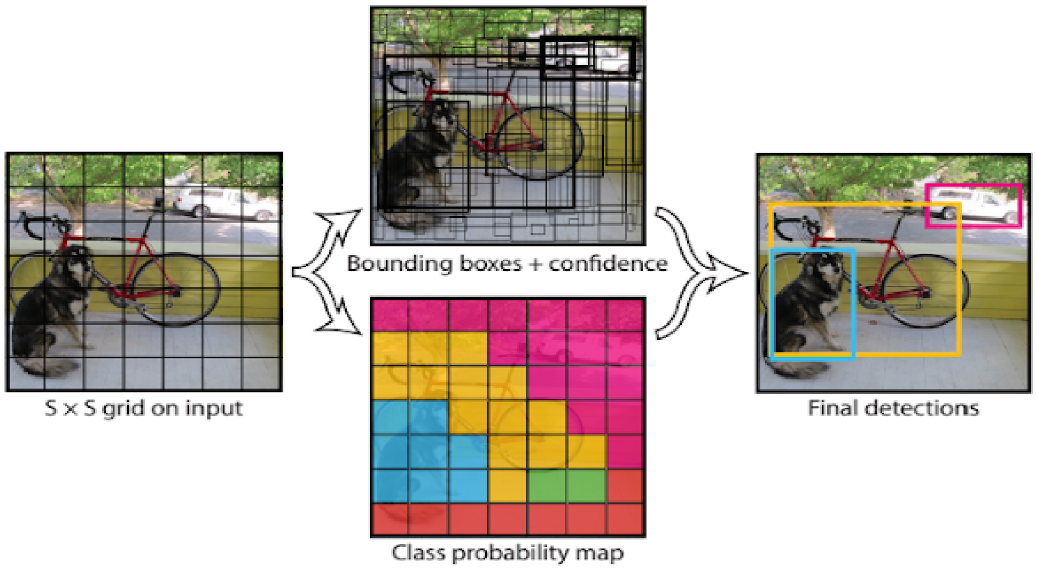


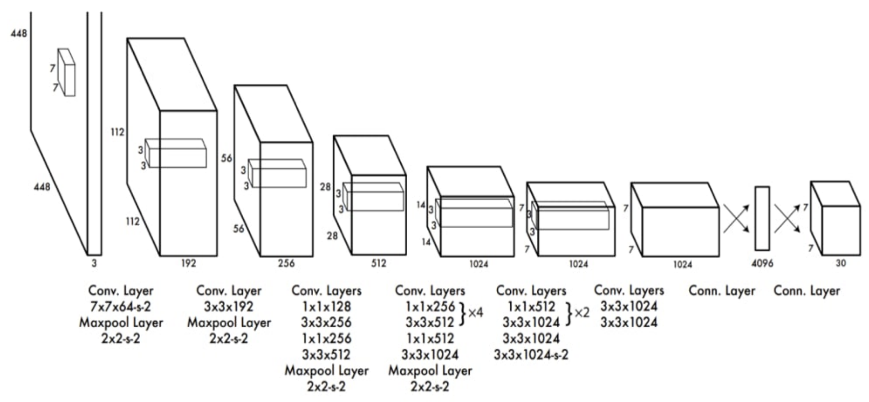
Figure 8: The yolo model

things in the image; for instance, it could have trouble spotting a flock of birds. The algorithm's geographical limitations are to blame for this. In paper [11], where a yolo algorithm model ilustration was shown in figure8.

YOLO is an object detection algorithm that revolutionized the field by introducing a new approach. Unlike traditional methods that utilize separate stages for object classification and localization, YOLO takes a unified approach by employing a single neural network to predict bounding boxes and class probabilities simultaneously. This end-to-end architecture allows YOLO to achieve real-time object detection with impressive accuracy. By making predictions in one pass through the network, YOLO reduces redundancy and achieves a good balance between speed and precision. The YOLO algorithm has been widely adopted and has become a popular choice for various computer vision applications, including autonomous vehicles, surveillance systems, and object recognition tasks.

The YOLO algorithm utilizes a straightforward deep convolutional neural network (CNN) architecture to perform object detection on input images. Here CNN model is the backbone of YOLO and holds one of the most important roles in identifying and localizing objects. In this CNN the architecture consists of many convolutional layers, which are responsible for extracting hierarchical features from the input image. These features are then fed into fully connected layers to make predictions about object classes and bounding box coordinates. By employing this CNN architecture, YOLO is capable of efficiently and accurately detecting objects in real time, making it a popular choice for various computer vision applications. This paper [11], where from the figue 9 we get a to know how the yolo model architecture formed.

During training in YOLO, it is essential to allocate a single bounding box predictor to be responsible for every object in the image. To accomplish this, yolo determines which predictor has the largest number of IOUs with a grounding truth


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 9: The yolo model architecture

bounding box. For this strategy, the predictors of this bounding box grow more specialized as every predictor becomes better at forecasting specific item sizes, aspect ratios, or classes. By assigning responsibility based on the highest IOU, YOLO improves the overall recall score by ensuring that each object is accurately predicted by the most suitable bounding box predictor.

The YOLO family of object detection models has undergone significant advancements and iterations over the years. It includes a total of 9 official versions: YOLO (2016), YOLOv2 (2017), YOLOv3 (2018), YOLOv4 (2020), YOLOv5 (2020), YOLOv6 (2021), YOLOv7 (2022), YOLOv8 (2023), and YOLO-NAS (2023). These versions represent the evolution of the YOLO algorithm, incorporating improvements in performance, accuracy, and efficiency.

2.1.5 Comparative analysis of yolo models

YOLOv2: YOLOv2 also known as YOLO9000, was released with a major upgrade and features that YOLO had. In the upgrade, YOLOv2 has an improved feature extraction and representation where it uses Darknet-19 as its CNN backbone which is another improved version of VGGNet architecture with both convolutional and pooling layers.

The main goals of YOLOv2 are to improve upon the speed and accuracy of YOLO. To do that, techniques such as multi-scale training, anchor boxes, and an improved detection system were implemented. Multi-scale training enables the model to improve its performance across many scales because it can learn from objects of different sizes. Anchor boxes were implemented to help in precise localization by predicting bounding boxes relative to anchor shapes that were defined previously.

In the new upgrade of the previous version of YOLOv2, now it can detect many object classes. It can identify a wide range of classes from different datasets using the WordTree hierarchy. Therefore this new feature that YOLOv2 has makes it a more capable object detection model with many variations.

In comparative analysis shown in [12] where in figure 10, YOLOv2 outperforms YOLO when it comes to accuracy and detection. Its improved speed, accuracy, and class detection is a significant improvement upon YOLO models.

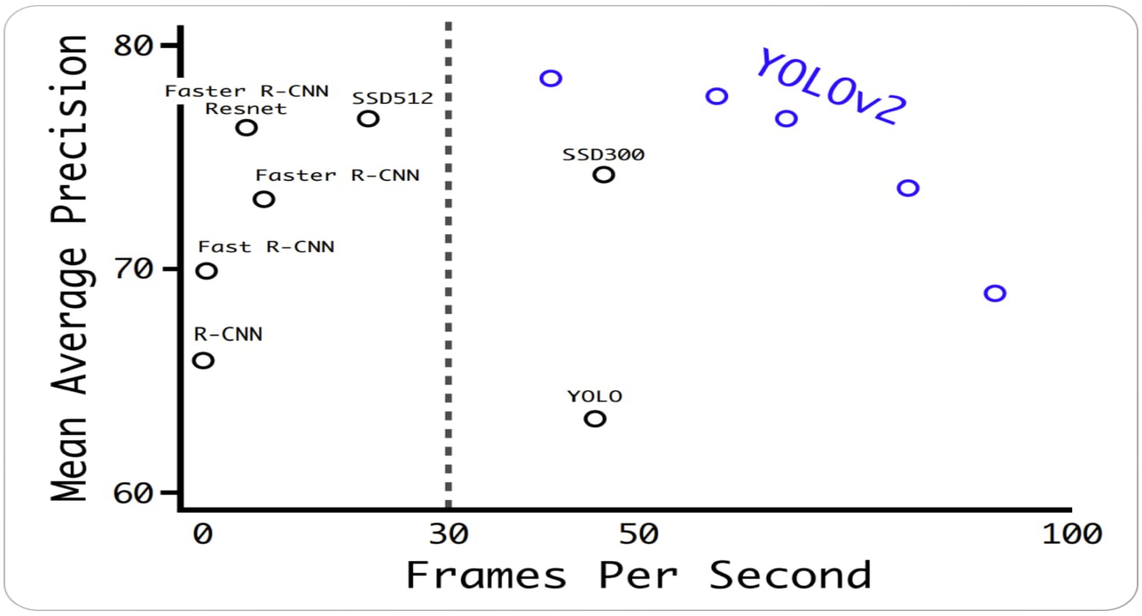
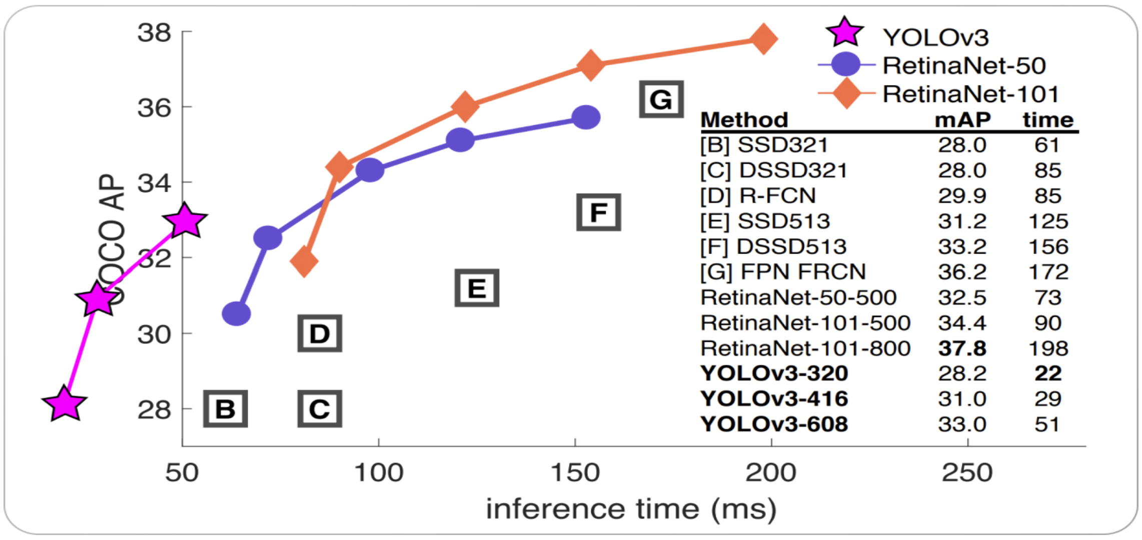


Figure 10: YOLOv2 and other models comparison

YOLOv3: YOLOv3 is an enhanced version of the YOLO object detection algorithm. It mainly increases accuracy and speed. YOLOv3 uses Darknet-53 CNN architecture with 53 convolutional layers, which is based on ResNet. This allows the model to achieve better results on object detection performance. In addition, it includes bounding boxes of different sizes and ratios, improving detection for objects of many sizes and shapes. It creates a Feature Pyramid Network(PFN), which allows the detection of many scales. As a result, the model receives boosted performance on small objects by getting multiscale representation. YOLOv3 can handle a broader range of object sizes with various aspect ratios which is a big improvement on the previous versions. In [12], shown in figure 11 we see the improvement comparison of yolov3 comparing other models

YOLOv4: YOLOv4, an advanced iteration of the YOLO object detection algorithm, introduced a new CNN architecture known as CSPNet. “Cross Stage Partial Network” (CSPnet) is the altered and better version of the ResNet architecture tailored for object detection despite its relatively shallow structure with more than 54 convolutional layers, CSPNet demonstrates exceptional performance on different object detection benchmarks. It serves as a significant improvement over YOLOv3.

YOLOv3 and YOLOv4 both models implement bounding boxes with varied scales and ratios to accurately measure the size and shape of detected objects. YOLOv4 proposes a novel technique called ”k-means clustering” to generate these bounding boxes. The approach involves applying an algorithm that is used to group the ground truth bounding boxes into clusters and the cluster centroids used as bounding boxes. This variation makes sure that the bounding boxes always align with the detected objects no matter what size and shape they are.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 11: YOLOv3 improvement comparison

Although YOLOv3 and YOLOv4 share a common loss function during model training, YOLOv4 introduces an additional component known as ”GHM loss.” This new term, based on the focal loss, aims to enhance the model’s performance when dealing with imbalanced datasets. By incorporating GHM loss, YOLOv4 addresses the challenge of training on datasets where certain object classes may be underrepresented or occur less frequently, resulting in improved detection accuracy.

YOLOv5: YOLOv5 is a model that builds upon its ancestors and is also open source. It introduces some new features and enhancements. Ultralytics actively maintains the model, by providing continuous updates and improving the algorithm.

Unlike the original YOLO, YOLOv5 uses a much more intricate architecture known as EfficientDet, based on the EfficientNet network architecture. YOLOv5 uses this architecture to achieve improved accuracy and enhance the ability to detect objects of many categories.

The novel approach of YOLOv5 is called ”dynamic anchor boxes”, which is employed to generate anchor boxes. Which is later used as a clustering algorithm to band those bounding boxes into many clusters and after that use the centroids of these clusters as bounding boxes.

Moreover, ”spatial pyramid pooling” (SPP) is applied to increase the detection capabilities for smaller objects. SPP incorporates a pooling layer which decreases the structure’s resolution of the feature map, allowing YOLOv5 to identify the object at different scopes. Although YOLOv4 also uses SPP, YOLOv5 implements different and advanced SPP to improve performance.

Although YOLOv5 and YOLOv4 both use comparable loss functions during model training, YOLOv5 has a novel component known as “CIoU loss”. This IoU loss function is constructed to mitigate the challenges that come from imbalance datasets, targeting to improve model performance.

YOLOv6: YOLOv6 is an upgrade to the earlier versions of object detection algorithms. It has many improvements, one of them being the implementation of EfficientNet-L2 CNN architecture. EfficientNet-L2 CNN architecture enhances efficiency and performance compared to the architecture used in YOLOv5. The model's framework is illustrated in the figure 12 in [13].

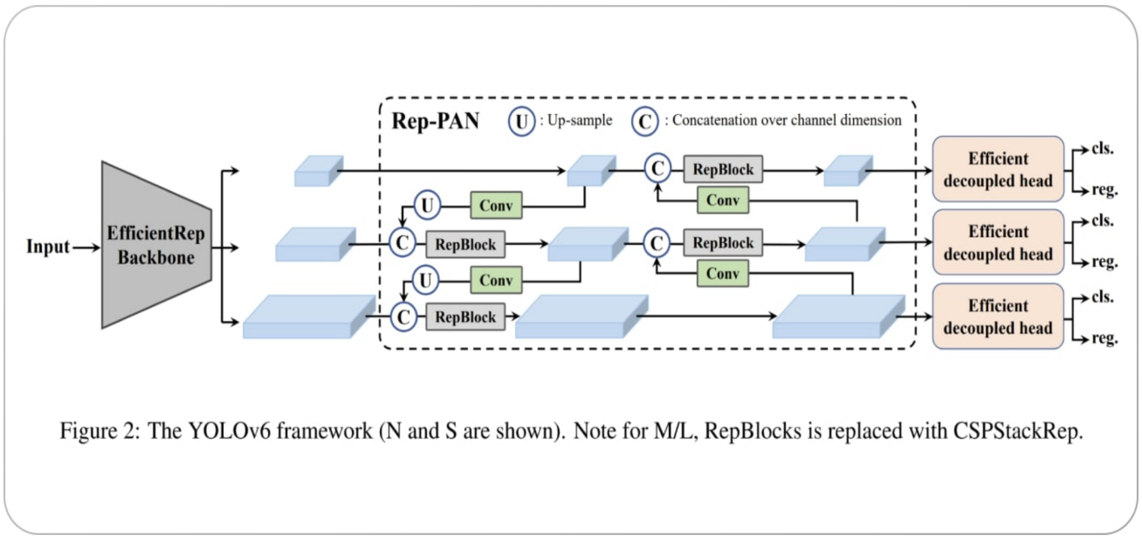
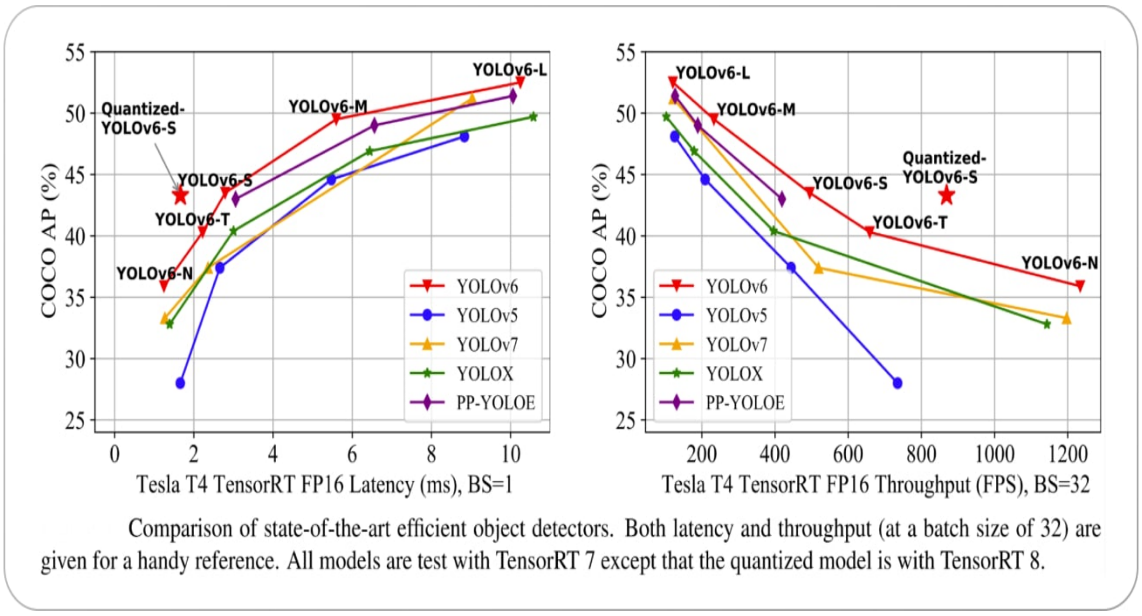


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

Figure 12: YOLOv6 framework

Anchor box generation sees a novel approach in YOLOv6 with the implementation of "dense anchor boxes".This method allows for a more precise alignment between the anchor boxes and the detected objects' size and shape.

In paper [13], the performance of YOLOv6 has been evaluated and compared to other cutting-edge methods, demonstrating its competitive results in figure 13. The comparative analysis highlights the effectiveness and advancements offered by YOLOv6 in the field of object detection.



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

Figure 13: YOLOv6 framework comparison

YOLOv7: YOLOv7 introduces several enhancements compared to its predecessors. Notably, it employs the use of bounding boxes, a set of predetermined boxes with different ratios. By incorporating nine anchor boxes, YOLOv7 achieves improved object detection capabilities, enabling it to effectively detect objects of diverse shapes and sizes. This advancement aids in minimizing false positives, enhancing the overall accuracy of the algorithm.

YOLOv7 introduces a significant enhancement through the utilization of a novel loss function called ”focal loss.” In contrast to the last YOLO versions that employed a standard cross-entropy loss, focal loss addresses the challenge of detecting small objects by assigning lower weights to well-classified examples and prioritizing challenging instances—objects that pose greater difficulty in detection. This targeted approach improves the model’s ability to accurately identify and localize objects, particularly those that are typically harder to detect.

Unlike its predecessors, YOLOv7 operates at a resolution of 608 by 608 pixels, surpassing the 416x416 resolution employed in YOLOv3. As a result, this increased resolution enables YOLOv7 to effectively identify smaller objects and achieve enhanced overall accuracy in object detection tasks.The figure 14 shows the YOLOv7 framework.

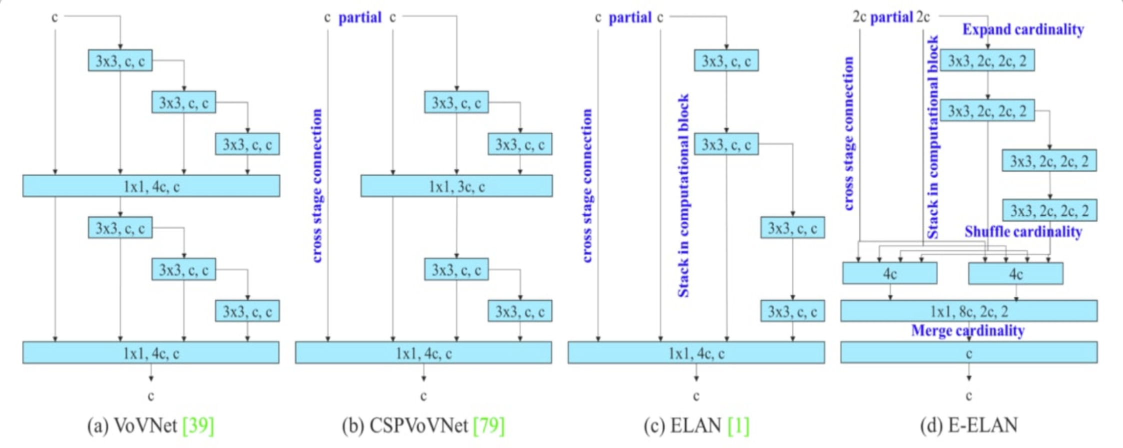


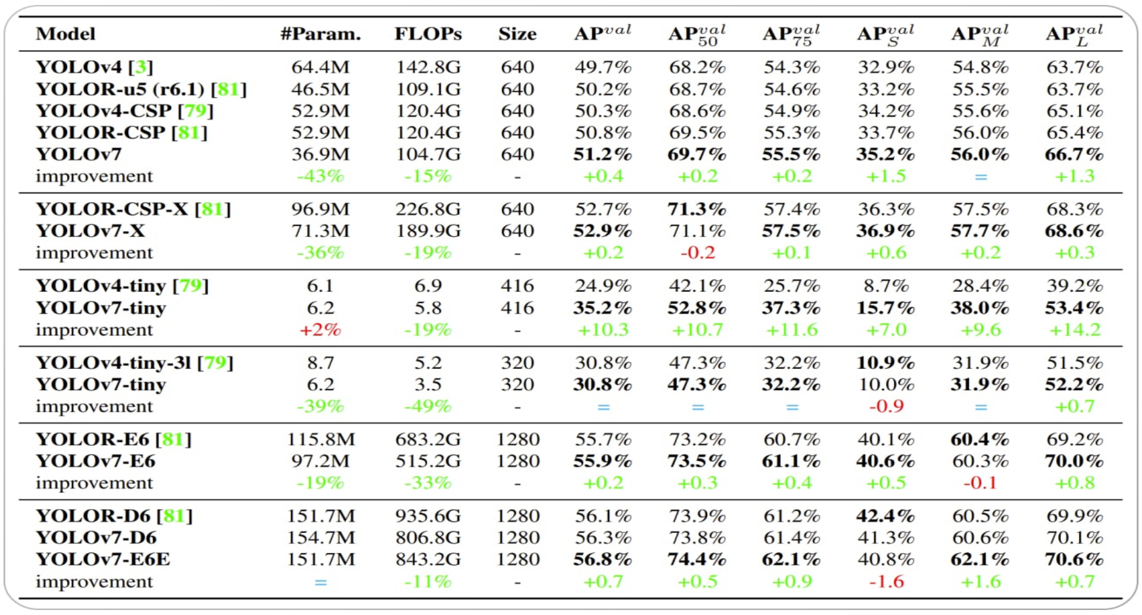
Figure 2: Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different feature maps and improve the use of parameters and calculations.

Figure 14: YOLOv7 framework[13]

One significant benefit of YOLOv7 is its remarkable speed. It exhibits impressive image processing capabilities, achieving a rate of 155 frames per second. This speed surpasses that of other cutting-edge object detection algorithms, including the original YOLO model, which could process a maximum of 45 frames per second. Such rapid processing makes YOLOv7 highly suitable for real-time applications that demand quick and responsive performance, such as surveillance systems and self-driving cars.

In paper [13], when it comes to accuracy, YOLOv7 demonstrates commendable performance in comparison in figure 15 to other object detection algorithms. On the widely-used COCO dataset, it achieves an average precision of 37.2% when evaluated at an IoU threshold of 0.5. This accuracy level is comparable to that of other leading state-of-the-art object detection algorithms. A quantitative comparison of its performance is depicted in the provided visual representation.

It is important to acknowledge that in terms of accuracy, YOLOv7 falls slightly behind two-stage detectors like Faster R-CNN and Mask R-CNN. These two-stage detectors often achieve higher average precision on the COCO dataset. However, it is worth noting that the trade-off for this increased accuracy is longer inference times, making YOLOv7 a more favorable choice for real-time applications where speed is a critical factor.



Model	**#Param.**	**FLOPs**	**Size**	**AP^{val}**	**AP$^{val}_{50}$**	**AP$^{val}_{75}$**	**AP$^{val}_{S}$**	**AP$^{val}_{M}$**	**AP$^{val}_{L}$**
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Figure 15: YOLOv7 comparative analysi

YOLOv8: The most recent addition to the YOLO network is YOLO-v8, released by Ultralytics. While a formal research paper release is still pending and further characteristics are waiting to be added to the YOLO-v8 system, early similarities indicate that it surpasses its predecessors and establishes itself as the new updated version of YOLO [14].

The provided figure 16, illustrates a comparison between YOLO-v8, YOLO-v5, and YOLO-v6 models that have been trained on 640 different images. It demonstrates that YOLO-v8 shows better throughput while maintaining the same figure of parameters, representing efficient hardware utilization and architectural enhancements. Ultralytics presents both YOLO-v8 and YOLO-v5, with YOLO-v5 showcasing excellent real-time performance. Based on the primary execution results from Ultralytics, it is proved that YOLO-v8 aims to prioritize high-speed inference on constrained edges.

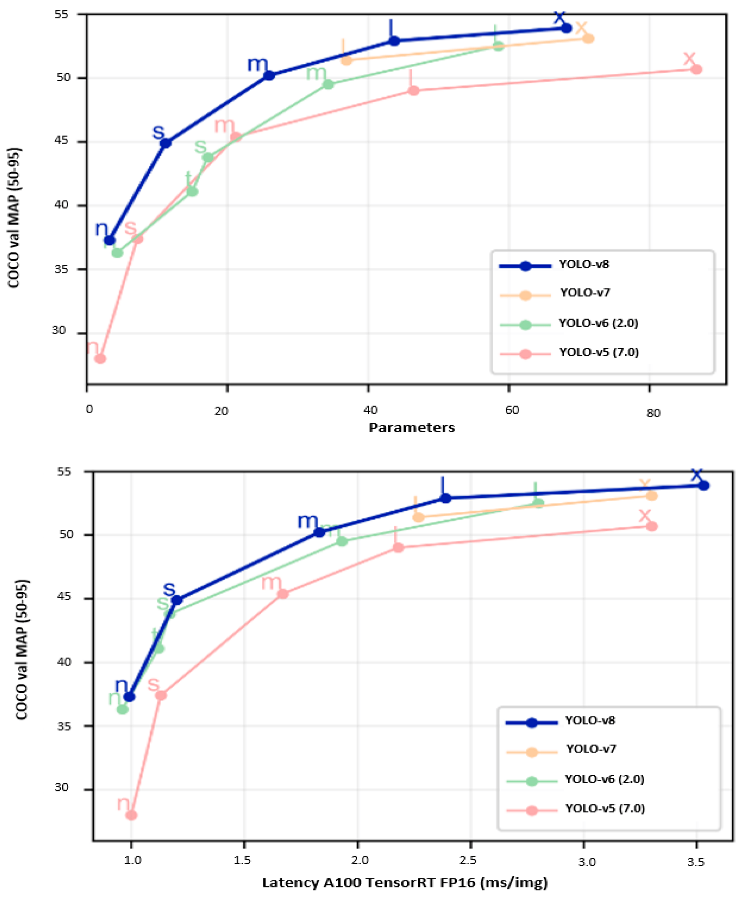


Figure 16: YOLOv8 comparison with predecessors

2.1.6 SSD

The Single Shot Detector (SSD) is a popular object detection algorithm used for efficient object detection in images and videos. It performs both object localization and classification simultaneously using a single pass of a convolutional neural network (CNN). Instead of relying on object proposals, SSD discretizes the output space into default bounding boxes with different aspect ratios and scales. It predicts the presence of objects and adjusts the bounding box coordinates to better match the object's shape. By combining predictions from multiple feature maps with different resolutions, SSD can handle objects of various sizes. Compared to methods that require proposal generation and resampling, SSD is simpler and eliminates the need for additional computation steps. It is easy to train, integrates well into detection systems, and achieves competitive accuracy without sacrificing speed. Experimental results on various datasets demonstrate the effectiveness and efficiency of SSD as a unified framework for object detection tasks.

2.2 Tracking algorithms

There are several tracking algorithms that are used based on different situations or different use cases based on the scenarios. These tracking algorithms might perform differently based on those particular situations. Here are some of the notable tracking algorithms:

2.2.1 SORT

A method for tracking multiple objects, Simple Online and Realtime Tracking (SORT) focuses on the point and systematic algorithms. To increase SORT's performance, appearance information is included in[15]. Their capability to follow different objects' overstretched times of obstacle is made conceivable by this alteration, which successfully brings down the recurrence of personality shifts.

This method combines frame-by-frame data using the Hungarian technique and Kalman filtering in picture space with an association metric that estimates the overlapping of the bounding box. With this straightforward method, good performance is achieved at rapid frame rates[15].

In[15], While attaining good tracking precision and accuracy overall, SORT shows a disproportionately large sum of identity shifts. This is because the employed association metric is only reliable in conditions of low state estimate uncertainty. As a result, SORT struggles to track around occlusions that frequently exist in front-view camera sequences.

2.2.2 DeepSORT

The SORT (Simple Online and Realtime Tracking) algorithm is combined with deep learning techniques to create DeepSORT. To track and re-identify objects, especially in busy environments, it employs the Kalman filter and a deep appearance descriptor model.

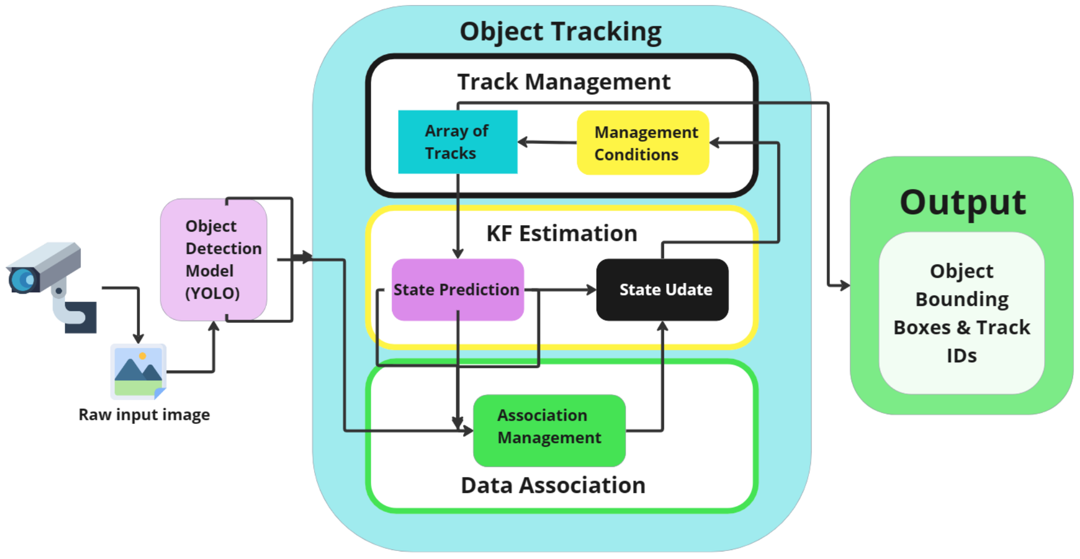


Figure 17: Overview of DeepSORT algorithm using Kalman filter

SORT struggles to track across occlusions, which frequently happen in front-view camera scenarios. To resolve this problem, a more knowledgeable measure that incorporates motion and appearance data should be used in place of the association metric. Specifically, a convolutional neural network (CNN) was used that has been trained on a sizable human re-identification dataset to distinguish pedestrians. By including this network, they improved the system's robustness against misses and occlusions while making it simple to use, effective, and relevant to online applications[15].

2.2.3 Kalman filter

Based on erroneous observations, the Kalman filter is a recursive mathematical process that determines the state of a dynamic system. The disciplines of computer vision, robotics, and control systems are just a few that use it extensively. When observations are imprecise and the system dynamics can be described by linear equations, the Kalman filter performs exceptionally well.

In the presence of Gaussian noise, and under the assumption of linearity and known model parameters, the Kalman filter offers an optimum approximation of the state. The mean square error between the estimated state and the actual condition is minimized. However, it might not function effectively in situations when the system dynamics are very non-linear or non-Gaussian.

To accommodate non-linear system dynamics, extensions to the Kalman filter, such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), approximate the dynamics using linearization or sampling methods.

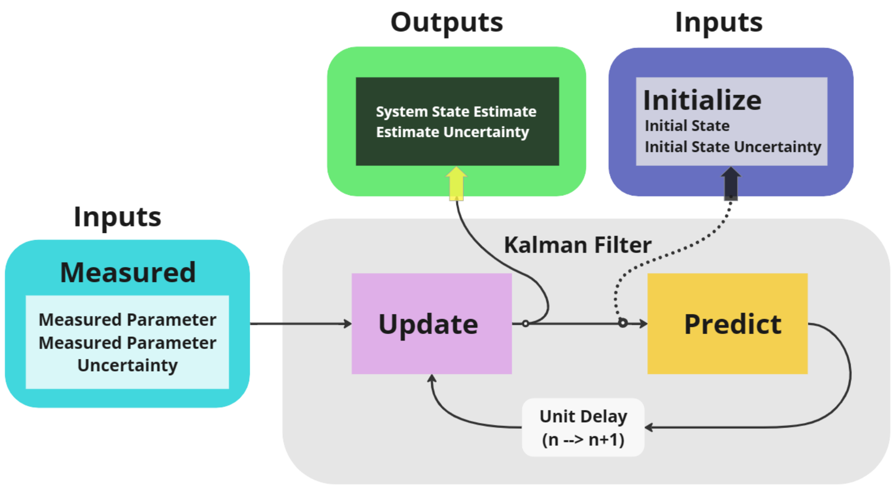


Figure 18: Kalman filter

2.2.4 Mean shift

Mean Shift is a well-liked non-parametric clustering approach in computer vision for applications like object tracking and picture segmentation. It is a mode-seeking method that locates clusters or areas of interest by repeatedly moving a window or kernel toward the mode (peak) of the data distribution.

The probability density function's gradients are used by gradient-based feature space analysis techniques to locate the maxima. Such techniques need an estimation of the likelihood of density, which among other reasons makes them complicated. The kernel is then moved by a predetermined length vector in the direction of the largest increase in density in the gradient-based techniques. The step size, or magnitude, must be properly selected. The challenge is figuring out how to select an appropriate step size because a tiny step size may hinder convergence.

The primary issue with gradient techniques is resolved by the mean shift algorithm. The underlying part behind the mean shift is to find out the points in the D-dimensional feature space as an empirical probability density function, with dense patches standing in for the local maxima of the underlying distribution. The local density estimate is gradient ascension in the feature space till convergence. Following the approach, stationary points match the distribution's modes, and the same stationary points are addressed as belonging to the same cluster. The adaptive step size of the mean shift is determined by the gradient of the probability density[16].

2.2.5 Hungarian algorithm

The Hungarian algorithm is a method that solves the association problem in tracking. The deep sort algorithm heavily depends on two crucial algorithms namely the Kalman filter and the Hungarian algorithm. Hungarian algorithm helps us to identify whether an object is the same as the previous frame.

2.2.6 Particle filter

Particle filter methods use a collection of particles to describe an object's state. They are suitable for tracking in complex circumstances because they can manage non-linear and non-Gaussian system dynamics.

The Particle Filter is a potent method used for state estimation and tracking in many disciplines, including computer vision. It is also known as Sequential Monte Carlo (SMC) or Sequential Importance Sampling (SIS).

The Particle Filter functions by using a collection of particles or samples to represent the state of a system. Each particle represents a theory about the state of the system and is paired with a weight that indicates how likely it is to be accurate. The Particle Filter, also known as Sequential Monte Carlo (SMC) or Sequential Importance Sampling (SIS), is a potent method used in many domains, including computer vision, for state estimation and tracking. A set of particles or samples are used by the Particle Filter to represent the state of a system. With a weight that reflects its likelihood of being accurate, each particle represents a theory about the system state.

Many other tracking algorithms sometimes are merged based on particular projects to achieve outstanding results based on the environment or the requirements of the project. Either way, all tracking algorithms have their strong and weak sides. Therefore based on projects they may vary.

2.3 Overview of a social distance monitoring framework

To evaluate real-time video or image data and gauge the distance between people in a scene, computer vision systems that measure and track social distances use deep learning models like YOLO, RCNN, and SSD. An overview of how such a system usually functions is given below:

Data collection: Using a camera or a dataset, the system first gathers video or still photos of the target area. The dataset may contain noise and varied angles.

Object detection and tracking: To locate and follow people in the video or picture frames, the system uses object detection techniques. To accurately detect and identify persons in a scene, methods like convolutional neural networks (CNNs) or deep learning models can be applied.

Correction for perspective: The system makes adjustments for perspective distortion brought on by the placement and angle of the camera. This guarantees precise distance measurements while taking into account the spatial organization of the scene and the camera's field of view.

Distance calculation: The system determines the separation between pairs of people in the scene using the corrected perspective. Depending on the individual implementation, it may employ depth data from the camera or geometric calculationstions.

Distance thresholding: The system compares the determined distances to a the predetermined threshold value that represents the advised social separation distance. People who stay inside the threshold are thought to be keeping a healthy social distance, but people who go past the threshold are identified as going against the distance rules.

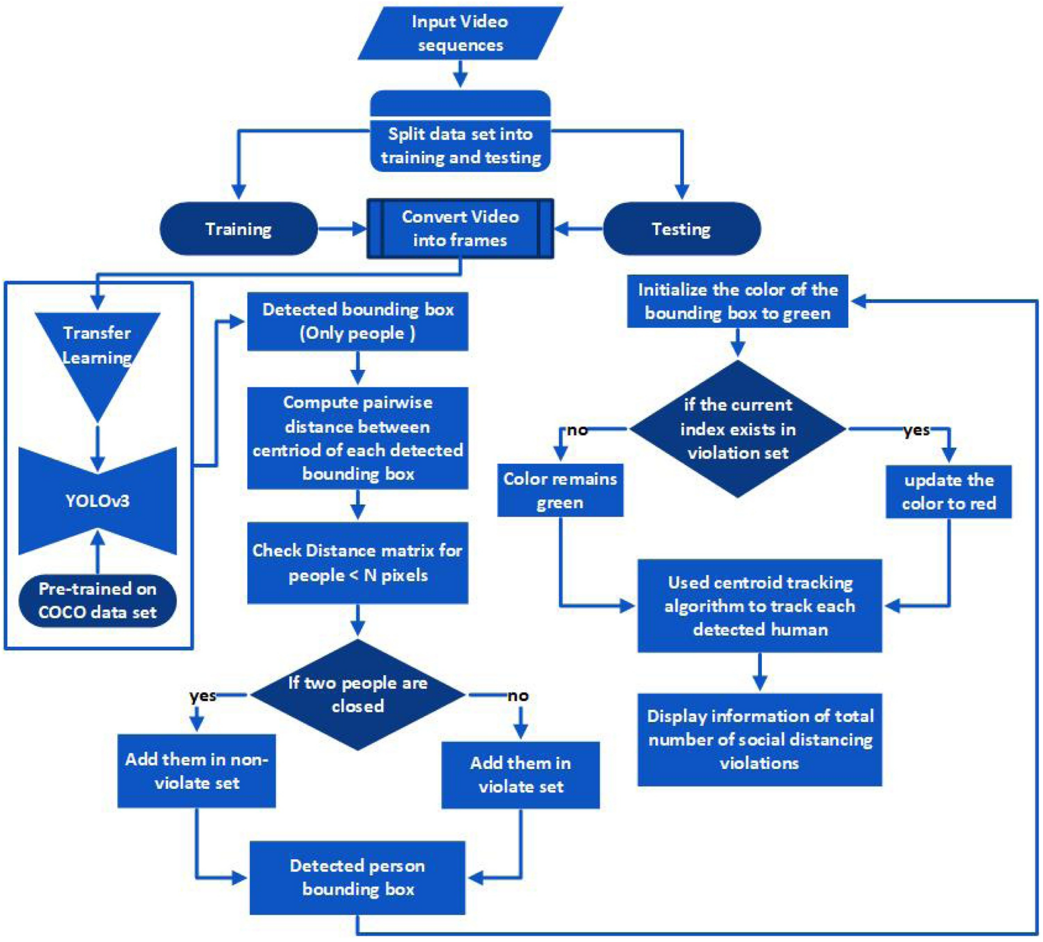
Monitoring and reporting: The system can continually observe the scene while instantly updating the distance readings. It can provide reports, give statistical data, or visualize social distance compliance over time, allowing authors- ties to spot trends and take the appropriate action.

Optimization and improvement: Depending on the unique environment and needs, the system can be improved and optimized. The accuracy and dependability of the distance measurements can be improved by using methods like background subtraction, noise reduction, and adaptive thresholding.

It's crucial to remember that, even though computer vision-based social distance measuring systems can be quite helpful, they are not infallible and must be used in conjunction with other safety precautions and recommendations made by health authorities.

In[17], Transfer learning is used to increase the effectiveness of the detection model for people in the above views, and the present architecture is expanded with a new layer of overhead training.

Once the detection is done, using bounding box information, the centroid distance is calculated. Using the Euclidean distance, the distance was identified. After the calculation of centroid distance, a predetermined threshold is used to find out whether or not the distance between any two bounding box centroids is smaller than the specified number of pixels. If two persons are near one another yet their distance value is greater than the required minimum social distance, the bounding box's color is updated to red and the bounding box's information is recorded in a violation set. The adoption of a centroid tracking algorithm enables the surveillance of individuals who transgress or go above the social distance threshold. The model output shows the overall number of social distance violations as well as persons bounding boxes and centroids that were found.



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

Figure 19: Overview of a social distance monitoring system[17]

A social distance monitoring system may also contain a log system to store data on violations and an alert system can be built around it. Mobile app development can send alert notifications to warn about the possible situation of the social distancing of nearby cities as in done in[18], MySD can provide the general people with a clever way to monitor and remind them to keep their distance when in public settings. To reduce the risk of contracting COVID-19 in busy or public places, MySD enables the development of an invisible safe zone around the users. The user will be more aware of abiding by social distance in the high-danger locations (i.e., Red and Yellow zones) by including the current zone information. The notice and vibration alerts will further assist the user in forcing themselves to keep a safe distance.

Chapter 3

Related Works

3.1 Literature review

In recent years, person identification and re-identification have been significant research areas in computer vision. Two research papers have addressed these problems using different techniques and datasets.

This paper [19] proposed a comparison between both traditional and deep learning-based methods for analyzing crowded environments. The methods can be divided into two categories: (1) crowd counting and (2) crowd activity detection. Besides examining the available datasets for crowd scenes, this research also presents Crowd Divergence (CD), a novel performance metric for crowd scene analysis methods. This metric predicts the difference between the calculated and actual crowd counts in crowded scene recordings. The survey shows that deep learning-based methods outperform traditional methods, especially CNN and GAN(Generative Adversarial Network) frameworks.

The paper [20] proposed a framework that combines the spatial and temporal data to provide visual aids for crowd monitoring along with a series of crowd mobility graphs (CMGraphs) to store space-time patterns from both of the actual CCTV footage and the geographical data of the public space. This framework uses deep learning-based computer vision models (YOLOv7 and Faster R-CNN) to automate person detection in CCTV video and uses geometric transformations and Kalman filter-based tracking algorithms (DeepSORT) to track individuals in video frames. Moreover, the authors designed a GCN-GRU model that shows CMGraphs make it easier to predict overcrowding at important exit/entry zones.

The paper [21] proposed an improved version of the IDLA architecture for identifying actors and actresses in movies. By learning the variations and similarities between pictures, the model outperformed the IDLA method significantly on the CUHK03 dataset and the authors' dataset.

The second paper [22] aimed to build a robust and continuously updated multi-shot exhibition of noticed reference personalities intermittently on the web using L2-standard descriptor matching and the Disconnection Timberland algorithm. Both studies provide insights into the potential applications of computer vision in person identification and re-identification, highlighting the ongoing efforts in this field.

Person re-identification (re-ID) across non-overlapping cameras is a challenging problem in computer vision, and two recent research papers have proposed different methods to address it. The first study [23] uses a multi-camera observation framework and matching of interest-point descriptors collected from brief video sequences to capture appearance variability. This approach showed promising results in a test assessment on low-resolution videos in a commercial mall, achieving an accuracy of 82% for a review of 78% in a fast and computationally efficient manner.

Both papers address the challenge of person re-identification but from different perspectives. While [24] focuses on improving the discrimination ability of the model by considering the person ratio and relationships, [25] proposes a novel approach for on-board re-ID using UAVs. Both approaches show promising results and contribute to the development of effective person-ID techniques. Person reidentification across different modalities and face detection from video clips are two important challenges in computer vision. In the paper [26], a relative networking design is proposed for person reidentification from both thermal and visible images. The approach involves combining two pooling processes, GAP and GMP, to capture both background information and detailed features. The proposed method is tested on two datasets, SYSU MM01 and RegDB, and it shows better results compared to traditional methods, ensuring better recognition accuracy. In the paper, the authors propose a deep learning-based approach for face detection from video clips using a convolutional neural network (CNN). They select images of faces to train the model for face detection and evaluate their approach on two datasets for familiar and unfamiliar environments. The proposed method shows better results than the Viola-Jones algorithm for face detection and provides higher accuracy in person identification compared to feed-forward neural networks.

Sometimes the re-identification process becomes a challenging task as different visual appearance variations take place. It can be caused by various backgrounds, illumination variations in poses, etc. To solve these kinds of challenges, the authors of paper [27], presented a method called ROIF which can make person re-identification more accurate in less periods by combining textural and chromatic features. In this research, an efficient way for the ROIF method was shown and the implementation was done by HSV histogram and chromatic content. Person re-identification involves identifying individuals across different camera views, but various challenges can arise due to differences in visual appearance. For instance, surveillance cameras can malfunction, resulting in grayscale videos instead of true-color videos. In such cases, the process is known as CGVPR, and the paper [2] proposes a new approach called SDPL to re-identify persons between grayscale and true-color mode videos. Furthermore, identifying individuals based on their clothing attributes can be more effective than using facial features. In paper [28], the authors present a method that focuses on clothing attributes by extracting features using HSV histograms and Histograms of Oriented Gradients (HOG) from raw footage. The verification process is then completed using the LSVM framework. In recent research studies, several methods have been proposed to improve person re-identification. One such method proposed in paper [29], is simultaneous detection and re-identification, which aims to identify a person in real time using clothing and other features. This approach showed better results than traditional methods and was tested on a public dataset named PRW.

Another approach proposed in research [30] is the distributed network person re-identification framework, which can measure performance between nodes of the network using the camera matching cost. This framework also allows for learning the network topology by deriving the distance vector, thus reducing the number of cameras required for inquiry. Lastly, in research [3], the authors propose using body symmetry and part-locality-guided DNDFE to enhance deep learning. By pooling body symmetry and using local normalized layers, deep learning can be improved, overcoming the drawback of small available databases for creating deep models.

3.2 Comparative analysis and discussion

In this comparative analysis, we have examined five research papers that propose different approaches for monitoring social distancing and detecting human presence using various object detection models.

Mahadi et al.[31] present an approach to monitoring social distancing and infection risk during the COVID-19 pandemic using YOLOv4. A hybrid Computer Vision and YOLOv4-based Deep Neural Network (DNN) model was created to automatically detect people in crowded settings, both indoors and outdoors, utilizing standard CCTV security cameras. In this paper, the proposed DNN model is integrated with a customized inverse perspective mapping (IPM) technique and the SORT tracking algorithm, which leads to reliable people detection and social distancing monitoring. This proposed method processed 7530 frames, giving 99.8% accuracy and the speed or fps as 24.1.

Narinder et al.[32] presents a deep learning-based framework designed to automate the process of monitoring social distancing through surveillance videos. The framework employs the YOLO v3 object detection model to distinguish humans from the background and utilizes the Deepsort approach to track the identified individuals using bounding boxes and assigned IDs. They achieved 23 fps by processing 7560 images with maP 84.6% and the total time it took was 5659 seconds.

Aquib et al.[33], their research paper presents the development of an algorithm using object detection techniques. Specifically, a CNN-based object detector is investigated to identify the presence of humans. The output of the object detector is then utilized to compute the distances between each pair of detected humans. This novel approach to the social distancing algorithm identifies individuals who come within a certain permissible distance by marking them with red indicators. The experiment was conducted on a 64-bit system with an Intel Core i3-5005 CPU@2.00 GHz processor, and Python was utilized in Google Colab. The INRIA image dataset, consisting of 6562 images, was employed for training. Remarkably, the mean average precision (map) score was an impressive 97.75%, achieved in 2169.9 seconds. However, the paper does not provide specific information regarding the frames per second (fps) during the experiment.

Table I: Comparative analysis based on fps, maP/accuracy, Total frames, detection algorithm

Study	**Fps**	**mAP**	**Total frames**	**Detection Algorithm**
Mahadi et al. [31]	24.1	0.998	7530	Yolov4
Narinder et al. [32]	23	0.846	7560	Yolov3
Aquib et al. [33]	No info	0.9775	6562	CNN
Inderpreet et al. [34]	32	0.98	3846	Yolov5
Jingchen et al. [35]	8.06 & 7.03	0.8844	9963	SSD300

Inderpreet et al.[34] presents a real-time surveillance system designed to analyze video feeds and determine if individuals detected in the footage are wearing masks. Additionally, the research focuses on monitoring social distancing compliance. The proposed approach involves utilizing YOLOv5 to detect humans in the CCTV frames. Subsequently, the detected faces undergo classification using Stacked ResNet-50 to ascertain whether a person is wearing a mask. Meanwhile, DBSCAN is employed to identify proximities among the detected individuals. For processing the CCTV feeds for mask classification, they trained their model using low-resolution facial portraits from various sources, including the Real-World Masked Face Dataset (RMFD), which contains a total of 3846 images. The training results showed a high accuracy of 96% and a low training loss of 12% when trained on the dataset that included both masked and unmasked images. For testing, they used 528 images and achieved a testing accuracy of 84% and a testing loss of 14%. The YOLOv5 model used in the study demonstrated good precision and recall of 0.6 and 0.98, respectively, with a mean average precision (mAP) of 0.98. The system was able to achieve an impressive frame rate of 32 frames per second (fps) on the input video, allowing for real-time analysis of face masks and social distancing of detected humans. This high processing speed makes the system suitable for real-time surveillance and monitoring applications.

Jingchen et al.[35] presents a novel approach for real-time social distancing monitoring using SSD object detection technology. The method leverages the SSD300 model to detect individuals in both videos and images. When the detected distances between people fall below a predefined threshold, a warning Red Line is labeled on these individuals, effectively implementing real-time monitoring of social distancing compliance. In this research, the authors utilized the PASCAL VOC dataset, comprising 9963 labeled images, for their experiments. The experiments were conducted on hardware with an Intel i5-8400 CPU, 8GB RAM, and an NVIDIA GTX 1060 3G graphics card with TensorFlow-gpu version 1.13.2. Their trained model achieved an mAP of 88.44%. The authors observed that the frames per second (fps) varied depending on the number of people detected. When four people were detected, the fps was measured at 7.03, and for two people, it reached 8.06 fps.

All the proposed methods utilized various object detection models (YOLOv4, YOLOv3, YOLOv5, CNN, and SSD300) for different applications such as crowd monitoring, mask detection, and social distancing compliance. Mahadi et al. [31] achieved the highest accuracy of 99.8% with a processing speed of 24.1 fps, making it suitable for real-time crowd monitoring. Inderpreet et al.[34] focused on real-time surveillance for mask detection and social distancing, achieving an impressive 32 fps and a high mAP of 0.98. Jingchen et al.[35] proposed a real-time social distancing monitoring system using SSD300, with an mAP of 88.44% and varying fps based on the number of people detected. However, Aquib et al. did not mention the fps, which is a crucial aspect in real-time applications.

Chapter 4

Custom Dataset

4.1 Custom Dataset

For this research, we have made a couple of custom datasets with custom annotations frame by frame.

In this research, a custom dataset was created and utilized for training the YOLOv7 and YOLOv8 models to tackle the task of social distancing detection. The dataset was divided into three sets: the training set, the validation set, and the testing set. The first two of our dataset was trained to detect persons at 20 fps whereas only one dataset Social Distancing version 1 was trained with a resolution of 1000x640. The rest of the dataset was trained with 25fps and 30fps with resolutions of 640 x 640 and 1080x640. The total number of images we trained is 1994.

Each image in the dataset has been meticulously annotated, with a single class, *Person*, indicating the presence of individuals in the images. The dataset we made so that we can solve the perspective change problem and the false detection problem can be reduced. We did this so that we could get our new custom weight so that those problems could be solved. Although the pre-trained dataset weight was capable of detecting person class in any situation whenever the angle changed or in some situation class detection were not accurate as it would detect mannequin-like objects or other objects that may look like a person but are not persons.

Even though we couldn't compete with the large datasets of Yolo models like coco and other large object detection datasets. but for our particular project, we were able to solve the perspective correction problem by making our custom dataset. And for the record perspective correction has been a huge challenge in computer vision for decades. The details of the custom dataset are as follows:

Table II: Custom dataset preprocessing data table

Video	**Training set (img)**	**Validation set (img)**	**Testing set (img)**	**Training(fps)**	**Resolution**	**Total (img)**
video 1	573	55	27	20	1000x640	655
video 2	184	52	25	20	640x640	261
video 3	277	79	41	25	640x640	397
video 4	275	78	40	25	640x640	393
video 5	202	57	29	25	640x640	288
video 6	670	78	67	25	1080x640	785
video 7	480	55	56	30	1080x640	591
video 8	501	45	50	30	1080x640	596
video 9	268	33	34	30	1080x640	335

4.2 Data processing and training

All the above datasets are for particular videos with different challenging angles. After we made all the datasets for those videos, we exported the zip files from Roboflow and merged them. We took all the images from the image folder. Likewise same for the training, validation, and testing folder, and a single folder where all the images that were pre-processed were stored.

After merging all the images, we trained them in YOLOv7 and YOLOv8 and got our expected ”best. pt” weight for both our YOLOv7 and YOLOv8-based overall system. Though we couldn’t compete with datasets like COCO, we were able to achieve suitable data to perform well for our system and solve the perspective correction problem.

4.3 Model Implementation

We trained our models on YOLOv7 and YOLOv8 as our proposed system is based on these models. the trained results were very good and YOLOv8 had better accuracy and overall performance compared to YOLOv7 but the results are almost the same to the point where it’s negligible.

Chapter 5

Methodology

5.1 Our implemented Detection algorithm

To implement our system we choose Faster R-CNN, YOLOv7 and YOLOv8. Here we implemented both YOLOv7 and YOLOv8 models where even though YOLOv8 performs slightly better but it's a relatively new model where we had so many problems implementing the model on our system. Even though we were able to solve those problems we still chose to implement our model in a lower version of yolo model, YOLOv7. YOLOv7 may be preferred over YOLOv8 for person identification for the following reasons:

YOLOv7 is more resistant to occlusion and distortion than YOLOv8, making it more suitable for identifying people in difficult circumstances. Besides, YOLOv7 is easier to train and deploy on devices with limited resources since it has fewer parameters than YOLOv8.

YOLOv8 is a newer and more sophisticated version of YOLOv7 for object identification and picture segmentation, according to the online sources we identified. It was made by Ultralytics, the company that also made YOLOv51. YOLOv8 claims to be faster and more accurate than all prior YOLO models, and to require less hardware and training time. TensorFlow Lite exporting is not currently supported by YOLOv8, which is still in development.

On the other hand, the cutting-edge object identification model YOLOv7 was unveiled in 2022. It utilizes convolutional neural networks (CNNs) to carry out image identification tasks and is based on the original YOLO architecture4. Without any pre-learned weights, YOLOv7 may be trained on tiny datasets and translated to TensorFlow Lite format for edge deployment. YOLOv7 has been successfully employed in a variety of applications, including autonomous driving, facial identification, lost and found, and pedestrian detection.

Therefore, for person identification, we may use either YOLOv7 or YOLOv8, depending on our use case and needs. However, until YOLOv8 supports TensorFlow Lite export, we might want to continue with YOLOv7 if we require a model that can operate on mobile or edge devices. We would use YOLOv8 or its instance segmentation variant7 if we wanted a model that is capable of both object recognition and picture segmentation. An overview of the system architecture of both YOLOv7 and YOLOv8 were described earlier.

5.2 Our implemented tracking algorithm

For this research we used DeepSORT tracking algorithm. DeepSORT enabled us to track persons accurately by generating a unique id to each person and using Kalman filter it estimates a new accurate state for each person, by that we were able to re-identify the person using this tracking algorithm. We made a system using both YOLOv7 & YOLOv8 and DeepSort and by that we were able to make an overall social distance monitoring framework.

There are lots of other tracking algorithms like mean shift, particle filter and also many other tracking algorithms. But as Kalman filter based DeepSORT algorithms state estimation is better than other algorithms its very much useful for person re-identification, which is a core part of our research.

5.3 Our Proposed System

Here in our research we applied two approaches where we used Faster R-CNN and YOLO models. And we also performed our tracking where one tracking is done using SORT tracking algorithm. and another approach where we used DeepSORT using Kalman filter for tracking.

5.3.1 Faster R-CNN/YOLOv8 & tracker(without DeepSORT) based system

Our idea involves using a monocular camera to identify people within a specific area. If the system detects a violation of social distancing rules, it will mark the people who are violating the rules and show their distance using a red line with the distance value.

Using either Faster R-CNN or YOLOv8 we detect the persons in the frames. After that using the tracking script that we wrote, we draw a bounding box. After the coordinates of the bounding box are placed then we move to the next method where we calculate the centroid to calculate the distance. After we compare our centerpoint distance to the threshold distance we detect whether one is violating the distance or not. The distance is highlighted whether its maintaining the social distance or not.

The overall system can work but not without human intervention. The system generates total violation based on the threshold that we pre-determined. This threshold distance can be customizable or optimized based on the systems use-cases. The figure bellow provides an overview of our method.

Determine bounding box and bottom center points: First, the method detects pedestrians present in the image domain by employing a deep CNN model

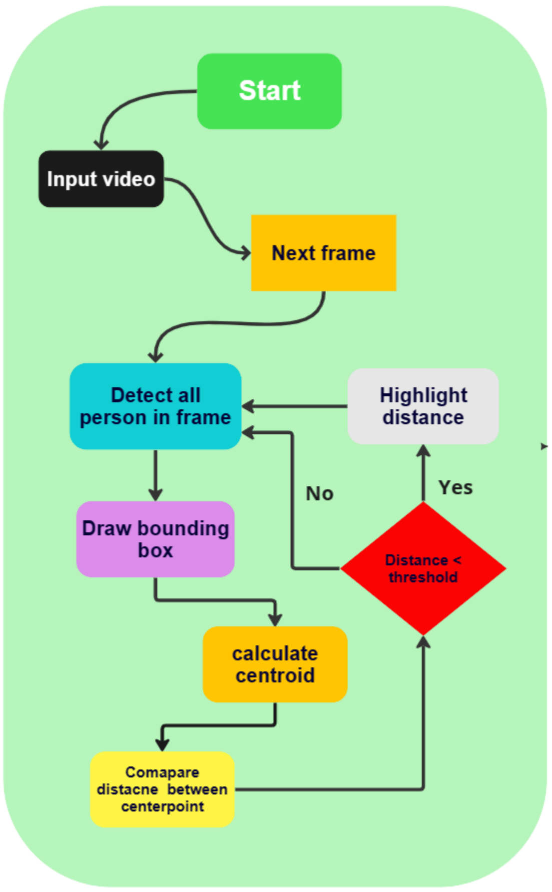


Figure 20: Faster R-CNN/YOLOv8 & tracker(without DeepSORT)based system

that has been trained on a dataset from real-world scenarios. Here we used Faster R-CNN and various models of YoloV8. The CNN maps an image to n tuples, where Each tuple represents a detected object in the image. It includes the object label, bounding box (given by pixel indices), and detection score. We only take the detection box if it is of a person. Then to find the bottom center point of the bounding box we can use the formula:

$$C(x, y) = \frac{(x + w)}{2}, (y + h) \tag{5.1}$$

Here (cx,cy) are the coordinates of the bottom centerpoints of the bounding box, (x,y) is the coordinate of the top left corner of the bounding box, w is the width and h is the height.

Social Distance detection: After we get the center points, we can use them to determine the distance between people. The formula for this is,

$$Distance = \sqrt{(c_x i - c_x j)^2 + (c_y i - c_y j)^2} \tag{5.2}$$

Here i and j indicate different centrepoints. We can compare all detected persons using this formula to determine if they are violating social distancing rules. If the distance is smaller than threshold level then they are breaking the social distancing rules. We can highlight the distance.

5.3.2 YOLOv7/YOLOv8 & DeepSORT based system

Our system contains a structure of a module for distance estimation and crowd monitoring which can do both seamlessly. We used both YOLOv7 and YOLOv8 for various reasons regarding mainly the online support and the deployability of projects containing these two models. Therefore, our proposed system offers tracking and detection of persons using a merge of two algorithms namely DeepSORT and YOLOv7/YOLOv8 model.

Training: We used stock videos that are available on the internet as our input video. To prepare the data for training, we utilized Roboflow tool to accurately annotate the objects of interest in the video. The annotations involved creating bounding boxes around the object which is in our case person to provide the necessary ground truth for training the object detection model.

After the annotations, the annotated videos were converted into individual frames to create the dataset for training. Then the dataset was splitted into three section training, testing and validation sets. Now for the actual training, we used the YOLOv7 and YOLOv8 model which gave us custom weight for each dataset.

Inference: The custom weights were then used in our own YOLOv7 and YOLOv8 models. Since our model runs in real time, after giving input video, it detects our trained object(person) and creates a bounding box around the object. At the same time, the model also assigns a unique id on top of each bounding box using DeepSORT algorithm and generates centroid or center point inside the bounding box. Using the centroid, the model calculates distance between each person depending on our given distance threshold.

Violation detection: To calculate whether people are disobeying the distance maintaining rule or not, we used two distance thresholds in our model, Violation and Compliance. In the first decision set, if the calculated distance is greater than violation threshold, it will check the second decision set which is if the calculated distance is greater than compliance threshold, it will keep the detected people out of threshold distance, meaning people are far away than the safe distance threshold.

If the calculated distance is less than compliance threshold it will show a green line between each centroid meaning that they are following the safe distance rule. If we do not do this separating, the model then creates both red and green lines based on the threshold distance between all the centroids which creates a clustered output video. Now in the first decision set, if the distance is less than the violation threshold, the model will initialize the red threshold line between the centroids.

Displaying Results: During the processes we mentioned above, when our model runs in real time, it also shows how many people detected, number of violations and compliances. After estimating the distance and tracking all persons, the python window pops up where we show the total number of objects, compliance and violation all frame by frame in real time.

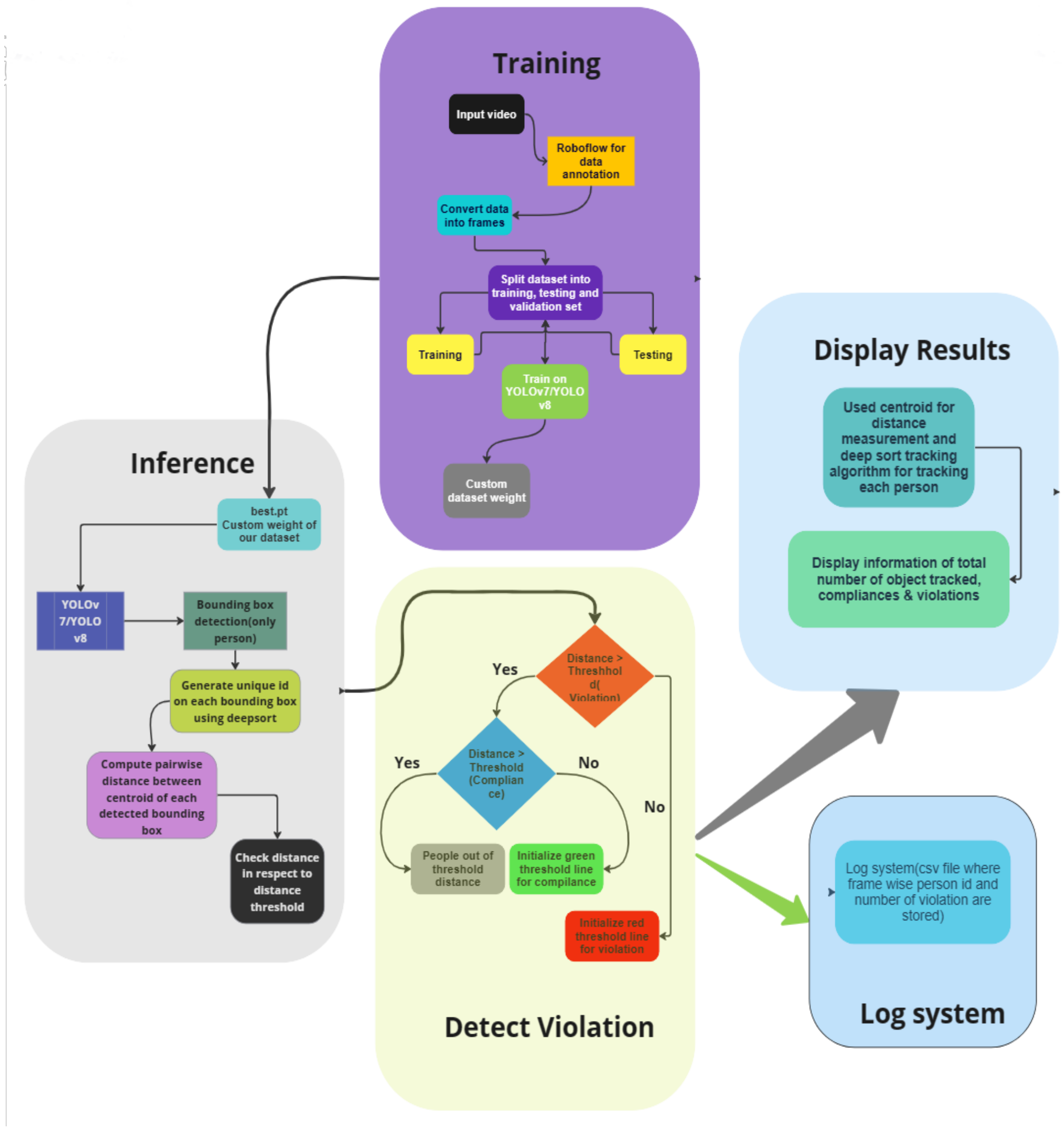

Training
Input video
Roboflow for data annotation
Convert data into frames
Split dataset into training, testing and validation set
Training
Testing
Train on YOLOv7/YOLO v8
Custom dataset weight
Inference
best.pt Custom weight of our dataset
YOLOv 7/YOLO v8
Bounding box detection(only person)
Generate unique id on each bounding box using deepsort
Compute pairwise distance between centroid of each detected bounding box
Check distance in respect to distance threshold
Display Results
Used centroid for distance measurement and deep sort tracking algorithm for tracking each person
Display information of total number of object tracked, compliances & violations
Detect Violation
Distance > Threshhold(Violation)
Yes
Distance > Threshold (Compliance)
Yes
No
No
People out of threshold distance
Initialize green threshold line for compliance
Initialize red threshold line for violation
Log system(csv file where frame wise person id and number of violation are stored)
Log system

Figure 21: YOLOv7/YOLOv8 & DeepSORT based system

Log System: At the end of the executions, a log file(.csv) is created where frame by frame id of persons and number of violations are stored. This log system can be later used to make alert systems or any other notification system. As we have a log system, therefore no human intervention is necessary to monitor the system.

To find the center of the bounding box, we use the centroid formula:

$$C(x, y) = (\frac{(x_1 + x_2)}{2}, \frac{(y_1 + y_2)}{2}) \tag{5.3}$$

C(x, y) is the center point with coordinates (x, y), while x1, y1, x2, and y2 are the x and y coordinates of the rectangle's top-left and bottom-right corners, respectively.

We can simply find the centroid, which represents the geometric center of the rectangle, by taking the average of the x and y coordinates of these two corners. It's similar to locating the midpoint that equally balances the rectangle in both the x and y directions.

Similarly, when we need to calculate the distance between two points in 2D space, we use the distance formula:

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5.4}$$

The Pythagorean theorem is used to develop this formula, which calculates the straight-line distance between two locations given their x and y coordinates. It is a fundamental computation that allows us to determine the distance between two objects.

Kalman filter state estimation: Kalman filter predicts the next state where the object might be in the next frame. the overview of the Kalman filter is as follows:

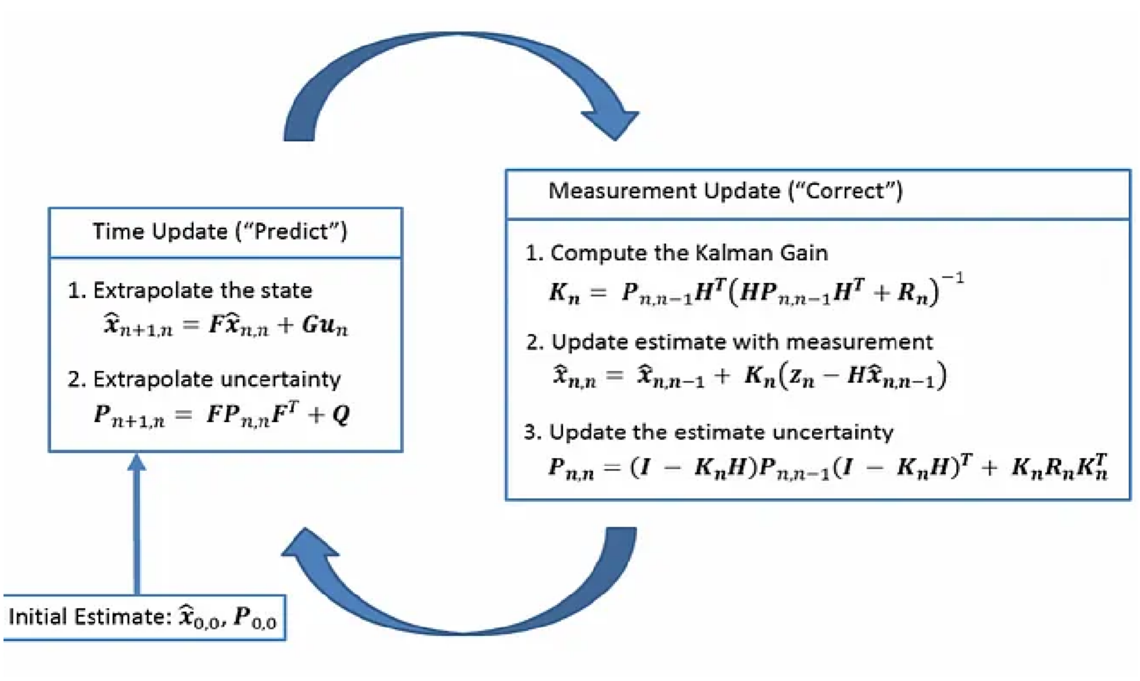

Measurement Update ("Correct")
1. Compute the Kalman Gain
$K_n = P_{n,n-1}H^T(HP_{n,n-1}H^T + R_n)^{-1}$
2. Update estimate with measurement
$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1})$
3. Update the estimate uncertainty
$P_{n,n} = (I - K_nH)P_{n,n-1}(I - K_nH)^T + K_nR_nK_n^T$
Time Update ("Predict")
1. Extrapolate the state
$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n$
2. Extrapolate uncertainty
$P_{n+1,n} = FP_{n,n}F^T + Q$
Initial Estimate: $\hat{x}_{0,0}$, $P_{0,0}$

Figure 22: Kalman Filter operation[36]

Chapter 6

Implementation and Results

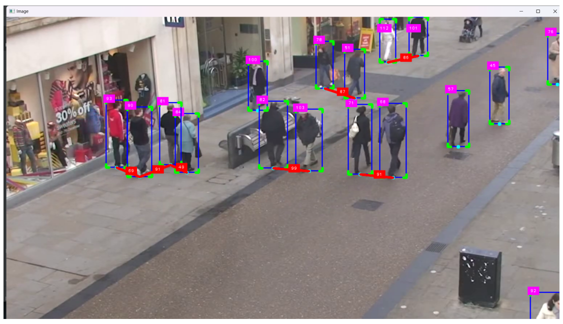
Based on the systems that were described in the methodology, we implemented our system both with a normal tracking system and an advanced deep sort tracking algorithm in our Faster R-CNN & YOLO-based detection model algorithms and achieved great results.

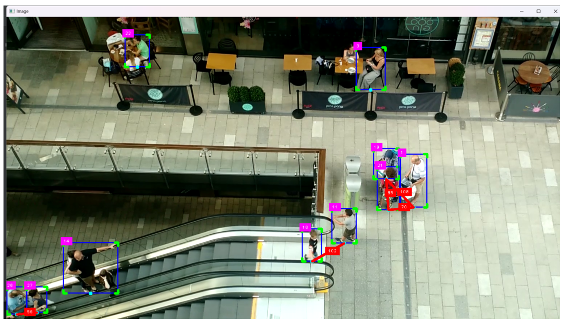
6.1 Faster R-CNN & YOLOv8 based system implementation(without DeepSort)

We conducted experiments using two deep-CNN-based object detectors, namely Faster R-CNN and YOLOv8. The pedestrian detection in the image and the corresponding social distance using the YOLOv8 model are shown in Figures **??** below The results indicate that some detections were missed, which could be due to two reasons. Firstly, occlusions caused by objects such as dining tables can lead to missed detections, as observed in the Mall Dataset. Secondly, if the size of the pedestrian is too small, missed detections may occur. The Oxford Town Centre dataset in 23a has a total of 4500 frames annotated. There were 1661 people in this dataset. Our system detected a total of 1664 pedestrians. The total distancing violation count was 1536. The density of pedestrians was low in this video. The threshold for distance violation was 125 pixels in this case.

In the Mall Video in 23bwe detected a total of 241 people. The total violation count was 72. The threshold for counting violations was 100 pixels. Out of the three videos, the Mall has a comparatively lower number of people violating the distancing rules. This is due to the density of people being very low in the mall. There were a few missed detections due to Tables obstructing the view of the person from the camera.

In the Airport video in 23c, the number of total detected people was 4268. The number of total detected distancing violations was 5442. This video had the highest density of people. So the number of violations was a lot higher than in other videos. This was mostly due to one person passing by multiple other people at close distance. The threshold of distancing violation was 70 for the video. Some detections were missed due to the people being very far from the camera and appearing too small


(a) Results from Oxford Town Ctr. video


(b) Results from the Shopping Mall video

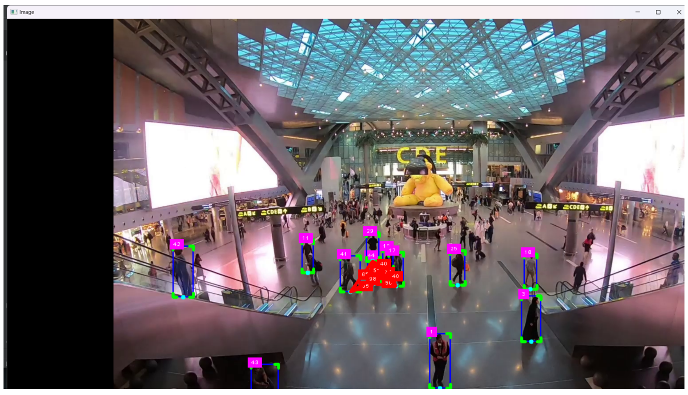

(c) Results from the Airport video

Figure 23: implemented models results of type O, M, and A

to be recognized by the model.

The results in 30indicate that some detections were missed, which could be due to two reasons. Firstly, occlusions caused by objects such as dining tables can lead to missed detections, as observed in the Mall Dataset. Secondly, if the size of the pedestrian is too small, missed detections may occur.

Both detectors in Table III have fast enough inference time to detect social distancing in real time. The accuracy of the detectors is based on results from the MS COCO dataset.

Model	**mAP(%)**	**Inference time(s)**
Faster R-CNN	42.1–42.7	0.145
YOLOv8	37.3-53.9	0.018-0.046

Table III: Comparison between models

In the comparison, we can see that Faster R-CNN is much more accurate than some of the YOLOv8 models. However, YOLOv8 is much faster than Faster R-CNN.

6.2 YOLOv7/v8 & Deepsort based system implementation

We performed our second system which performed better compared to the faster R-CNN-based system. Our system was first implemented on pre-trained model weight and some perspective corrections were needed. Therefore, we trained our dataset and used our custom weight to get better results. In different videos we encountered different angles for each video and the log system generated the total number of frames and total number of violations that occurred on that video for each person. Therefore, our system does not need human intervention.

6.2.1 YOLOv7 & Deepsort based system results

The comparative analysis of the video perspective according to the frames of each different perspective video is described in the following table IV.

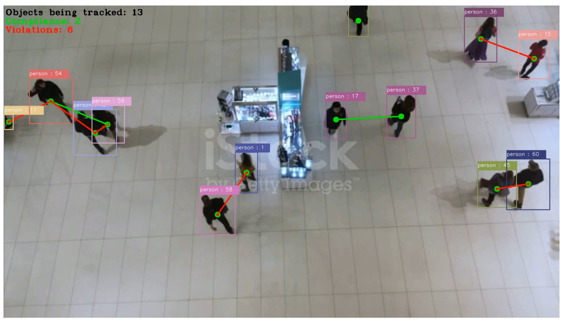
Table IV: Custom dataset(YOLOv7 ”best.pt” weight) pre-processing data table

System	**Video**	**Fps**	**Total frames**	**Total violations**	**Gpu**
YOLOv7 + Deep sort	pixels video 2670	4.26	341	1572	Gtx 1070
YOLOv7 + Deep sort	pixels video 2670	11.86	341	1572	T4(collab)
YOLOv7 + Deep sort	pixels video 2670	2.5	341	1572	Gtx 1050 ti
YOLOv7 + Deep sort	oxford town center	7.14	7501	12505	Gtx 1070
YOLOv7 + Deep sort	oxford town center	16.87	7501	12505	T4(collab)
YOLOv7 + Deep sort	oxford town center	4.16	7501	12505	Gtx 1050 ti
YOLOv7 + Deep sort	town center 2	15.19	3408	754	Gtx 1070
YOLOv7 + Deep sort	town center 2	23.45	3408	754	T4(collab)
YOLOv7 + Deep sort	town center 2	10.26	3408	754	Gtx 1050 ti
YOLOv7 + Deep sort	pedestrians	20.45	733	165	Gtx 1070
YOLOv7 + Deep sort	pedestrians	31.45	733	165	T4(collab)
YOLOv7 + Deep sort	pedestrians	13.13	733	165	Gtx 1050 ti
YOLOv7 + Deep sort	cctv4	8.83	393	2467	Gtx 1070
YOLOv7 + Deep sort	cctv4	13.45	393	2467	T4(collab)
YOLOv7 + Deep sort	cctv4	3.12	393	2467	Gtx 1050 ti

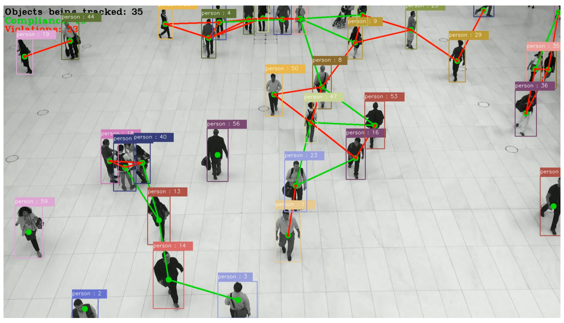
From the analysis of the output that is real-time is saved in an ”input_output” folder where the output is saved based on the input video after processing. From those, some of the frames are presented with descriptions as follows:

From 24a, our YOLOv7 model processed a social distancing video with 393 frames. It detected 2467 violations, indicating people not following social distancing guidelines. The model achieved an average FPS of 8.83 on the GTX 1070 GPU, 13.45 on the T4 GPU, and 3.12 on the GTX 1050ti.

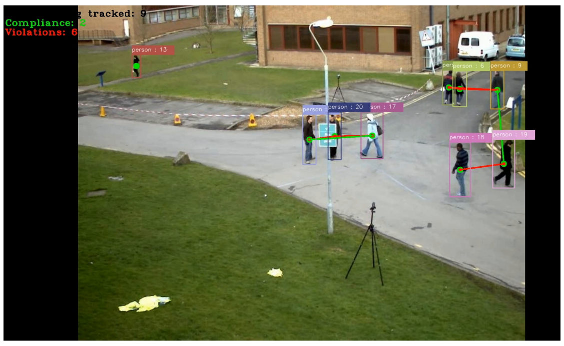
In 24e input video was evaluated using 733 frames, and 165 breaches of social distance standards were detected. The T4 (Colab) GPU displayed the fastest processing performance, with an average FPS of 31.45, surpassing the GTX 1070 (20.45 FPS) and the GTX 1050ti (13.13 FPS). These data indicate that the T4 (Colab) GPU performed the best among the three GPUs in this input video.



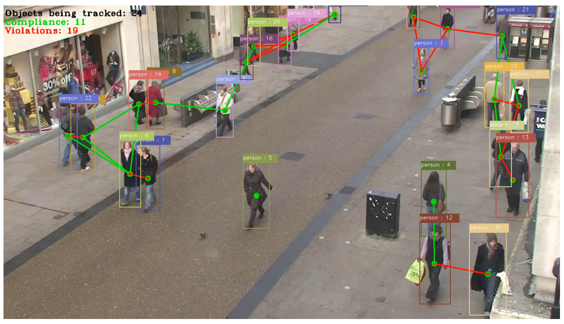
(a) Results from cctv4



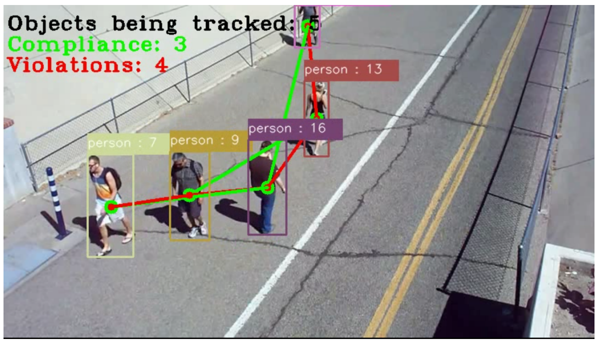
(b) Results from Pexels video 2670



(c) Results from town center 2



(d) Results from Oxford town center



(e) Results from Pedestrians

Figure 24: Implemented system results of type C, p, T2, T1, and P2

From 24b, Our YOLOv7-based system was tested on three different GPUs: GTX 1070, T4 (Colab), and GTX 1050ti. The analysis was conducted on a video containing 341 frames with a total of 1572 social distancing violations. The GTX 1070 achieved an average FPS of 4.26, the T4 (Colab) reached 11.86 FPS, and the GTX 1050ti performed at 2.5 FPS. The results demonstrate varying processing speeds across the GPUs, with T4 (Colab) showing the highest FPS, followed by GTX 1070 and GTX 1050ti with lower processing speeds.

In 24c The input video consisted of 3408 frames, and 754 social distancing violations were detected across all tested GPUs. The T4 (Colab) GPU showed the highest processing speed with an average FPS of 23.45, surpassing the GTX 1070 at 15.19 FPS and the GTX 1050ti at 10.26 FPS.

Also from 24d, The testing video has 7501 frames, with a total of 12505 social distancing infractions found. The GTX 1070 GPU produced an average FPS of 7.14, the T4 (Colab) fared quicker with 16.87 FPS, while the GTX 1050ti did the worst with 4.16 FPS.

6.2.2 YOLOv8 & Deepsort based system results

When we implemented our YOLOv8 system we faced many issues because the model is relatively new in the field and is not well optimized with the deep-sort tracking algorithm. Also, the system had a GPU initialization problem which was later solved by some research regarding this problem. Both YOLOv7 and YOLOv8 work seamlessly but although YOLOv8 has a better detection rate system-wise, violation detection numbers may vary between these two models.

When we implemented the system on YOLOv8, there were some changes in the data, and as previously shown for YOLOv7, we conducted the same implementation using custom weight for YOLOv8 and ran the system again to get the table of data preprocessing table V for YOLOv8.

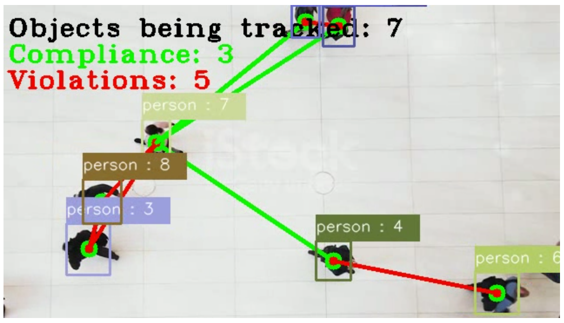
Table V: Custom dataset(YOLOv8 ”best.pt” weight) pre-processing data table

System	**Video**	**Fps**	**Total frames**	**Total violations**	**Gpu**
YOLOv8 + Deep sort	pixels video 2670	6.65	341	1491	Gtx 1070
YOLOv8 + Deep sort	pixels video 2670	13.11	341	1491	T4(collab)
YOLOv8 + Deep sort	pixels video 2670	1.28	341	1491	Gtx 1050 ti
YOLOv8 + Deep sort	oxford town center	7.69	7501	11789	Gtx 1070
YOLOv8 + Deep sort	oxford town center	19.56	7501	11789	T4(collab)
YOLOv8 + Deep sort	oxford town center	3.26	7501	11789	Gtx 1050 ti
YOLOv8 + Deep sort	town center 2	17.86	3408	689	Gtx 1070
YOLOv8 + Deep sort	town center 2	27.86	3408	689	T4(collab)
YOLOv8 + Deep sort	town center 2	13.54	3408	689	Gtx 1050 ti
YOLOv8 + Deep sort	pedestrians	24.67	733	101	Gtx 1070
YOLOv8 + Deep sort	pedestrians	40.87	733	101	T4(collab)
YOLOv8 + Deep sort	pedestrians	16.54	733	101	Gtx 1050 ti
YOLOv8 + Deep sort	cctv6	15.45	393	187	Gtx 1070
YOLOv8 + Deep sort	cctv6	20.76	393	187	T4(collab)
YOLOv8 + Deep sort	cctv6	9.57	393	187	Gtx 1050 ti
YOLOv8 + Deep sort	cctv4	10.45	393	1987	Gtx 1070
YOLOv8 + Deep sort	cctv4	15.77	393	1987	T4(collab)
YOLOv8 + Deep sort	cctv4	4.51	393	1987	Gtx 1050 ti

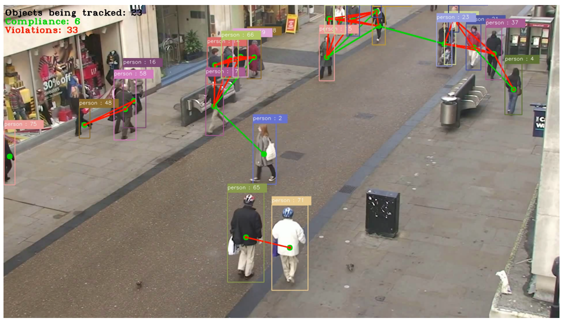
For pixel video 2670, The visual processing for this input video was done on the GTX 1070, T4 (Colab), and GTX 1050ti GPUs. A total of 1491 social distancing violations were found in the video’s 341 frames, across all GPUs.

According to our experiment, the T4 (Colab) GPU outperformed both the GTX 1070 (6.65 FPS) and the GTX 1050ti (1.28 FPS) in terms of processing performance, functioning at an average of 13.11 FPS.

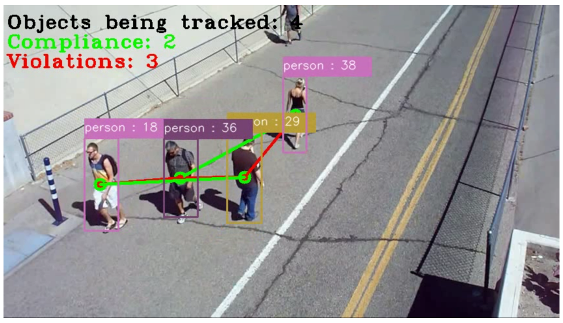
In our yolov8 model with deepsort, the Oxford town center input video has a total of 7051 frames and the number of violations is 11789. The T4 (Colab) GPU achieved the fastest processing performance, with an average FPS of 19.56, outperforming the GTX 1070 at 7.69 FPS and the GTX 1050 ti at 3.26 FPS.



(a) Results from cctv6



(b) Results from Oxford Town Center



(c) Results from Pedestrians

Figure 25: Implemented system results of type C6, Ox and p3

After running the town center 2 input video through our model with three different GPUs, we received the following FPS: GTX 1070 at 17.86, T4 (Colab) at 27.86, and GTX 1050ti at 13.54. The total number of violations was 689 in 3408 frames.

The input video pedestrians had 733 frames, and there were 101 social distancing violations recorded across all GPUs. The T4 (Colab) GPU achieved the fastest processing speed, with an average FPS of 40.87, outperforming both the GTX 1070 at 24.67 FPS and the GTX 1050ti at 16.54 FPS.

This input video cctv4 has 393 frames, with a total of 1987 social distancing violations detected across all GPUs. The T4 (Colab) GPU achieves an average FPS of 15.77, surpassing both the GTX 1070 at 10.45 FPS and the GTX 1050ti at 4.51 FPS.

6.3 Result Analysis

TableVI represents the performance analysis of Faster R-CNN and YOLOv8 with tracking(without deep sort) and also YOLOv7/YOLOv8 with Deep sort tracking algorithm. The performance metrics used for assessment included precision, recall, mean Average Precision at IoU threshold 0.5 (mAP@0.5), and mean Average Precision from IoU threshold 0.5 to 0.95 (mAP@0.5:.95).

The results demonstrate that both YOLOv7(with Deep sort) and YOLOv8(with Deep sort) achieved impressive performance in person detection. YOLOv8 exhibited a slight advantage over YOLOv7 in all evaluation metrics, showcasing its superiority. Here is a comprehensive analysis of the results:

6.3.1 Performance Analysis of Both Systems

From the TableVI below we can see that all models performed quite well but based on efficiency and overall structure of the system, they may vary because of the use cases.

Table VI: Performance analysis table

System	**Precision**	**Recall**	**mAP@0.5**	**mAP@0.5:.95**
Faster R-CNN + tracking (without deep sort)	0.93	0.90	0.91	0.79
Yolov8 + tracking (without deep sort)	0.96	0.97	0.99	0.80
Yolov7 + tracking (Deep sort)	0.985	0.969	0.986	0.728
Yolov8 + tracking (Deep sort)	0.98	0.973	0.993	0.827

6.3.2 Performance Metrics

To understand the basic formulas of the above terms some concepts need to be cleared:

Precision: A measure of performance called precision assesses how accurately the model's correct predictions were produced. By dividing the entire amount of positive projections by the number of real positive predictions, it is calculated.

Recall: is the percentage of occurrences that were found to be relevant.

F1-score: The F1 score is a metric commonly employed to evaluate the performance of a binary classification model. It becomes particularly useful when dealing with imbalanced datasets. The F1 score is calculated as the harmonic mean of accuracy and recall, combining the two metrics. By doing so, it strikes a balance between the model's ability to correctly identify positive instances (recall) and its overall accuracy. This balance makes the F1 score a suitable evaluation measure, especially in situations where class distribution is uneven.

- **True positive(TP):** Number of samples that were identified and confirmed to be positive.
- **False positive(FP):** Quantity of samples that were mistakenly thought to be positive.
- **True negative(TN):** Number of samples that have been flagged as negative and are also negative.
- **False negative(FN):** Number of samples that were believed to be negative but came out to be positive.

Table VII: True & False positives & negetives

	Ground Truth	**Predicted**
TP	Positive	Positive
FP	Negative	Positive
TN	Negative	Negative
FN	Positive	Negative

Precision: Faster R-CNN(without deepsort) and YOLOv8(without deepsort) achieved a precision of 0.93 and 0.96. YOLOv7(with deep sort) achieved a precision of 0.985, indicating that 98.5% of the predicted persons were correct. YOLOv8(with deepsort), with a precision of 0.98, also performed exceptionally well in accurately detecting persons in images. These high precision scores suggest that the majority of the predicted persons are indeed true positives.

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

Recall: Both systems demonstrated excellent recall values. Faster R-CNN(without deep sort) and YOLOv8(without deep sort) achieved 0.90 and 0.97. YOLOv7(with deep sort) achieved a recall of 0.969, while YOLOv8(with deep sort) achieved a recall of 0.973. These values indicate that the models successfully identified nearly 97% of the actual persons present in the images, showcasing their ability to detect the most positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

Using precision and recall we get the evaluation metric of the F1-score of our systems.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.3}$$

mAP@0.5: YOLOv7(with deepsort) achieved a mean Average Precision of 0.986 at IoU threshold 0.5, while YOLOv8(with deepsort) achieved an even higher score of 0.993. Faster R-CNN(without deep sort) and YOLOv8(without deep sort) had 0.91 and 0.99 respectively. These results indicate that both models effectively localized persons with relatively high IoU thresholds, further validating their robustness in detecting objects.

mAP@0.5:.95: The mAP score from IoU threshold 0.5 to 0.95 provides insights into the models' performance across a broader range of IoU thresholds. YOLOv7(with deepsort) attained a score of 0.728, and YOLOv8(with deepsort) achieved a higher score of 0.827, indicating their ability to maintain high precision and recall even for more challenging detections.

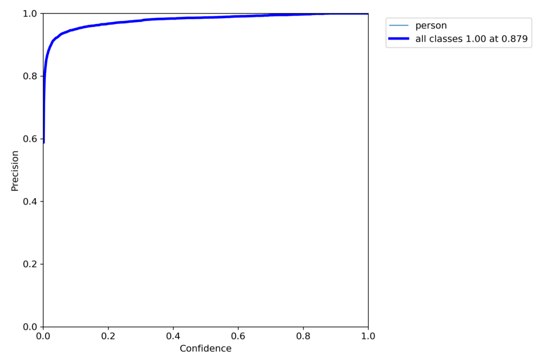
Faster R-CNN(without deep sort), YOLOv8(without deep sort), YOLOv7(with deep sort), and YOLOv8(with deep sort) demonstrated outstanding performance in identifying persons in images. YOLOv8 exhibited a slight edge over YOLOv7, achieving higher precision, recall, and MAP scores. These results suggest that YOLOv8 is a more effective model for real-time person detection tasks, where accuracy and speed are crucial.

All performance metrics are very desirable and we got excellent results which means our proposed system was a huge success combined with the functionality of the overall system. These performance metrics demonstrate the overall performance of our systems.

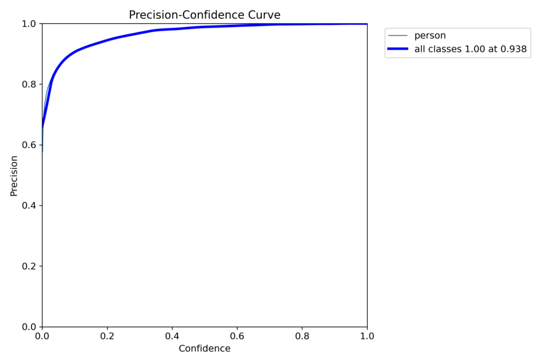
Therefore all the necessary performance metrics curves are presented below. These performance metrics demonstrate the overall performance of our systems.

6.3.3 Performance Metrics Visulalization

Precision:



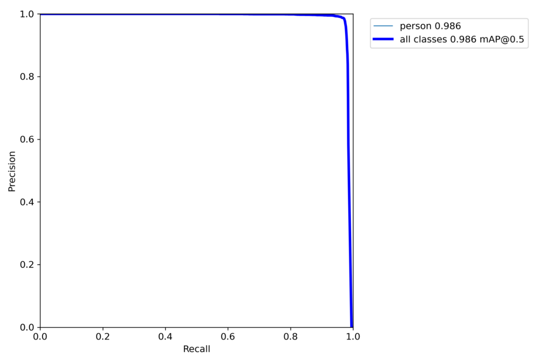
(a) Precision curve of YOLOv7



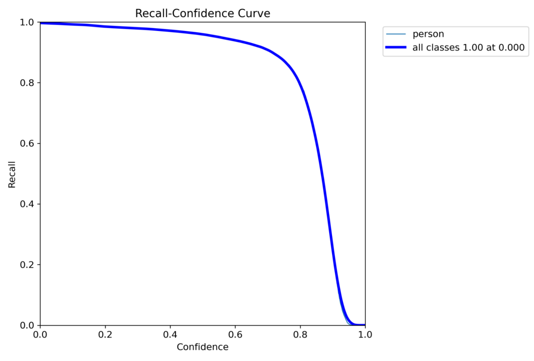
(b) Precision curve of YOLOv8

Figure 26: P curve of YOLOv7 and YOLOv8

Recall:



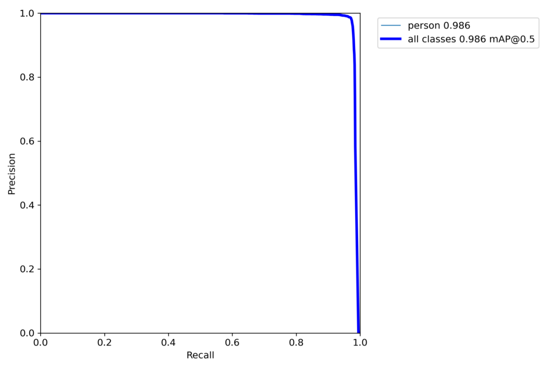
(a) Recall curve of YOLOv7



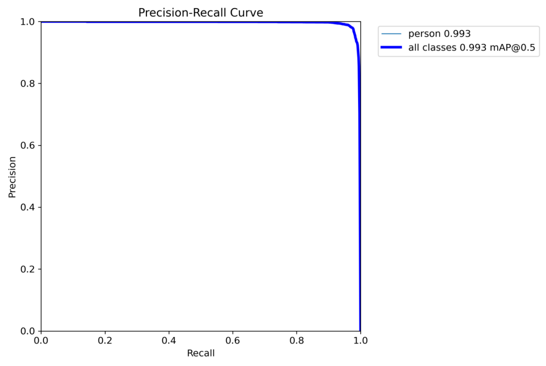
(b) Recall curve of YOLOv8

Figure 27: R curve of YOLOv7 and YOLOv8

PR curve:



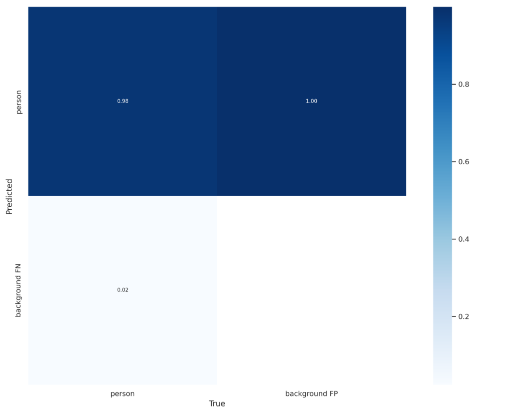
(a) Precision & Recall curve of YOLOv7



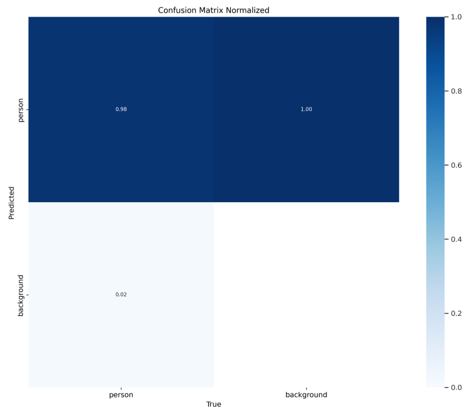
(b) Precision & Recall curve of YOLOv8

Figure 28: PR curve of YOLOv7 and YOLOv8

6.3.4 Confusion Matrix & F1-score

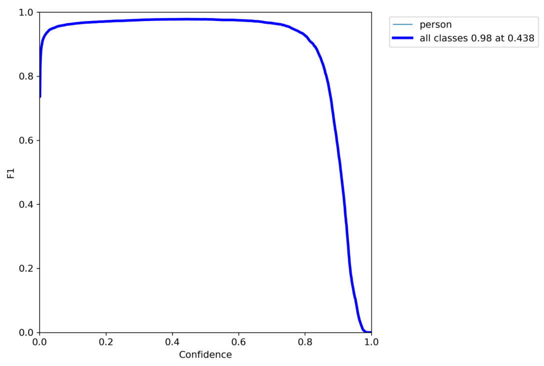


(a) Confusion matrix of YOLOv7

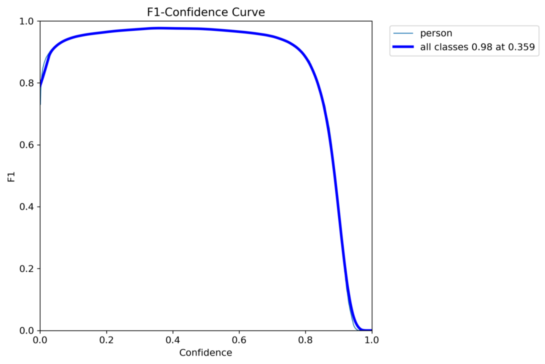


(b) Confusion matrix of YOLOv8

Figure 29: Confusion Matrix of YOLOv7 and YOLOv8



(a) F1 curve of YOLOv7



(b) F1 curve of YOLOv8

Figure 30: F1 curve of YOLOv7 and YOLOv8

6.3.5 Log system

After running the simulation, all the data(violation count, generated IDs) are stored frame by frame in a CSV file format. The data is processed in real-time for each input video.

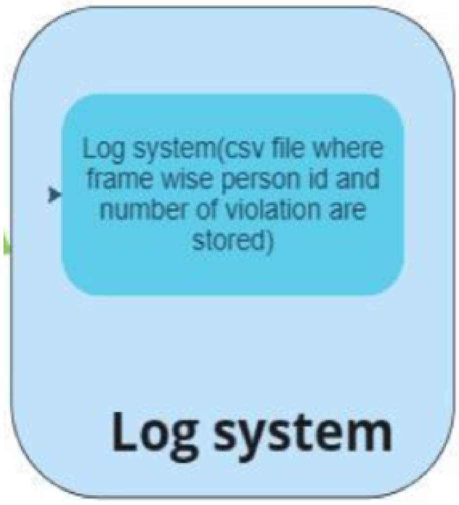
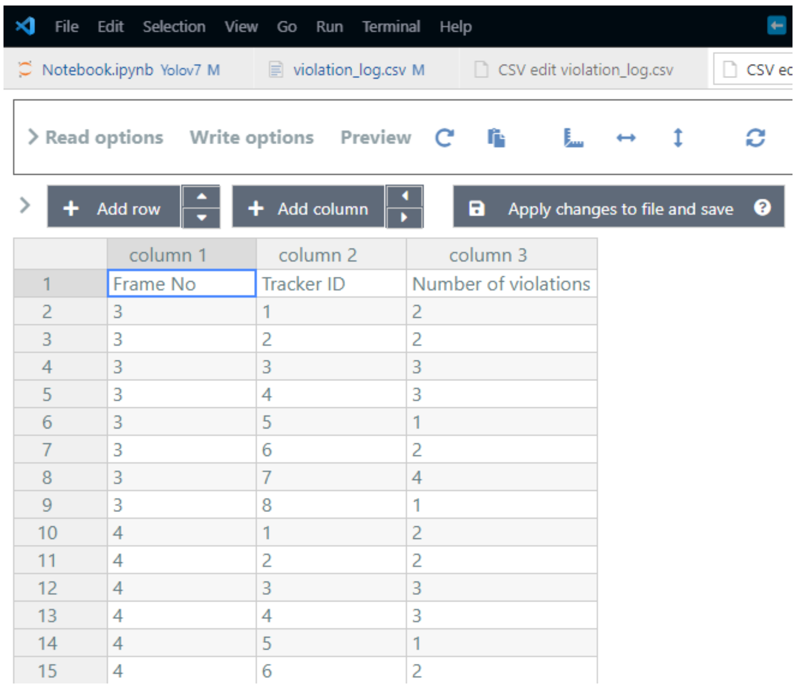

Log system(csv file where frame wise person id and number of violation are stored)
Log system

Figure 31: Log System Diagram

When the CSV file is generated it has all real-time data of violation and generated IDs from which an alert system can be generated. The generated data can be also used as a dataset for an identification system implementation.


File Edit Selection View Go Run Terminal Help
Notebook.ipynb Yolov7 M
violation_log.csv M
CSV edit violation_log.csv
CSV ec
Read options Write options Preview
Add row
Add column
Apply changes to file and save

	column 1	column 2	column 3
1	Frame No	Tracker ID	Number of violations
2	3	1	2
3	3	2	2
4	3	3	3
5	3	4	3
6	3	5	1
7	3	6	2
8	3	7	4
9	3	8	1
10	4	1	2
11	4	2	2
12	4	3	3
13	4	4	3
14	4	5	1
15	4	6	2

Figure 32: Log System Diagram

Chapter 7

Conclusion & Future Work

Overall in this research, There are two systems that we proposed one is based on Faster R-CNN and YOLOv8 detection algorithms with custom tracker, and another is based on YOLOv7/YOLOv8 detection algorithm with deepsort tracking algorithm using Kalman filter for state estimation.

Future work may focus on increasing the accuracy and robustness of the proposed techniques used for social distancing monitoring. Future researchers can use IoT (Internet of Things) devices on such low-cost, effective, and dynamic systems in real-life scenarios. Iot-based devices can help to monitor social distancing in crowded public places like shopping malls, bank booths, train stations, and universities. The device can capture real-time footage using a camera module and run algorithms to detect social distancing. If the distance is close it can send audio-visual cues and alert people. The device can also provide data and insights to the authorities to regulate social distancing in those public places.

Future researchers can combine with other computer vision techniques such as thermal imaging etc for better accuracy and robustness. This research can aid in both future pandemics and the spread of new infectious diseases. Although the COVID-19 pandemic made computer vision surveillance and social distance monitoring popular, it can also be used in other areas. Computer vision can also be used to examine crowd behavior patterns and security purposes. This can make it easy to anticipate and prevent future vandalism and criminal activity in busy public areas.

To address the challenges posed by COVID-19 and similar potential health threats, a novel method is being considered. The proposed solution involves the deployment of an advanced monitoring system capable of tracking social distancing practices and promptly identifying individuals failing to comply with prescribed regulations. By employing various technologies, such as surveillance cameras, drones, and wearable devices, this system can collect and analyze data of individual proximity in crowded environments. Consequently, it can identify locations where social distancing measures are inadequately observed, enabling timely interventions to curtail further transmission.

Bibliography

[1] J. Si, H. Zhang, and C.-G. Li, "Person re-identification via region-of-interest based features," in *2014 IEEE Visual Communications and Image Processing Conference*, IEEE, 2014, pp. 249–252.

[2] F. Ma, X.-Y. Jing, X. Zhu, Z. Tang, and Z. Peng, "True-color and grayscale video person re-identification," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 115–129, 2019.

[3] J. Zhu, H. Zeng, J. Huang, *et al.*, "Body symmetry and part-locality-guided direct nonparametric deep feature enhancement for person reidentification," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2053–2065, 2019.

[4] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, pp. 137–154, 2004.

[5] T. R. Gadekallu, M. Alazab, R. Kaluri, P. K. R. Maddikunta, S. Bhattacharya, and K. Lakshmanna, "Hand gesture classification using a novel cnn-crow search algorithm," *Complex & Intelligent Systems*, vol. 7, pp. 1855–1868, 2021.

[6] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using yolo: Challenges, architectural successors, datasets and applications," *multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.

[7] S. Bhattacharya, P. K. R. Maddikunta, Q.-V. Pham, *et al.*, "Deep learning and medical image processing for coronavirus (covid-19) pandemic: A survey," *Sustainable cities and society*, vol. 65, p. 102 589, 2021.

[8] H.-J. Lee and J.-H. Chung, "Hand gesture recognition using orientation histogram," in *Proceedings of IEEE. IEEE Region 10 Conference. TENCON 99.'Multimedia Technology for Asia-Pacific Information Infrastructure'(Cat. No. 99CH37030)*, IEEE, vol. 2, 1999, pp. 1355–1358.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[12] J. Redmon and A. Farhadi, "Ieee 2017 ieee conference on computer vision and pattern recognition (cvpr)-honolulu, hi (2017.7. 21-2017.7. 26)," in *2017 ieee conference on computer vision and pattern recognition (cvpr)-yolo9000: better, faster, stronger*, IEEE Conference on Computer Vision & Pattern Recognition. IEEE, 2017, pp. 6517–6525.

[13] C. Li, L. Li, H. Jiang, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[14] M. Hussain, "Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, 2023.

[15] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 3645–3649.

[16] D. Demirović, "An implementation of the mean shift algorithm," *Image Processing On Line*, vol. 9, pp. 251–268, 2019.

[17] I. Ahmed, M. Ahmad, J. J. Rodrigues, G. Jeon, and S. Din, "A deep learning-based social distance monitoring framework for covid-19," *Sustainable cities and society*, vol. 65, p. 102 571, 2021.

[18] M. E. Rusli, S. Yussof, M. Ali, and A. A. A. Hassan, "Mysd: A smart social distancing monitoring system," in *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, IEEE, 2020, pp. 399–403.

[19] S. Elbishlawi, M. H. Abdelpakey, A. Eltantawy, M. S. Shehata, and M. M. Mohamed, "Deep learning-based crowd scene analysis survey," *Journal of Imaging*, vol. 6, no. 9, p. 95, 2020.

[20] V. W. Wong and K. H. Law, "Fusion of cctv video and spatial information for automated crowd congestion monitoring in public urban spaces," *Algorithms*, vol. 16, no. 3, p. 154, 2023.

[21] D.-H. Im, Y.-S. Seo, H. Kim, E. Hwang, and J. Park, "Person re-identification in movies/dramas," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2020, pp. 1596–1598.

[22] P. Witzig, E. Upenik, and T. Ebrahimi, "Open-set person re-identification through error resilient recurring gallery building," in *2021 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2021, pp. 245–249.

[23] O. Hamdoun, F. Moutarde, B. Stanciulescu, and B. Steux, "Person re-identification in multi-camera system by signature based on interest point descriptors collected on short video sequences," in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, IEEE, 2008, pp. 1–6.

[24] H. Choi and M. Jeon, "Deep neural network for person re-identification in a non-overlapping camera network," in *2017 International Conference on Control, Automation and Information Sciences (ICCAIS)*, IEEE, 2017, pp. 193–196.

[25] N. Roy and S. Debarshi, "Uav-based person re-identification and dynamic image routing using wireless mesh networking," in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, 2020, pp. 914–917.

[26] H. Zhou, C. Huang, and H. Cheng, "A relation network design for visible thermal person re-identification," in *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, IEEE, 2021, pp. 511–515.

[27] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "Real-time vehicle detection based on improved yolo v5," *Sustainability*, vol. 14, no. 19, p. 12 274, 2022.

[28] A. Li, L. Liu, K. Wang, S. Liu, and S. Yan, "Clothing attributes assisted person reidentification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 869–878, 2014.

[29] N. Perwaiz, M. M. Fraz, and M. Shahzad, "Smart visual surveillance: Proactive person re-identification instead of impulsive person search," in *2020 IEEE 23rd international multitopic conference (INMIC)*, IEEE, 2020, pp. 1–6.

[30] N. Martinel, G. L. Foresti, and C. Micheloni, "Person reidentification in a distributed camera network framework," *IEEE transactions on cybernetics*, vol. 47, no. 11, pp. 3530–3541, 2016.

[31] M. Rezaei and M. Azarmi, "Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic," *Applied Sciences*, vol. 10, no. 21, p. 7514, 2020.

[32] N. S. Punn, S. K. Sonbhadra, S. Agarwal, and G. Rai, "Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsort techniques," *arXiv preprint arXiv:2005.01385*, 2020.

[33] M. A. Ansari and D. K. Singh, "Monitoring social distancing through human detection for preventing/reducing covid spread," *International Journal of Information Technology*, vol. 13, no. 3, pp. 1255–1264, 2021.

[34] I. S. Walia, D. Kumar, K. Sharma, J. D. Hemanth, and D. E. Popescu, "An integrated approach for monitoring social distancing and face mask detection using stacked resnet-50 and yolov5," *Electronics*, vol. 10, no. 23, p. 2996, 2021.

[35] J. Qin and N. Xu, "Reaserch and implementation of social distancing monitoring technology based on ssd," *Procedia Computer Science*, vol. 183, pp. 768–775, 2021.

[36] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.