

Image Processing Based Human Detection and Social Distancing Measurements with Monitoring via Fine Tuned Deep Learning and Computer Vision

by

Anurag Datta

18101369

Kaniz Fatema

19201132

Nowshin Tasnim

19101655

Faria Sitara

18101295

Mrithik Kanti Das

17101047

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 and Engineering

Department of Computer Science and Engineering
Brac University
May 2022

© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that,

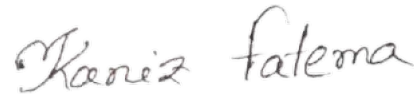
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which 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:



Anurag Datta

18101369



Kaniz Fatema

19201132



Nowshin Tasnim

19101655



Faria Sitara

18101295



Mrithik Kanti Das

17101047

Approval

Image Processing based human detection and social distancing measurements with monitoring via fine-tuned Deep learning and Computer vision” submitted by

1. Anurag Datta (18101369)
2. Kaniz Fatema (19201132)
3. Nowshin Tasnim (19101655)
4. Faria Sitara (18101295)
5. Mrithik Kanti Das (17101047)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on May 29, 2022.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty, PhD

Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Assistant 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

Ethics Statement

Hereby, we, the members, consciously assure that for the manuscript /insert title/ the following is fulfilled:

- This content is the authors' original work that has never been published before.
- The authors' research and analysis are accurately and completely represented in the work.
- Co-authors and co-researchers are appropriately acknowledged in the study for their significant contributions.
- The findings are contextualized appropriately in light of previous and ongoing research.
- All sources are cited appropriately. Text that has been literally copied must be identified as such with quote marks and a suitable reference.
- All authors personally and actively contributed to the article's creation and will accept public responsibility for its content.

Violations of the Ethical Statement standards may result in serious consequences. We agree with the preceding declarations and certify that this submission complies with the rules of BRAC University.

Abstract

The COVID-19 pandemic has significantly affected day to day lifestyle all over the planet by disequilibrating social order. It moreover added anxiety about the capability of the world's democracies to cope with the vital and crucial emergencies. We should urgently restore a necessary methodology to fathom the emergency and rigorously depict a course forward. The Division of Public Health authorities have recommended everyone to uphold social distancing with a view to diminishing the number of physical encounters. To keep a record of social distancing from an overhead standpoint, we established a computer vision deep learning framework. Our schemed system utilized the object recognition paradigm to spot and identify people in video sequences or frames.

In our research, we assess the classification performance of two distinct multilayer neural network models named YOLO using OpenCV and TensorFlow which are used in the implementation process of an automatic recognition system. Amongst these using SSD, CUDA, and CUDNN we achieved a success rate in the classification. Neural networks were trained on a dataset where we used COCO dataset methods. At a time when neural networks are increasingly being utilized for a spectrum of uses, it is essential to select the proper model for the classification process that can attain the ultimate accuracy with the least amount of training duration. The demonstration created by us allows the insertion of images and the creation of their datasets, this allows the user to train a model using their chosen parameters. The models can then be saved and used in other systems. Moreover, to prevent future crucial situations and by keeping in the head about COVID affected situations on various global aspects this work will become an integral part of contributing to the term "Social Distancing" by implementing this sustainably and with one of the best results outcomes in our proposed image processing based human detection and social distancing measurements with monitoring via fine tuned deep learning and computer vision. Because coronavirus sickness has had such a negative influence on the world economy, this research tries to reduce the further impacts while minimizing resource loss. Also, create a very accurate detection mechanism to aid in the tracking of social distancing. In these types of serious situations, adequate actions must be taken and help to assist further research and work as an example for future works on this segment.

Keywords: Human detection, Social distancing, YOLO algorithm, Image processing, Deep learning, TensorFlow, Computer vision, MobileNet SSD

Dedication

This thesis is dedicated to our university's mentors, without whom we would not have been capable of completing this thesis. Our professors were more than just academic mentors; they were also a source of encouragement and support when we needed it most.

Acknowledgement

In the first place, we would like to thank Almighty Allah for allowing us to complete our thesis on time and without deterrent.

Having said that, we would like to express our gratitude to Dr. Amitabha Chakrabarty (PhD), our distinguished instructor and supervisor, for his unwavering support and tenacious oversight, which allowed us to complete our project. In addition, we would like to thank our supportive friends who have been there for us during the difficult times. Lastly, to our parents, without their unwavering support it may be impossible. With their gracious assistance and prayers, we are now on the verge of graduating.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Abstract	v
Dedication	vi
Acknowledgment	vii
Table of Contents	viii
List of Figures	x
List of Tables	xi
Nomenclature	xiii
1 Introduction	1
1.1 Thoughts behind the Model	1
1.2 Motivation	2
1.3 Research Objective	2
1.4 Problem Statement	5
2 Literature Review	8
3 Research Methodology	12
4 Data and Preliminary Analysis	20
4.1 Human Detection and Social Distancing Measurements	20
4.2 Model Dataset Pre-Trained Libraries	24
4.2.1 Anaconda in Python	24
4.2.2 NVIDIA GPU	24
4.2.3 CUDA	25
4.2.4 cuDNN	27
4.2.5 TensorFlow Object Detection API	27
4.2.6 TensorFlow Model Garden	28
4.2.7 Protobuf	28
4.2.8 COCO API	29

4.2.9	SSD MobileNetv2	29
4.2.10	YOLO	31
4.2.11	Matplotlib	31
4.3	Input and Training Dataset	32
4.3.1	YOLOv5	32
4.3.2	TensorFlow	34
5	Result Analysis	38
5.1	Detecting pedestrian points and measuring distance	38
5.1.1	Performance of TensorFlow	39
5.1.2	Data output generated from TensorFlow	40
5.2	“You Only Look Once”- YOLO staging	40
5.2.1	Performance of YOLO	41
5.2.2	Data output generated from YOLO model	42
5.3	Comparison	42
6	Conclusion and Future Work	44
6.1	Conclusion	44
6.2	Future Work	45
	Bibliography	51

List of Figures

1.1	Overview of the tasks regarding real-time object recognition. Applying real-time object recognition applications to low-cost Edge AI platforms is a challenging task. An adequate inference time is needed to obtain the classification real-time [61]	2
3.1	Structures of YOLOv5s [31], [73]	13
3.2	Human detection workflow using YOLOv5	13
3.3	YOLOv5 Performance	14
3.4	YOLOv5 Performance [66]	15
3.5	Identifying through image processing	15
3.6	Generalized methodology for image classification TensorFlow framework	16
3.7	A schematic TensorFlow dataflow graph for a training pipeline, containing subgraphs for reading input data, training, checkpointing state and preprocessing [32]	16
3.8	TensorFlow sample graph [26]	17
3.9	The layered TensorFlow architecture[32]	18
3.10	Use of CPUs and GPUs in different stages of TensorFlow	19
4.1	The flowchart of the working procedure of YOLOv5 model	21
4.2	CUDA WorkFlow	25
4.3	CUDA Architecture	26
4.4	Keypoints detected by OpenPose on COCO Dataset	29
4.5	MobileNetv2 Architecture	30
4.6	COCO Dataset in YOLOv5	33
4.7	Determining pedestrian points and measuring distance	34
4.8	TensorFlow Installation	35
4.9	Determining pedestrian points and measuring distance in TensorFlow model	36

List of Tables

5.1	TensorFlow accuracy table	39
5.2	Performance of TensorFlow	39
5.3	Data output from TensorFlow model	40
5.4	YOLOv5 accuracy table	41
5.5	Performance of YOLO model	41
5.6	Data output from YOLOv5 model	42
5.7	Comparison of YOLOv5 and TensorFlow	43

Nomenclature

API Application programming interface

AVI Audio Video Interleave

CCTV Closed Circuit Television

COCO Common objects in Context

CPU Central Processing Unit

CSV Comma Separated Values

CUDA Compute Unified Device Architecture

cuDNN Deep Neural Network Library

DPM Deformable Parts Model

FD Fourier Descriptor

FPS Frames Per Second

GMM Gaussian Mixture Model

GPU Graphics Processing Unit

GUI Graphical User Interface

HD High definition

HTML Hypertext Markup Language

ID Identity Document

ILVRC ImageNet Large Scale Visual Recognition Challenge

IOU Intersection Over Union

JPEG Joint Photographic Expert Group

JSON JavaScript Object Notation

mAP Mean Average Precision

MATLAB Matrix Laboratory

NVIDIA Next Version

ONNX Open Neural Network Exchange
openCV Open Source Computer Vision Library
PNG Portable Network Graphics
PyPI Python Package Index
RAM Random Access Memory
RFCN Random Field Convolutional Network
SIGMODELS Special Interest Group on Machine Learning Models
SOTA State Of The Art
SSD Single Shot Detector
SVG Scalable Vector Graphics
VGA Video Graphics Array
VGG Visual Geometric Group
YOLO You Only Look Once

Chapter 1

Introduction

1.1 Thoughts behind the Model

In 2022, researchers did research on various complex segments and technology has reached an unbelievable era since its invention. It will rise more on future days but still we are facing a devastating pandemic named COVID-19 from last portion of 2019, this virus showed, still we are not capable to prevent pandemic but this makes us realize more to think about before and after precautions into ongoing and future problems that might arise on this world [57]. As always technology always preserves the biggest role when in any situation arises. The word “Social Distance” is the greatest notion for regulating efforts to stop COVID-19 from spreading. The primary purpose is to reduce affected rates and maintain social distance. To control the spread of this disease, persons should stay at least 1-meter (m) apart, according to WHO (World Health Organization) guidelines [57]. And in this field here comes technology with implementation and research on various innovative methods of social distancing on various segments as well. Because of coronavirus, sickness has had such a negative influence on the world economy, this research tries to reduce the impacts of the disease while minimizing resource loss. Secondly, to create a highly precise technique for spotting individuals that will assist in the observation of social distancing. It is imperative that adequate action be taken in this critical scenario. Real-time human distancing is a challenge in Computer Vision (CV) that involves detecting, localizing, and classifying many objects in a real-time stream of frames as quickly and accurately as feasible.

In this figure, it states how the object detection places both classification and localization together. Again, when both these perform detecting multiple objects per image and also calculating the FPS. It is difficult to keep track of social distance in real-time circumstances. There are two methods to do it: manually and automatically. The manual method necessitates a large number of physical eyes to ensure that everyone is carefully adhering to social distancing rules. This is a difficult task because one cannot keep their eyes open 24 hours a day, seven days a week for constant monitoring. Many real eyes are replaced by CCTV cameras in automated surveillance systems[18], [33], [57]. CCTV cameras record video, which is inspected by an automated surveillance system. When any suspicious event occurs, the system marks the human with high risk and low risk. Security staff can take appropriate action in light of this information. As a result, the automated monitoring system

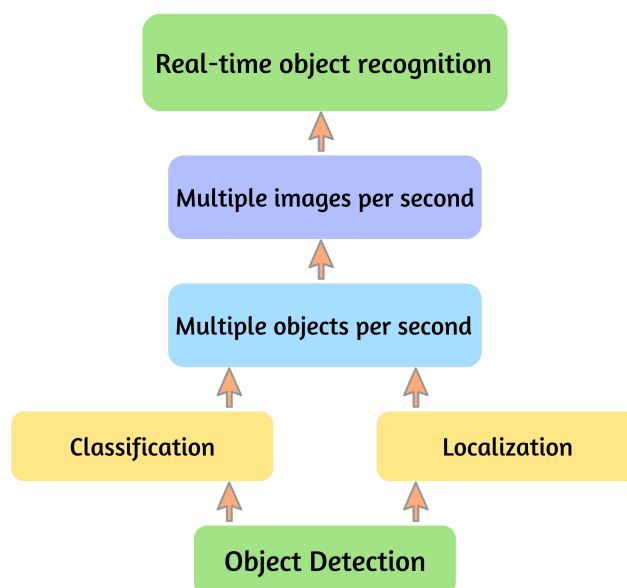


Figure 1.1: Overview of the tasks regarding real-time object recognition. Applying real-time object recognition applications to low-cost Edge AI platforms is a challenging task. An adequate inference time is needed to obtain the classification real-time [61]

has overcome numerous of the manual monitoring method's drawbacks.

1.2 Motivation

Coronavirus, in short, COVID-19 disease has established its foothold across the entire planet. It is most commonly transmitted through direct physical contact. It is necessary to install appropriate crowd monitoring and management systems in public places as a precautionary measure to reduce the likelihood of unexpected outbreaks and to improve healthcare delivery. By implementing measures of social distancing at an earlier stage, it is possible to significantly cut down on the number of new infections. This idea served as the impetus for the proposal made in this research paper, which outlines a real-time crowd monitoring and management system for the classification of social distance. So, in future if this type of disease arises, we would be ready to defend the mass gathering in crowded places so that people can be safe and sound.

1.3 Research Objective

Distancing one person from another in a social circle has shown to be an effective strategy for preventing the transmission of viruses like the COVID-19 (World Health Organization 2009). Around the world, government officials advise folks to change their habits and keep a safe distance from each other [58].

Because of this pandemic several governments are looking for technological answers. Asia has utilized a variety of technologies to combat COVID-19. The most often

utilized technology is phone location tracking, which saves COVID-19 positive individual's data and monitors their proximity to healthy people. Both Germany and Italy are utilizing anonymized location data. Similarly, South Korea released an app called Corona 100 m that tracks sick persons and alerts healthy people within 100 m of them. An app created in India lets users keep a safe distance from someone who has tested corona positive. In addition, India, South Korea, and Singapore are using CCTV footage to hunt down COVID-19 sufferers. China uses AI-powered thermal cameras to spot hot persons amid crowds [62]. A number of other researchers, including Xin use wireless signals to detect persons by analyzing phase inconsistencies and waveform alterations. As a result, it is not viable without the use of several receiving antennas. High-level feature correlations may be discovered with the use of AI, computer vision, and machine learning. Some researchers have attempted to anticipate the disease trend of certain regions using a mix of visual and geo-location cellular information, developing crowd counting and density estimate methods in public places or determining the distance of persons from prominent bevy. However, these kinds of studies are hampered by issues like a scarcity of skilled workers or the high costs of setting up and maintaining infrastructures. It is now possible for computers to analyze and interpret visual data from digital images, thanks to recent advances in Computer Vision, Deep Learning, and Pattern Recognition. These abilities may be used to empower, motivate, and undertake social distancing observation and measures. Cameras equipped with 'smart' technology, such as Computer Vision, may observe persons and determine whether or not they meet social distance norms. As a result, these gadgets require incredibly precise human detection algorithms. People detection in picture sequences is one of the most significant sub-branches of object detection and computer vision [46].

There are several advanced approaches that can help in object detection. Prior to classification, region proposal approaches were utilized by Convolutional Neural Networks (CNN) to create scores and then generated the bounding boxes around the object of interest for visualization and statistical analysis, as seen in the following examples: Despite the fact that these approaches are efficient, they need a substantial amount of training time. YOLO explores a regression-based method to dimensionally divide the bounding boxes and interpret their class probabilities because all these CNN-based algorithms need classification [44]. This is a state-of-the-art object detector designed for use in real-time applications. A single neural network is used to process the whole image. This algorithm creates bounding boxes and probabilities for each location in a picture by dividing it into little parts. The bounding boxes are weighted by projected probability [75]. In spite of this method still the accuracy of small object detection remains. For this, we tried to propose an efficient method of social distance measurement.

Newer deep learning algorithms have improved upon the earlier ones in many ways, making it easier to train computers to recognize patterns. This is how the Deformable Parts Model (DPM) works in conventional object detection methods, where the classifier runs on a slice of the picture in a sliding window approach. Bounding boxes for region suggestions are used by R-CNN, Fast R-CNN and Faster R-CNN to train the classifiers. These algorithms, notably Faster R-CNN, have high performance and accuracy [62]. Numerical computations may be performed using the

free and open-source TensorFlow library. Developed and maintained by Google under the Apache 2.0 license, it is available as open source. As long as anyone using Python, they'll be able to access the C++ API using the API. Because it was designed for both research and industrial use, Tensor Flow differs from other Deep Learning numerical libraries such as Theano in this regard. Single CPU systems, GPUs, and big distributed systems with hundreds of units may all benefit from this technology. However, the revealed model can distinguish 80 distinct elements in images and videos, as well as being significantly faster and more accurate than Single Shot Detection (SSD) [62].

Previously there was a lot of work and research done on this segment and many researchers used YOLO prior versions and other frameworks but like YOLOv5 no other model is this lightweight and adequately accurate in terms of result using dataset too. In addition, we used COCO dataset as a pre-trained model and rather than other methods, COCO dataset method plays a vital role in the consumption of less time which motivates us to use this more.

Another core task we want to achieve is a better graphical interface and mobility of use from our end so we used CUDA and CUDNN in TensorFlow which is one of the best GPU accelerated frameworks and creates a virtual environment if there is a lack of GPU and works like charm.

It is the most recent and most advanced version of the YOLO object detection series, and it has raised the standard for object detection models, outperforming Efficient-Det and previous YOLO versions thanks to the continued efforts of 58 open-source contributors. For the sparse prediction object detector, the head can be one-stage (YOLO, SSD) or two-stage (Faster R-CNN). Recent object detectors have layers too (Neck) that collect maps with features and are between the backbone and the top of the head. TorchScript, ONNX, OpenVINO, TensorRT, CoreML, TensorFlow Saved Model, GraphDef, Lite, Edge TPU, and TensorFlow.js can all be used to convert PyTorch(.pt) models. COCO is a big dataset for detecting, segmenting, and captioning objects. 330K photographs, 1+ million object instances, 75+ classifications of objects, 90+ item classifications, 5 captions for each image, and 250,000 People with pertinent aspects are just a few of the attributes [73].

About our second core model, TensorFlow is the more updated edition of Google's neural network-based system developed and launched by GOOGLE too. It is much more than a framework for fine-tuned computer vision intelligence. TensorFlow is a deep learning open-source software model created by Google. By teaching a computer to spot patterns on its own and/or make the best decisions feasible. This model was chosen because it successfully supports multi-dimensional arrays in mathematical expressions, deep neural networks, and machine-learning ideas are well-supported, the same code can be performed on both a GPU and a CPU, and it has exceptionally big data sets and machine maneuvered. All of these circumstances make it visible that YOLO and TensorFlow these two selected models we used are one of the best and depending on the factors which are time, lightweight, open-source access, use availability, accurate results, pre-train dataset ability, and most importantly contribute on the segment social distancing we choose these two model

for best outcome indeed. Various research initiatives have been done to develop as stated previously, a stronger competitive advantage in a mechanism to track physical distance is required, this thesis focuses on specific contexts to fill a research gap and to assist with social distance monitoring responsibilities, a real-world unit distance mapping approach has been used. Moreover, this work can help in the future to prevent a similar type of devastating pandemic situation where social distancing and implementation is a must.

The purpose of this study was to quantify and monitor the amount of social separation during the epidemic. For their own welfare and the environmental benefits, individuals will be encouraged by this research to preserve social distance from each other. Our system's main objective is to detect and decide whether or not someone is a member of the institution, anybody who violates the right amount of space while retaining social distance. It's likely that he or she will be monitored and instructed on the need to keep social distance if that's the situation. Also, keep an eye out for any rule violations and encourage him or her to participate in more social activities.

1.4 Problem Statement

After the COVID outbreak in this world, the world has been affected in various ways including economic factors. Millions of people died worldwide and this disease was uncontrollable and devastating for so long. This motivates a lot of researchers to work on this matter to prevent or take precautions as well. The word 'social distancing' started playing a vital role for implementing various systems in all aspects including technology specifically in computer vision and deep learning too. But previously there were not many efficient methods to detect and prevent social distancing detection. However, no one has yet focused on low-light situations too. Aside from that, no real-world unit distance mapping solution has been developed. So, in densely populated countries it is hard to detect and maintain social distancing manually so it's tough to prevent pandemic situations too. Moreover, previous models used to implement physical detection are not that accurate and heavyweight too and are more time-consuming than our selected models YOLOv5 and TensorFlow.

In terms of computer vision and deep learning, there were many obligations including lack of datasets, support issue of pre-trained datasets, and lack of GUI using graphical framework and environment successfully. For which we use CUDA and cuDNN NVIDIA-based graphical framework and module to create a virtual environment like GPU does use CPU and work flawlessly.

Our proposed paper is intended to identify those in the crowd including in low-light scenarios who are failing to maintain social distance. A three-stage approach for monitoring social distances and producing zone-based infection risk assessments includes people identification, tracking, and inter-distance calculation. This model can be used from CCTV footage or any other sources in all supported formats because of the system's real-time capability.

YOLOv5 is the most recent addition to the YOLO series. This works as the ace for

most of the developers to overcome previous problems and bindings that occurred by models or frameworks used to implement object detection. YOLOv5's running speed has been substantially increased, with the fastest speed reaching 140 frames per second. Meanwhile, YOLOv5 is compact, with a weight file that is roughly 90 percent lower than YOLOv4, allowing it to be deployed to embedded devices. YOLOv5 has a greater accuracy rate and a stronger capacity to distinguish small objects than YOLOv4. The YOLO model is a target detection system that is both standardized and real-time. It's the most sophisticated real-time object detecting system available. After RCNN, faster-RCNN, and SSD, it is another milestone target detection algorithm and unfortunately, all the previously used algorithms frameworks don't succeed much like yolo before which is one of our core reasons to pick YOLOv5. In addition, Darknet is a CUDA-based open-source neural network toolkit constructed in C. For normal usage problems, YOLO's real-time object detection technology in unifying target area prediction with target classification assessment in a single neural network model, moreover speed issue is addressable. By turning the detection mechanism into a regression issue, YOLO simplifies complex numerical steps at ease. This again makes us choose to avoid previous complex issues.

As a result, predicting incorrect background object information is difficult. Moreover, for the threshold limit sometimes this model still needs to improve but then again it is one of the tops in the current Technology World. And motivate us to work with these two by implementing Computer Vision too with our hard-working contribution to this segment. And the remaining bindings we faced in YOLOv5 can be overcome using TensorFlow. Its robust graphical interface and best numerical calculation ability on various dimensions make us to choose this as another core model of our work and YOLOv5 and TensorFlow both help us to prevent previous restrictions on computer vision by other models and will make our work more vital and step in object detection implementation.

Maintaining Social Distancing is one of the most important and successful tactics for containing the pandemic. Governments are enacting regulations limiting the minimum interpersonal distance between persons in order to comply with this constraint. People who want to practice social distance should keep at least 1.5 meters apart, work from home, avoid gatherings, and travel as little as possible, according to government guidance [50]. Separating humans in space, on the other hand, is incredibly difficult. Different people require the same important sites, such as stores, employment, and health care facilities [50]. COVID-19 has had a significant negative impact on a wide range of human activities. While pharmacological remedies are being researched and used, individual inhabitants and their adherence to public health recommendations bear some of the responsibility for decreasing the virus's impact. However, encouraging people to follow these rules has proven difficult [41]. At the beginning of COVID, the government adopted restrictive measures including shutdown. But when the number of deaths and confirmed cases started to peak the measures were eased and regular activities were partially restored at the beginning of June. Even the authorities have no control over the implementation of social distancing rules as people are frequently breaching the recommended healthcare guidelines in daylight. Violators are going unpunished due to a lack of appropriate surveillance. Even they are committing the same things again and again. People

flocked to various places without following social distancing guidelines. Authority has warned of punishment if anyone does not wear a mask and maintain recommended healthcare. But none maintained social distancing guidelines and also there was no one to monitor strictly. Heavy transit corridors, aviation corridors, personal property, construction zones, and gas plants are all critical or hazardous sites that require visual monitoring [35]. In Bangladesh, social separation is close to impossible according to Dr. ANM Nuruzzaman, a physician who previously worked as a director in the Directorate General of Health Service. Social distance is a method of preventing the spread of infectious disease. However, in a densely populated country like Bangladesh, social distancing is difficult to implement in many areas. Take, for example, the slums of Dhaka and Chittagong, which are home to millions of people. They live in such close quarters that enforcing social distance would be impossible [41]. Even after the pandemic is controlled, there will remain economic incentives for temporal distancing. Social distancing will be a fact of life to come. So, it needs to be done as smartly as possible and efficiently [38].

From this scenario, we decided to implement a social distancing analyzer that will be initially detecting humans in real time and after detecting it will measure the distance between them to detect people maintaining social distancing and people who are not maintaining social distancing. We have implemented two such models for this purpose. One is the TensorFlow model, which is based on MobileNet SSD dataset and COCO dataset. Another is the YOLOv5 model which is based on the COCO dataset and YOLO weights which is a pre-trained dataset. Our biggest challenge which we faced while implementing the TensorFlow model was that this model requires very high-end libraries and this model was itself a big size data. While installing the TensorFlow API object detection it required huge time and also it consumed immense RAM of the device which later on affected the performance of the device. Again, for the GPU support which is not mandatory for TensorFlow but in presence of the GPU supports the system to make it reliable. When TensorFlow object detection API will run, it will attempt to register on GPU devices, otherwise it will fail and TensorFlow object detection API will run over the device's CPU. These GPU prerequisites are NVIDIA GPU, CUDA Toolkit v11.2 and CuDNN 8.1.0 which are very big data in size and also for high end configured devices. This system might not be compatible with normal configured devices. Also, these are not the updated system which is for the updated system consumes more data and also not reliable for all devices. Also, TensorFlow model installation and testing requires more time and consumes more RAM than any other model. On the other hand, we also faced problems while implementing the YOLOv5 model which was using the precise YOLO weights or pretrained models. Installation and configuration of the model and the libraries was easier than that of the TensorFlow model. Determining the distance between the pedestrian points was also challenging in YOLOv5 using the other equations until the Euclidean equation was applied. Finally, determining the human detection and measuring the distance was precisely completed and we again faced difficulties in gathering the input video from which we generated the data of 15,456 on each of the models.

Chapter 2

Literature Review

Since the start of COVID-19, many courses of technical actions have been taken by numerous countries to put an end to the spread of the virus. A lot of technologies are manifested to be effective in helping people or authorities to abide by the social distance norms and regulations. Viola and Jones [1] proposed a typical solution for object identification in 2001. For feature extraction, they utilized Haar features, and in order to classify them, they utilized cascade classifiers utilizing the AdaBoost learning algorithms. This strategy is fifteen times quicker than the traditional regular techniques. By putting visual attributes and motions together, Fu-Chun Hsu recommended a crossover procedure to identify the head and shoulders. The researchers found that the HOOOF descriptor is a superior decision for dividing moving articles in video sequences and thus, can actually oversee jumbled and impeded circumstances [9]. In order to prevent the escalation of COVID-19, another study preferred an active surveillance framework by alerting people in a targeted area. This system made a two-fold improvement. In the first place, to identify SD contraventions and after that send non-intrusive audio-visual cues they presented a real-time system which is based on vision that employs state-of-the-art deep-learning methods. Secondly, by constructing a unique critical social frequency rate, they display that if the frequency of the passerby is kept below this rate, the odds of SD contraventions can be kept near nil. The suggested approach is also legitimate: neither any personal information is collected nor any person is targeted in this case, and at the same time no human administrator was present at the time of procedure. Real-world datasets were used to test this system [36].

In deep learning models in case of object detection, Convolutional Neural Networks are commonly utilized. CNN, a deep learning method which pulls a photo as input, sets biases and weights that are learnable to multiple levels of the photo, allowing those to be distinguished from one another. The progression of a Convolutional Neural Network is that it could be executed with less complexity and poor-quality entry data upon an embedded system [21]. A variety of deep learning models for object recognition, namely R-CNN, YOLO and SSD are used on countless implementations. With a view to evaluating the motions in video scenes, these models are capable of producing successful results. Different researchers, for example Xin, utilize wireless signals to distinguish people by detecting phase variations and then make changes in detection in amplitude waveforms. This, however, is not conveniently incorporated in all public areas as it necessitates numerous receiving antennas [27].

In order to discover persons a mechanical solution by applying video box frames was proposed by Ebrahim et al. A gaussian mixture along with background subtraction was used by the author with a deep learning detector [28]. A deep learning (CNN) algorithm was introduced by another author for person detection in another method. With a view to identifying people with high precision and minimal processing they exercised a blend of machine learning and deep learning technologies. Unluckily, the mentioned approach encountered issues with real-time detection due to its slow pace [23]. In an inactive gathering for a number of people who remain at one place for a prolonged time, researchers proposed a method in approach. A SVM was used for identifying patches being essential mass. Text features were used to get these patches [11].

Lately Advanced strategies have proven to be effective in solving object detecting challenges. R-CNN [32], CNN and Faster R-CNN [8] applied region proposal methodologies for getting the object count ahead of category, then for vision and numerical analysis, produced boxes surrounding the target [12]. In spite of the fact that these strategies are impressive, they require a longer training time need [12]. Because these CNN-based algorithms use classification, YOLO proposes a method based on regression for separating the enclosed boxes dimensionally and analyzing class chances [16]. For all part to consider as object in this manner, this developed architecture separates the image into numerous pieces that represent enclosed frames, coupled upon class likelihood marks. This method delivers tremendous increases in regards of pace meantime sacrificing efficiency. This detector module has strong generalization abilities, allowing it to display a complete image [51].

To distinguish persons from background, another suggested scheme engages the YOLOv3 including the DeepSort technique for locating recognized public through enclosed frames plus assigned ID number. With respect to FPS, mAP and loss values defined by localization and object categorization, the results are confronted to those of other models such as for example SSD and faster R-CNN [45].

In this paper, to accurately identify human the author offers deep learning focused object detection model that is used in conjunction along a thermal image social distancing classification technique. For the sake of recognizing, at the same time tracking people at home and outside environments, an innovative deep learning detection technique is formed using the YOLOv2. This method is used for the creation of a comprehensive AI system for tracking people, social distancing categorization, and monitoring body heat by using images of thermal camera [49].

The basic review and differentiation of object detection methods is presented in a study, which incorporates two classes of object detectors formed on time, accurateness and parameter values with varied input image sizes. Fast R-CNN, R-CNN and Faster R-CNN are covered in two level detectors, while SSD, YOLO v1, v2, v3 in one level detectors. The results of the comparison reveal that YOLOv3 tiny improves object detection speed as well as maintains accuracy. Object recognition and localization can also be extended from static images to a movie including a dynamic stream of images [34].

A research paper indicates a system that may be used to observe public venues such as ATMs, malls, and hospitals for any kind of breach of social distancing. For the sake of measuring distance between persons in the frame, the simulated model employs deep learning methods with the OpenCV library, as well as a YOLO model trained on the COCO dataset to recognize people in the frame [64].

An improved social distance monitoring system was proposed. For people detection, tracking, and Euclidean measurement, they employed the YOLO method. The system investigates whether or not social distance standards are being followed. If people abide by the rules, they will be bound in a green anchor box in the frame, while those who do not, will be bound in a red anchor box, and a laser beam will be emitted on that person, along with a siren buzzing to alert them. If the system discovers a crowd (more than a certain number of people), it will issue a warning to maintain social distance. It will send an alert SMS to authorized persons after the fifth announcement, asking them to intercede in order to tackle this crucial problem [79].

For recognizing social separating, the research portrays a strategy utilizing deep learning on how to survey the distance between individuals with a view to diminishing the impact of the COVID plague. The detecting instrument was made by breaking down a video feed to caution individuals to stay away from another. Based on the YOLOv3 technique the open-source object detection pre-trained model was utilized to distinguish passer-by by utilizing the video outline from the camera as input [40]. A model for a task associate was proposed in another research in view of a deep learning neural network. Here, for perceiving a portion of the constituent pieces of a vehicle YOLOv5 is utilized. After trying YOLOv5s and then YOLOv5m, it was very clearly resolved that for this kind of identification issue, YOLOv5s can be more than adequate [60].

A real-time surveillance framework was proposed by a group of authors that would extract a video stream as an input and not only assess if individuals seen in the video are wearing a face mask. But also, this exploration screens people for social distance detection. By utilizing YOLOv5, the proposed procedure takes input data from a CCTV feed and distinguishes people in the edge, while DBSCAN is utilized to detect the proximities among the noticed individuals, these found appearances are then arranged utilizing Stacked ResNet-50 to decide if the individual is wearing a mask or not [65].

This project represents a technique for the detection of social distancing in workplaces or other encased spaces. This paper proposes a Deep Neural Network (DNN) model based on YOLOv5 for making the method of monitoring and checking social distancing in an indoor environment via object detection. They perceive office objects of familiar measurements and employ bounding frame boxes in indoor circumstances to regulate social distance continuously. The enclosed boxes of persons and special things will be given by the Object detector [69].

Author put forward a sidewalk disturbance recognizable model that can success-

fully identify various sorts of Pavement upsets under various circumstances. They upgraded the YOLOv5 model and acquainted attention mechanisms so that it can improve the strength of the model. The model works better at object detection in contrast with any other method [74].

Latest findings have shown that a person's face, as well as their walking style, can be used to identify them through video surveillance cameras. Nevertheless, the technique for distinguishing a person in a congested circle is challenging to escalate [7].

In an approach, A method was developed with a low-resolution camera for finding people on foot using background reduction and real-time classification of foreground silhouettes. In a different technique [39], the moving object is retrieved using GPU-based GMM (Gaussian Mixture Model) background subtraction. Then, two complementary features for moving object classification, one is region-based description that is Histogram of Oriented Gradient (HOG), and the another one is contour-based description which is Fourier Descriptor (FD) are extracted. Then the two of these descriptors will be integrated victoriously into SVM, allowing for enhanced execution [10].

Using TensorFlow, Roth constructed a strong social distance detector for pinning down humans and proctoring social distance [48]. The author created a novel appearance-based object detection system based on artificial intelligence technology and used TensorFlow for implementation. Basically, to portray an object's visual appearance as a loosely structured collection of local context regions linked by discrete key traits, or fragments is the fundamental objective. This technique is compelling at detecting several objects [29].

The motive of this research was to look into modern open source-based technologies for object detection in sports, specifically football players. An SSD along with the MobileNet-model was trained and tested using TensorFlow's API [25]. With a dataset comprising images gathered from video footage of two football matches, the model was tested as a) pre-trained and b) with fine-tuning. Another suggested method focuses on integrating and optimizing human identifying algorithms in real-world applications on embedded platforms like Nano, NVIDIA Jetson TX, as well as automatic weighing up of current person detecting models in the matter of not only perfection but also execution, and datasets which are applicable for distinguishing along with marking people inside the building. In order to train and assess, deep learning models operate on datasets that are general purpose such as COCO, PASCAL VOC and ILSVRC [63].

In this research, a system was established for pointing out an object using MobileNet for SSD. They integrate MobileNet and SSD frameworks to create a deep learning-based technique that is both swift and captivating. Using SSD detector and MobileNet, a high-accuracy object identification approach was pulled off, accelerating the performance speed up to 14 fps, thus making this suitable for any cameras which only work at 6 fps [54].

Chapter 3

Research Methodology

The proposed Image Processing Based Human Detection and Social Distancing Measurements with Monitoring via Fine-Tuned Deep Learning and Computer Vision is intended to identify those in the crowd who are failing to maintain social distance. People detection, tracking, and inter-distance calculation are all part of a three stage approach for the purpose of monitoring social distances and performing zone-based infection risk assessments. Due to the system's real-time capability, it can be used with a wide range of CCTV security cameras, from VGA to Full-HD [47].

For our first model we chose YOLOv5. Based on YOLOv4, YOLOv5 is the most current version of the recent YOLO development. With the YOLOv5 method, the system is trying to find people who are breaking health rules by socially separating themselves from others, and then tell them to stay away from other people intending to stop the extension of the COVID-19 virus. The processing speed of YOLOv5's has increased dramatically, reaching 140 frames per second at it's fastest. Even though it's 90 percent smaller than the previous version, it's still very compact [70].

YOLOv5 comes with a built-in dataset that has been pre-formation with the COCO (Common Objects in Context) dataset. This COCO is basically a dataset that helps people recognize, segment, and label objects. In this dataset, there are more than 200,000 images that have been labeled with 80 different types of things, including the human class. So, YOLOv5 can be used to detect social distance in situations where the main system first finds people. It doesn't matter which dataset you use because the COCO dataset is the default one for YOLOv5 [70].

Furthermore, The YOLOv5s advanced algorithm was used because the data loader is used to send each batch of training data through the data loader and to improve the data at the same time. It can perform three types of data improvements: color space correction, scaling and mosaic enhancement. To separate all input images, this model employs the $S \times S$ grid system. Object detection falls under the purview of each grid. Grid cells currently predict the boundary boxes of the detected object. These five key characteristics are present in each box: coordinates, object width and height, as well as the box's confidence score. Furthermore, when compared to YOLOv3, YOLOv5 is faster and more accurate. Another advantage of using YOLOv5 for object detection is its faster processing time than YOLOv3 [76]. CSPResNext50, CSPDarknet53, and EfficientNet-B3 served as the model's

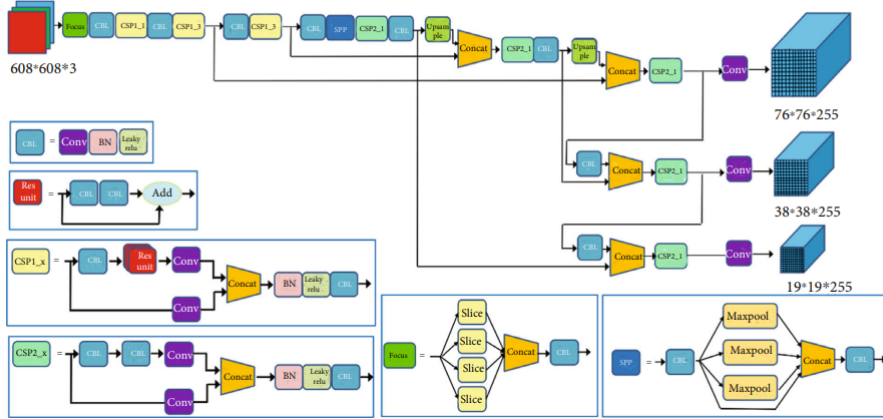


Figure 3.1: Structures of YOLOv5s [31], [73]

backbone for the YOLOv5 object detector. Both the CSPResNext50 and the CSP-Darknet53 make use of DenseNet. The mosaic data augmentation, which tiles four images together, improved the accuracy of finding smaller objects in YOLOv5. The YOLOv5 was trained on the video using pre-trained YOLOv5 weights and a scale factor of 0.00392 with a spatial size of 416×416 . The algorithm detects people in videos and uses four coordinates to pinpoint their location [78]. As part of YOLOv5, new data enrichment methods, including SAT, mosaic training, and multi-channel feature replacement of FPN fusion with PANet, have been introduced [53], [67].

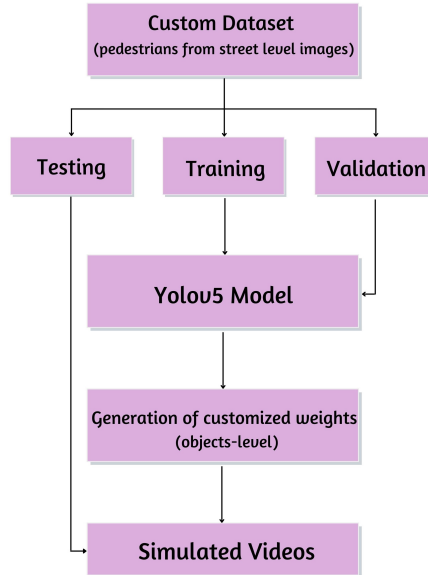


Figure 3.2: Human detection workflow using YOLOv5

YOLOv5 has been implemented using the Matplotlib, Torch, Torch vision, cv2 library, which are more user-friendly for developers. YOLOv5's end employs the same Mosaic data enhancement method as its output end. Arithmetic progression with a small target is used in the usual project training. Typically, less ambitious than the medium and large objectives. Our set of data also contains a load of small targets,

but the distribution of these targets is uneven, which makes it more difficult to analyze [55]. Random scaling and splicing add loads of small targets which then makes the network stronger, and there are many other advantages, such as a rich data set, random scaling, and random distribution. In order to reduce the number of GPUs, some may argue that random scaling and standard data enhancement can be used as well. As a result, the size of the mini-batch does not have to be prohibitively large, as the author anticipates that many people will only be able to use a GPU. As a result, a GPU will produce better results [55].

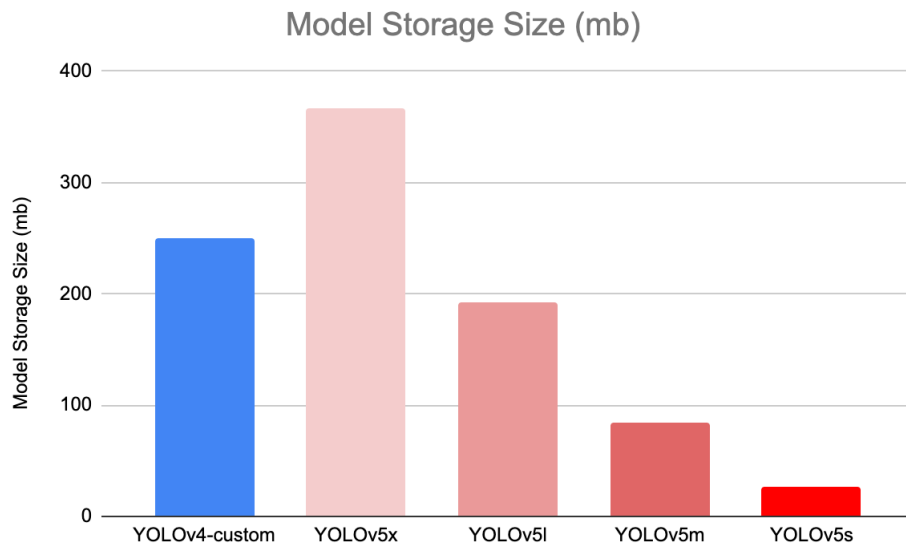


Figure 3.3: YOLOv5 Performance

In this figure, we see YOLOv5s require very small space or storage. Whereas, YOLOv4, YOLOv5x and the other variants of YOLOv5 are taking more storage space in the performance graph.

The mAP and speed of the YOLOv5 series models are shown in Figure 3.4. The mAP and processing time increase as the weight increases. For mAP, YOLOv5x has the best results, at 86.55 percent, which is 3.9 percent higher than the lowest. On the other hand, the YOLOv5s takes the shortest time to process an image on GPU, 19.14 ms, which is over half the time of the YOLOv5x. Another calculation shows that the FPS of YOLOv5s is 52, which is sufficient for real-time processing.

In contrast, the mAP of the YOLOv5x and YOLOv5l are nearly identical, with the YOLOv5l being faster as figure 3.4 is representing. The same thing happens with YOLOv5s and YOLOv5m. It is more practical to use the speedier option with comparable correctness [66].

The figure 3.5 depicts the identification of those individuals who failed to maintain an appropriate social distance. The first step was to mark the people in danger with red points, and then to identify them using image processing. The people who are maintaining minimum threshold are at low risk and yellow point defines them.

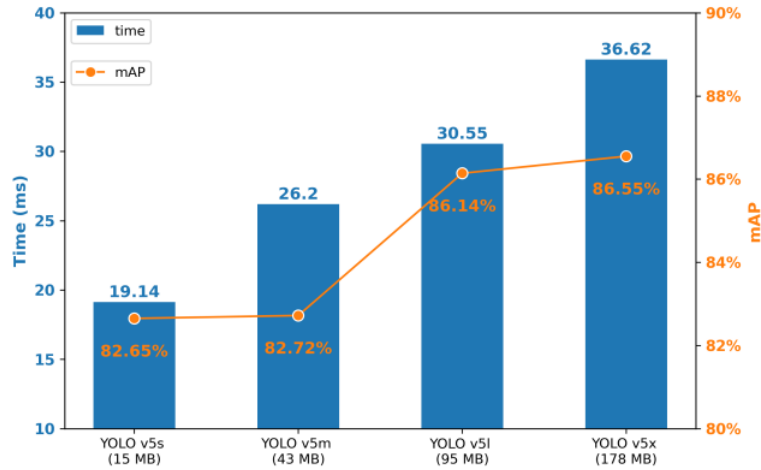


Figure 3.4: YOLOv5 Performance [66]

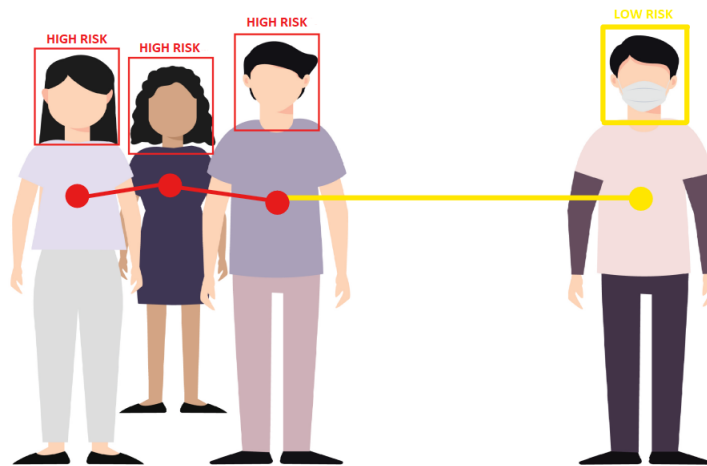


Figure 3.5: Identifying through image processing

For our second model here, we have used TensorFlow, which is more than just a machine intelligence framework. TensorFlow is an open-source model which is developed by Google for machine learning and deep learning. There are legions of powerful algorithms grouped under these two headings, all aiming to solve the same problem: teaching a computer to recognize patterns on its own and/or to make the best decisions possible. We chose to work with this model because of it can effectively handles multi-dimensional arrays in mathematical expressions, here deep neural networks and machine-learning concepts are well-supported, the same code can be run on both a GPU and a CPU and it has extremely large data sets and machine scalability.

Using a unified dataflow graph, TensorFlow depicts how an algorithm works as well as how it performs. A combination of large dataflow systems and record-low parameter servers serves as inspiration for our work. Traditional dataflow systems, on the other hand, only allow graph apogee to depict functional computations on

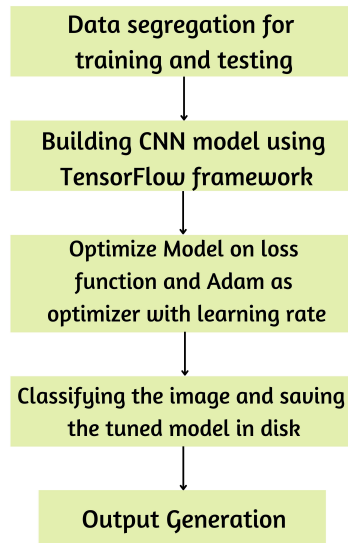


Figure 3.6: Generalized methodology for image classification TensorFlow framework

rigid data. Vertices in TensorFlow can depict computations that update or change their state. Using Tensors which are basically multi-dimensional arrays, TensorFlow makes it possible to communicate between distributed sub-computing nodes. Programmers could indeed experiment with various parallelization strategies using TensorFlow’s single programming model, such as offloading computation to servers that hold common state. Contrary to popular belief, large-scale learning does not necessitate asynchronous replication. Synchronous replication has yielded promising results for our coordination protocols [14].

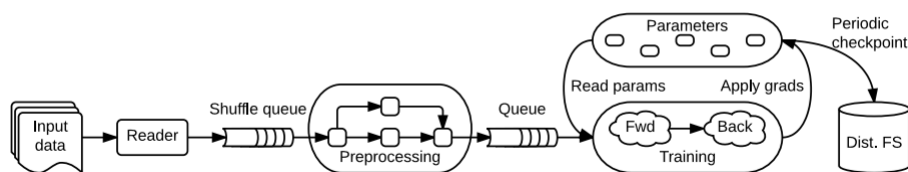


Figure 3.7: A schematic TensorFlow dataflow graph for a training pipeline, containing subgraphs for reading input data, training, checkpointing state and preprocessing [32]

Multiple sub-graphs with shared variables and queues are depicted in the figure above, which is representative of a typical training application. An input batch queue and model parameters are the foundation of the main training subgraph. The training subgraph is used to update the model based on multiple batches of input data in order to implement data-parallel training. Preprocessing transforms individual input records while the I/O subgraph deciphers records from a categorized file system to fill the queue (image decoding and referring random distortions). A

more efficient specialization [13]. Adding quantization support for mobile devices and high-throughput datacenter applications, as well as, to accelerate quantized computation, using the gemmlowp low-precision matrix multiplication library has been a major focus of TensorFlow research work this year [13].

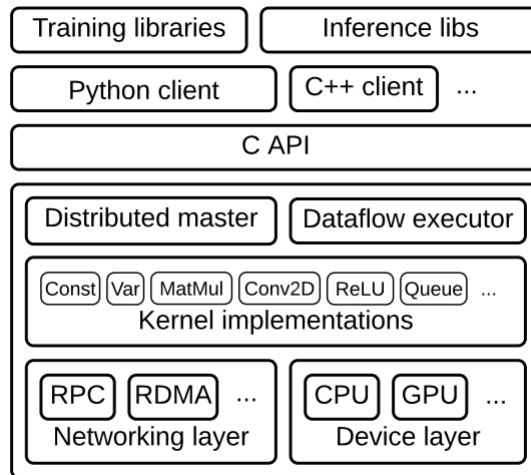


Figure 3.9: The layered TensorFlow architecture[32]

For the installation of this model, we have used some steps. First installed the TensorFlow PIP package. Then we had to verify our installation and installation of CUDA Toolkit, CUDNN was done. It is more efficient and convenient to use the Object detection API for training models and improving performance NVIDIA GPU, CUDA Toolkit v11.2 cuDNN 8.1.0 had to be met in order for TensorFlow to run on our GPU. There had to be an environment set up. Then we tried it out on a new terminal to make sure everything was working. The TensorFlow installation is now complete. After that, it's time to set up the TensorFlow Object Detection API on your system. There was a download of TensorFlow Model Garden. Model and training parameters are configured with Protobuf in the TensorFlow Object Detection API. The Protobuf library must be downloaded and compiled prior to utilizing the framework. For the new values of the environment variables to take effect, we had to start a new terminal. In addition, the COCO API was installed. The Object Detection API can be installed by downloading and installing the object detection software. From within TensorFlow models/research, run the following commands. It appears that our installation is working as expected.

As of figure 3.10, TensorFlow by default maps nearly all of the GPU memory of all GPUs visible to the process (subject to CUDA_VISIBLE_DEVICES). This is done to make better use of the devices relatively precious GPU memory resources by reducing memory fragmentation. We have used the tf command to restrict TensorFlow to a specific set of GPUs.

The TensorFlow Model Garden and Google's Special Interest Group on Machine Learning Models (SIGMODELS) have teamed up to create exemplary implementations of well-known machine learning models in public spaces (TFMG). In order to

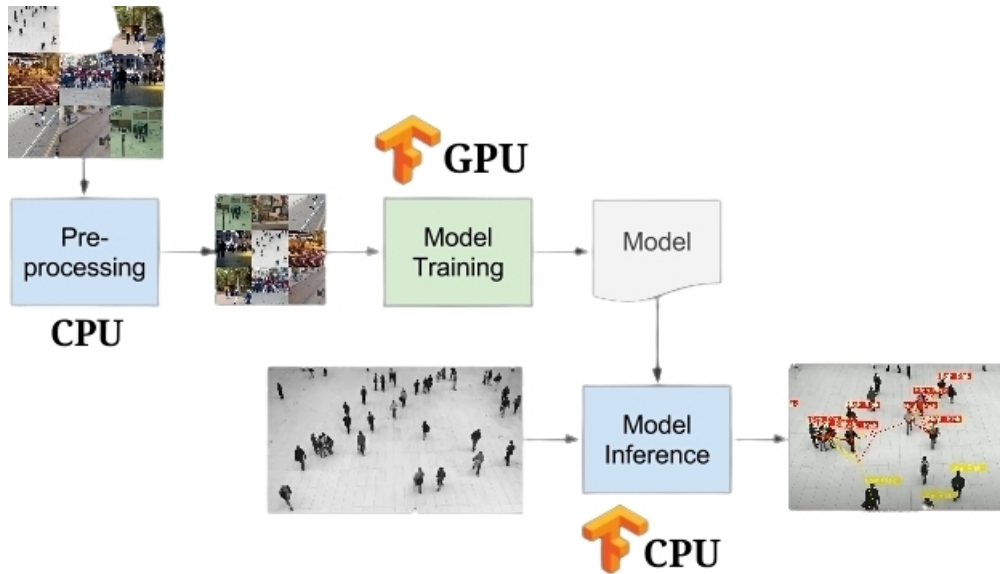


Figure 3.10: Use of CPUs and GPUs in different stages of TensorFlow

address the issue of machine learning code reproducibility, the Model Garden serves as a repository of Exemplary Machine Learning Models. TensorFlow Model Garden, TensorFlow Hub Torchvision, and some models advertised on Kaggle are some of the early attempts to develop exemplar collections in the machine learning community [37]. Pre-trained models and datasets from across the TensorFlow ecosystem can be found in this repository. Computer vision datasets, models and common image transformations can be found in Torchvision (integrated into Pytorch). User-created trained models can be uploaded to Kaggle and shared with the community for re-use. It is a repository of machine learning models and datasets built with TensorFlow's high-level API. This is similar to these efforts [65].

In spite of the fact that TensorFlow can be installed and utilized without Anaconda, we choose this for the reason of the ease with which it allows us to manage packages and set up new virtual machines. Before the version 20.3, pip incorporated or installed a package and all dependent Python packages without checking for conflicts. It would install a package and any dependencies, despite the ongoing situation. A different version of NumPy library is required to install a different package than TensorFlow via pip, an existing installation of TensorFlow could become inoperable. When the package appears to be working, the results can vary. Because of this, the conda package manager has historically been distinguished from pip. There are many ways to share custom conda packages, including Anaconda Cloud, PyPI, and other repositories. Python 2.7 and Python 3.7 are included in Anaconda2 and Anaconda3, respectively. It is possible, however, to create new environments using conda-packaged versions of Python. We had to open a new terminal window and type the command to install Anaconda. An Anaconda virtualization environment became necessary in the later stages [17].

Chapter 4

Data and Preliminary Analysis

We have described in total a three step approach that incorporates the human detection, tracking and their inter distance measurements as a whole for social distance monitoring and zone based infection risk assessments. This system can be racially balanced and deployed to different models of security surveillance CCTV cameras with any form of resolution from VGA to HD approaching real time.

4.1 Human Detection and Social Distancing Measurements

The social distance measurements can be utilized using numerous models. The model's can be used for detecting humans in the particular frame, along with their unique localisation bounding centroids at different frames. Human detection is segmented mainly in three sections. It includes an input module as well as other processes such as expansion, a foundation for feature extraction, and a target for forecasting object classes and their positions on the output [42]. TensorFlow object detection API makes three predictions for each distant position for an image at different calibrations, the problem of not being able to detect the low resolution objects will be removed effectively. Objectness, boundary box regressor and classification scores are computed for each prediction [68]. The object detection API for TensorFlow is a free and open source framework. The object identification infrastructure can be easily developed, trained, and located thanks to the use of TensorFlow. In this API model there is already included pre-trained libraries that are known as Model Zoo. These pre-trained libraries are trained on various datasets like, COCO dataset and Open Images dataset.

We have implemented the YOLOv5 model using OpenCV, Computer vision and Deep learning. To detect items, YOLO uses deep learning to merge the identified object's parts into one neural network. Images are divided into $S \times S$ pieces or grids using the YOLO method's technology [71]. The grid cell is able to detect the object if the object's center lies within one of its cells. Each grid cell predicts the pedestrian locations. Model assurance and accuracy in the object's container are both represented by this value. The value of the pedestrians are calculated using the Euclidean distance equation. Based on the YOLO development, YOLOv5 is the

most recent. YOLOv5 has been improved to great extent with the fastest speed of hitting 140 FPS. YOLOv5 is so compact compared to YOLOv3 and YOLOv4 that it can be also implemented on any embedded system. In terms of accuracy, it outperforms the competition and has a significantly greater capacity for detecting little items from frames [71]. First, we have to determine the source through which we will be giving the input to our system. Then we will establish the connection with python script through opt which is the directory where we install our unbundled packages. Then in the procedure comes model selection. We have to upload the model for using the dataset in the form of weights. Then generalizing the data as what data should be imputed we have to set the input in the dataset. Then it requires augmentation of each class and is agnostic for determining the true coordinates. Then after determining the people in every frame/image using a file writer, we will be able to determine the pedestrian points of people.

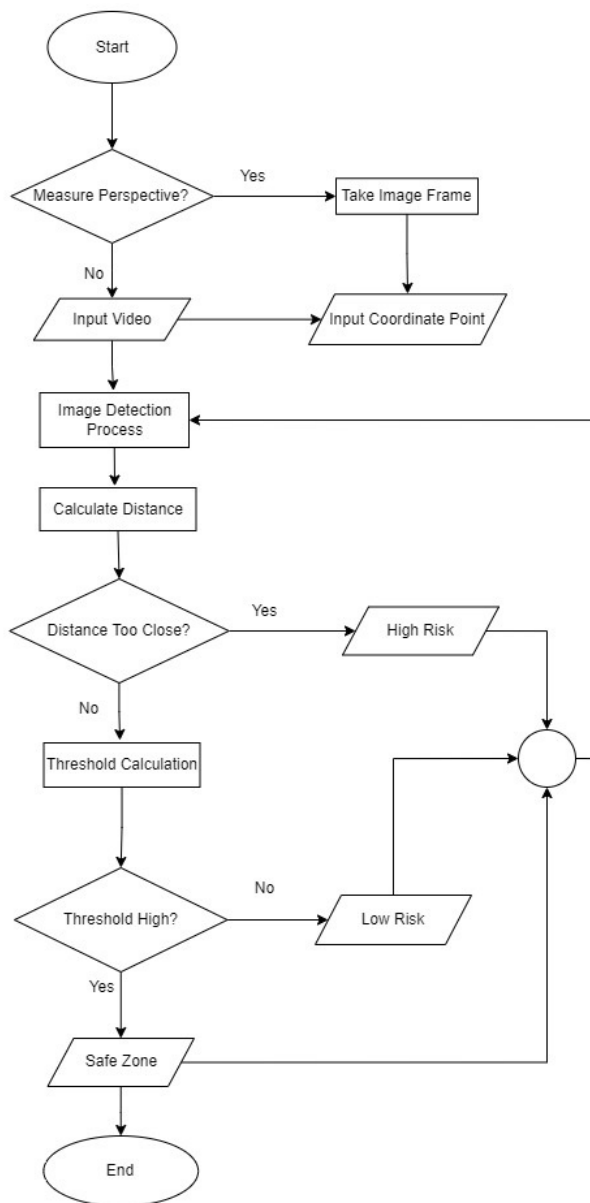


Figure 4.1: The flowchart of the working procedure of YOLOv5 model

In the figure 4.1, we can see the working procedure of the YOLOv5 model in a form of flowchart. The model starts with measuring the perspectives and with positive results it takes the image frame, otherwise it inputs the video directly. After taking the image frame it takes coordinate points and then starts image processing. After that the model starts calculating the distance between the pedestrian points and if the distance between the points are too close it will show High Risk and if it is not too close it will calculate the threshold value and then it will determine whether it is high or low. If yes it will show Low Risk otherwise it will show a green pedestrian point that indicates a safe zone. Finally, all the outputs will loop around the same process until the video or stream comes to an end. This model will also determine the distance of the pedestrian using the Euclidean distance equation to measure any human distance using two points. Euclidean distance is defined as the measurement of distance between two locations in Euclidean space in order to determine the relationship between angles and distances. The Euclidean distance is the most commonly utilized distance between two points x and y in d -dimensional space. Making two-dimensional calculations at the coordinates of (x_1, y_1) and (x_2, y_2) . Finally, each and every frame will be synced together to make an AVI file using a video writer.

For the implementation of our second model we have used both the TensorFlow model and SSD Mobilenet using Deep learning and Computer vision. TensorFlow model is a powerful and configurable machine learning system with a lot of capability. Through this paper we aim to provide the fundamental framework for human detection and measuring distances among them using the TensorFlow model. Open source numerical computation and large-scale machine learning libraries, such as TensorFlow, are the major focus of this project. It is a collection of models and algorithms that integrate machine learning and deep learning techniques. Using TensorFlow, developers may create a graph of computation [15]. Nodes in the network represent mathematical terms, whereas connections in the network represent data. Due to this, the developer may need to invest more effort into designing the application's logic as a whole rather than worrying about details like how to connect the output of one function to its input in another action [29]. Therefore, in order to apply the TensorFlow model for recognizing humans in the frames, we must first load the model from a file into a TensorFlow graph and then describe the output that we require. The graph that stores the model must be passed through for each and every frame or image in order to produce the desired outcome. Additionally, filtering out weak predictions or objects we don't wish to find is an important part of the process [48]. The first phase in this model's detection of people in the frame is model selection. The COCO API dataset is used to train all of the TensorFlow object detection model zoo's models for model selection. This dataset has 120,000 photos with a total of 880,000 identified items. The step will be creating a situation which will be the entity responsible for the execution of the operations defined in the graph. This model all require the installation of CUDA Toolkit v11.2 and cuDNN 8.1.0 that will provide high tuned implementations for standard routines which include normalization, activation layers and forward and backward convolution. cuDNN is CUDA's One of the most powerful GPU-accelerated neural network libraries in the world, used by both researchers and framework developers. There are many other devices which have low level GPU performance which might not carry out the desired framework that this high end GPU accelerator can. cuDNN accel-

erates Chainer, Keras, MATLAB, MxNet, PyTorch and TensorFlow. TensorFlow object detection API is also needed to train, create, and deploy the object detection models. We also need a TensorFlow model garden to aim at demonstrating the effective practices of modeling so that it is efficient for us as it provides TensorFlow users a centralized place to find examples for models and reusing libraries. Protobuf compilation is used again in this TensorFlow object detection API to construct modeling and training libraries. It is one of the general forms of storing data that can be transported later efficiently, as it compacts the data and enforces the structure on data. It is used to generate the graph between the pedestrians.

The next process that is starting of every frame that will require processing. The model's desired inputs and our desired model outputs will be included in this section. This will basically do the work of passing every frame or image through the model. Now, after calling the function on each frame we will initialize the predictions and detections and return all the transformed points that will detect humans in the image. Again, we have to also set the human coordinations, images and also the distance threshold, where we will use variables for simulating the formula for measuring the distance between the centroids which are known as pedestrian points. Now, for determining the distance first we have to generate the center for the pedestrian. We have used the midpoint equation to find the center of the pedestrian,

$$C(X, Y) = \left(\frac{X_{min} + X_{max}}{2}, \frac{Y_{min} + Y_{max}}{2} \right) \quad (4.1)$$

From the equation, these values or outcomes will be generated for each of xmin and xmax, height and width, and ymin and ymax to determine the area of focus for this pedestrian. Now, when we are done identifying the pedestrian points, we need to measure the distance between the pedestrian points so that we can identify whether the people are at safe distance or not, are they maintaining social distance or not. This mathematical formula for determining the separation in physical distance between any two objects in a frame,

$$d(c1, c2) = \sqrt{(X_{max} - X_{min})^2 + (Y_{max} - Y_{min})^2} \quad (4.2)$$

Through this equation now we will identify the distance and label people regarding the identification of Low Risk and High Risk. After generating the label we have to generate the output as an AVI file. Finally, this output will be also generated in a CSV file which will show the Frame ID, Date and Time, Total People, People at Low risk and People at High Risk.

4.2 Model Dataset Pre-Trained Libraries

4.2.1 Anaconda in Python

In the past decade, Python has become one of the most popular programming languages for scientific research. Its open-source nature and large online community are among the factors that have made it so popular. Many fields, including data analytics, artificial intelligence, and scientific research, have seen improved productivity as a result of Python's use. Python's use in the field of data science is gaining more and more traction. Python was deemed to be the most popular programming language among those that were surveyed for Anaconda's 2021 State of Data Science study. 63 percent claimed they use Python either frequently or always, making it the most popular language overall. A toolset for researchers and scientists is provided by Anaconda, a free and open-source software package. Anaconda provides access to a variety of Python/R environments with a single package install. Code development is made much easier using these platforms or apps, commonly referred to as integrated development environments (IDEs). They provide a similar function to word processors like Microsoft Word, Google Docs, and Pages, but in reality, they are much more. One may use IDEs for a wide range of tasks such as code-reviewing, data-visualization, data-inspection and variable storage. They can also be used for collaborative projects. The programming language is the same regardless of the IDE's presentation and peculiarities. This means that changing the IDE in Anaconda does not have a major impact on your Python code. The most difficult part of getting started with Python is getting head around the syntax. It is easy to transfer coding skills from one IDE to another. One IDE is not inherently superior to another, each has its own benefits and downsides in [17]. A distinct Python version is required for each type of application. Because of a reliance that is available in prior versions but has changed in current ones, the program must execute on a certain version of the language. When it comes to separating apps, virtual environments are perfect. We can quickly and simply move between the two programs by using a virtual environment. The Python community has contributed a large number of prebuilt functions, which you may use by downloading the toolkit. Libraries containing these functionalities may be simply acquired using Anaconda. Python may be installed and used in a variety of ways, but Anaconda's graphical user interface (GUI) is straightforward, well-supported, and contains the most significant libraries and IDEs as part of the installation. Keeping all of these libraries up-to-date is made much easier with Anaconda. This means that instead of having to install many IDEs, libraries, and functions for Python individually, Anaconda can accomplish it all [17]. When it comes to Python data science and machine learning, Anaconda is the easiest approach to execute. Open-source packages and libraries may be accessed using this toolset, which is designed for solo practitioners. As a starting point for both new and more experienced Python programmers, this is the ideal way to get started with the language.

4.2.2 NVIDIA GPU

The Graphics Processing Unit, often known as a GPU, functions as a co-processor to speed up the performance of the CPU. Loading some of the most time-consuming

and complicated tasks onto the GPU (Graphics Processing Unit) speeds up the CPU (Central Processing Unit) based applications [3]. NVIDIA has introduced a new computing platform for its graphics processing units called CUDA. This platform is designed for general purpose computing (GPU). The effectiveness of using this platform for statistical machine learning applications is analyzed in this research. Both the transfer rates to and from the GPU and the performance of matrix vector operations on the GPU are measured. A description and evaluation of a GPU-based implementation of a sparse matrix vector product are presented here [5]. The initial component of NVIDIA CUDA is an integral image method that is based on prefix sum. The artificial intelligence that is accelerated by the Tensor Cores that are integrated into NVIDIA's Turing GPUs is, in turn, being utilized to speed up games. GPUs have a number of opportunities for growth in the automobile sector. They feature picture recognition skills that are unparalleled, just as anyone would expect [5].

4.2.3 CUDA

In the Compute Unified Device Architecture (CUDA), the GPU may be used as a data-parallel device without the requirement to transfer calculations to a graphics API. CUDA is a hardware and software architecture. CUDA changes the hardware's characteristics from a graphics card to a multi-threaded coprocessor. BLAS and FFT implementations are included in CUDA, however NVIDIA only offers a C/C++ API for them [4]. That portion of the program running on the GPU is handled by CUDA, which utilizes hundreds or thousands of threads. These threads are laid out in the form of a grid with blocks. One, two or three-dimensional threads can be used in the grid, and each block can be either one, two, or three-dimensional threads.

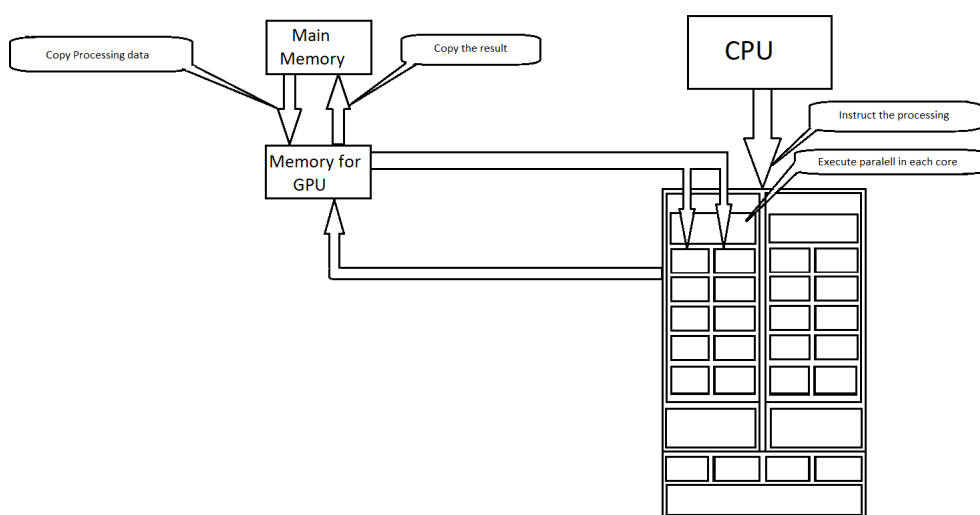


Figure 4.2: CUDA WorkFlow

It is possible to set the grid and block dimensions at runtime, with each thread

being able to get its own thread and block ids. NVIDIA hardware will only provide synchronization and access to fast shared memory for threads that are part of the same block when several blocks of threads are being performed on a single physical SM.

The figure 4.2 shows how the CPU is responsible for its primary execution. When a kernel call is made, the program will continue executing on the CPU using a function that is not part of the kernel. At the same time, the execution of the kernel function takes place on the GPU. This allows us to perform processing in parallel on both the CPU and GPU. Another name for this technique is “heterogeneous programming”. Memory transfer between the host and the device is the key bottleneck in the execution of applications. Both will have their execution delayed until this procedure is fully finished.

The shared memory paradigm is utilized by the CUDA architectural framework. Allows for memory transfers between the Host and the Device Memory operations can be kept on the chip by using local shared memory and registers. Various elements of the graphics accelerator’s memory are partitioned. Operation of the Local Thread is carried out in the Local Memory or Register. Communication between threads takes place in the shared memory. The blocks and grids communicate with each other via the global memory.

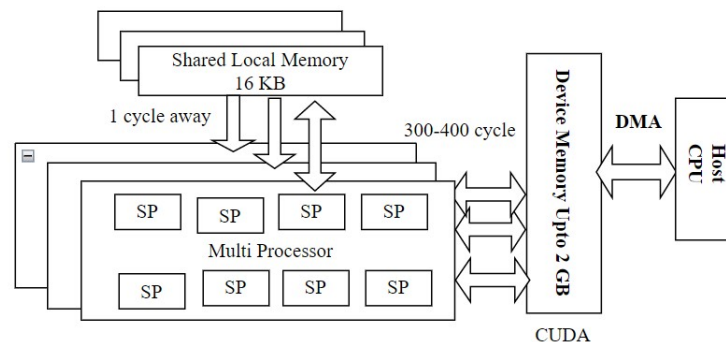


Figure 4.3: CUDA Architecture

Using a graphics processing unit as a general purpose computer allows for simultaneous processing on low-cost hardware. Programmers can use NVIDIA’s Compute Unified Device Architecture (CUDA) to develop data-parallel applications that run on the GPU. The CUDA paradigm considers a GPU to be a coprocessor capable of handling several tasks at once. It is possible to offload a kernel, a data-parallel compute process, to the GPU for execution. Simultaneous execution is referred to as SIMT, and it involves hundreds of threads running the same code on various data sets.

Each thread may identify its spatial position using its thread ID and thread block ID, and can thus access the data associated with that place. Multiprocessors (14-

30 in recent versions) are housed in a single GPU chip in the CUDA architecture. 8 or more stream processors are used to run up to 1024 concurrent threads on each multiprocessor, all of which are running the same code. Each thread in a multiprocessor may access 16KB of shared memory with a 1 clock cycle access delay. All multiprocessors have access to a global memory that ranges in size from 256 MB to 1 GB and has a greater access latency. This memory is known as device memory (300-400 cycles). The cost of accessing global memory can be amortized by combining accesses from many threads. It is also feasible to send huge blocks of data from the main memory to the device memory using DMA [26].

4.2.4 cuDNN

The NVIDIA CUDA Deep Neural Network library, also known as cuDNN, is a GPU-accelerated collection of fundamental building blocks for use in deep neural networks. All the typical convolutional and activation techniques are well-implemented in cuDNN thanks to its high-quality implementations. cuDNN is the ready GPU accelerator for deep learning researchers and framework developers across the world. Rather of wasting their time tinkering with low-level GPU performance, they may focus on training neural networks and designing software applications instead. A wide range of deep learning frameworks, such as Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle and PyTorch can benefit from cuDNN's performance improvements [19]. cuDNN's APIs are designed to be open to the entire community of neural network frameworks. This resulted in us coming up with a design that is framework agnostic, which means that users of cuDNN are not forced to adopt any certain software framework, methodology, or even data architecture in order to utilize it.

The cuDNN interface is meant to be independent of neural network topology. Data about threads and CUDA run-time management are kept in modest amounts of global state. Datatypes like filters and tensors are also represented as opaque structures. It's not that these objects don't have a purpose; they're just lightweight representations of data held by the client program, not by the library itself. API methods that conduct basic operations on user-controlled buffers make up most of the API's content. With a basic object model and stateless API, the library easily interacts with other frameworks. All of cuDNN's algorithms enable single- and double-precision forward and backward propagation. Convolution, pooling, and neuron activation functions are all examples of these. Variable data layout and strides are supported, as is indexing of input picture sub-sections. An additional collection of 4d-tensor transformation procedures makes it easier to manipulate 4d-tensors [30].

4.2.5 TensorFlow Object Detection API

In computer vision, the phrase object detection refers to approaches for finding and classifying things. Static and moving photos may both be used for object identification algorithms. These pre-trained object detection models are utilized with open-source libraries like OpenCV's DNN library and TensorFlow Object Detection API to detect a wide range of objects, from humans to television monitors [25]. A

solution to issues with object detection can be found in the package that is provided by the TensorFlow Object Detection API. Real-time object detection in a picture is now possible using this method. Out-of-the-box TensorFlow object recognition structures include the SSD (Single Shot Detector), Faster R-CNN (Faster Region-based Convolutional Neural Network), and R-FCN (Random Field Convolutional Network) (Region-based Fully Convolutional Networks).

In addition, these feature extractors are essential since they play a significant role in the framework's speed/achievement trade-off like MobileNet, Inception, and ResNet, among others. In reality, training a convolutional network from scratch needs a lot of time and a lot of data. TensorFlow API is used as a model for transfer learning to overcome this issue. Using a model developed for one function as the starting point for a model for another function is known as transfer learning. This method is commonly used in deep learning because of the high computational and temporal resources required to create neural network models. Instead of beginning from scratch, a pre-trained model can be used as a starting point for training the system, rather than constructing a new model from scratch [22].

The goal of the TensorFlow Object Detection API is to simplify the process of constructing, training, and deploying object detection models. It is described as an open-source framework built on top of TensorFlow. As a result of this, the TensorFlow object detection API offers the user with a variety of pre-trained object recognition models that can be fine-tuned and used in object identification tasks. A variety of pre-trained models may be utilized with TensorFlow's Object Detection API. It was decided to use an SSD model with MobileNet in this study. There are 2.5 million annotated instances in 328 000 photos in the MSCOCO Dataset, which includes 91 object classes, such as human and 91 different kinds of animals. Mean Precision (mAP) is stated to be 21 on the COCO dataset for the SSD MobileNetv1 coco-model. Both a pre-trained version of the SSD model without any fine tuning and a version that was well-tuned for our own dataset were evaluated and compared in this study [25].

4.2.6 TensorFlow Model Garden

The TensorFlow Model Garden is a repository where TensorFlow users can find different ways to implement state-of-the-art (SOTA) models and modeling solutions. TensorFlow users will be able to make the most of this functionality for their research and product development endeavors if the Model Garden is successful in demonstrating the most effective modeling techniques. TensorFlow Hub will continue to fulfill its function as a repository where users may search in an uncomplicated manner for pre-trained models that are ready for use.

4.2.7 Protobuf

Protocol buffers are much like JSON, XML, or HTML in terms of structural notation. Because of its small size and good typing, Protobuf is a popular choice. The

Protobuf GraphDef is required to define a graph in TensorFlow. It is possible to export and import this GraphDef Protobuf. A Google-created binary format known as Protobuf or Protocol Buffers, is used to serialize data across different services. A number of the most popular programming languages are already supported by this protocol, thanks to Google’s decision to make it open source.

4.2.8 COCO API

With the COCO API you can do tasks like object identification, segmentation, facial recognition, item segmentation and caption creation on a big range of images. This package offers application programming interfaces (APIs) for MATLAB, Python, and Lua that facilitate loading, processing, and displaying COCO annotations. For computer vision, the COCO dataset provides demanding, high-quality visual datasets, which are largely neural networks. For instance, COCO is frequently utilized for benchmarking algorithms in order to evaluate and compare their effectiveness in real-time object recognition.

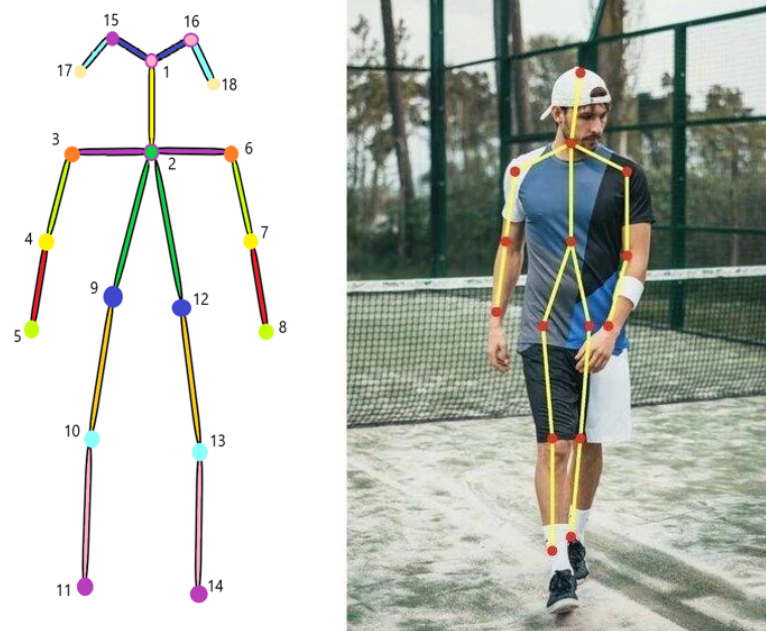


Figure 4.4: Keypoints detected by OpenPose on COCO Dataset

In the Figure 4.4 , 18 pre-trained Keypoints (classes) are annotated with three values in the COCO Keypoints (x,y,v). The coordinates are shown by the x and y values, while the visibility of the key point is indicated by the v value (visible, not visible).

4.2.9 SSD MobileNetv2

The SSD system, unlike the Faster R-CNN system, employs a single-stage detector architecture for detections. As a result, the region proposal stage is eliminated, and detection is carried out in a single iteration of the network. The initial layer of the system is a network that extracts features from the data. Standard backbone

networks, such as ResNet and VGG, are examples of what is referred to as the base network[59]. SSD is quicker than the region-based techniques since it predicts the bounding boxes and class probabilities all at once. Faster R-accuracy CNN's is also comparable. Furthermore, the feature maps' various sizes aid in producing boxes of varying dimensions, which in turn aids in detecting objects of different sizes.

Machine learning architecture models include SSD MobileNetv2 and SSD MobileNetv1. SSD MobileNetv2 is capable of simultaneously detecting a large number of items. SSD MobileNetv2 is designed for light-spec mobile devices. There are linear bottlenecks and shortcut links in SSD MobileNetv2. An example of a Convolutional Neural Network (CNN) model is the SSD MobileNetv2 algorithm. It is a branch of computer science that uses human neural networks to build artificial neural networks that can recognize and identify items. Storage Device MobileNetv2 makes use of depth- and point-wise convolution [52].

For mobile visual recognition comprising object identification and semantic segmentation, MobileNetv2 represents a considerable advancement over MobileNetv1. In addition to the TensorFlow-Slim Image Classification Library, MobileNetv2 is available in the Collaborator for immediate exploration. You may also download the notebook and use Jupyter to examine it locally. Pre-trained checkpoints and modules for MobileNetv2 may be accessible from various sources respectively.

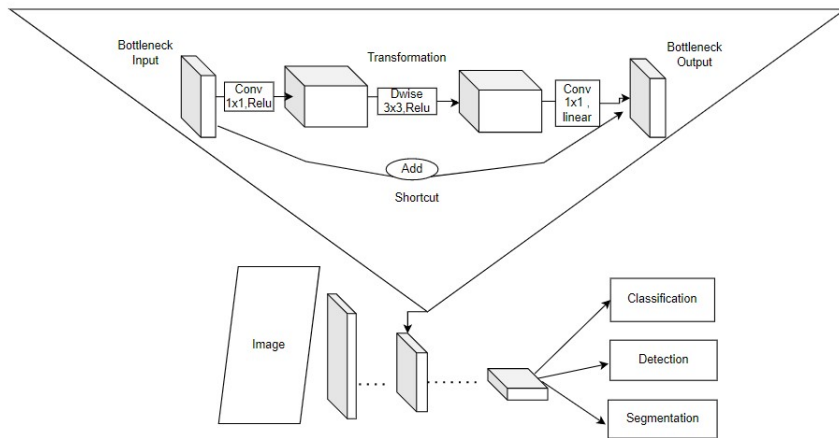


Figure 4.5: MobileNetv2 Architecture

Improved efficiency and power are achieved by making use of the MobileNetv2 in conjunction with the MobileNetv1. An entirely new design has been implemented for the depthwise separable convolution block, as seen in Figure 4.5. The novel depth-separable convolution block has three convolutional layers. Before the depthwise convolution layer, the input feature map is multiplied by an 1x1 convolution layer to increase its number of channels. The input feature map is filtered using a 3 x 3 depthwise convolution layer in the middle layer, similar to the one used in MobileNetv1. A convolutional layer with a size of 1 x 1 is used as the final layer.

Nevertheless, this last layer of convolution is used to transform data with a large number of channels into one with a much smaller number of channels, decreasing the number of channels in the input feature map. This last layer is also known as a bottleneck layer due to the fact that it reduces the quantity of data that passes across the network [6]. The model's intermediate inputs and outputs are encoded in the bottlenecks, while the inner layer embodies the model's capacity to transition from lower-level ideas like pixels to higher-level descriptors like image categories. Finally, shortcuts allow for faster training and improved accuracy, much like classical residual connections.

4.2.10 YOLO

The YOLO (You Only Look Once) object detection technique reduced the computational complexity problems associated with R-CNN by structuring the object identification problem as a single regression problem, where bounding box coordinates and class probabilities are computed simultaneously. Though it was proved that YOLO had a large performance advantage over R-CNN (for example on an Nvidia Titan-X GPU, 45 frames per second), it was also shown to have a much greater localization error than more current R-CNN variations like Faster R-CNN. Instead of predicting bounding box coordinates directly, the RPN in Faster R-CNN uses hand-picked priors to forecast offsets and confidences for the anchor boxes. Each anchor is coupled with four locations for the box and two score values that assess the likelihood of object and object not object of the proposed box [20]. So far, YOLO has been improved to five versions and recognized as one of the top object identification algorithms, incorporating several of the most original concepts from the computer vision research field. As of YOLOv5, the 5th generation, it was not produced by the original YOLO creator. In engineering, the You Only Look Once (YOLO) object identification technique of proYOLOv5 was superior. Instead of C, YOLOv5 utilizes the Python programming language. That eases the process of installing and configuring IoT devices. In addition, the PyTorch community is far larger than the Darknet community, which indicates that PyTorch will get more contributions and growth opportunities in the near future. Due to the fact that YOLOv4 and YOLOv5s are developed in two separate languages on two different frameworks, comparing their performance is challenging. It wasn't until a while later that YOLOv5 was able to establish its superiority over YOLOv4 in some situations and acquire some trust from the computer vision community [52].

4.2.11 Matplotlib

A 2D plotting and imaging software, matplotlib largely serves the needs of scientists, engineers, and financial analysts. As a Python command-line tool, matplotlib may be used directly from the command line, or as a GUI application (GTK, Wx, Tk, Windows). JPEG, PNG, PostScript, and SVG are all supported for print output. In addition to the ability to create numerous axes and figures on a single page as well as dynamic navigation and a variety of predefined line styles, symbols, photos and antialiasing, alpha blending as well as calendar and financial charts are just a few of the features. It is compatible with both numeric and NumArray expressions [2]. The

notion of the matplotlib code is broken up into three distinct sections. In the first place, MATLAB's command-line interface is the collection of functions that allow a user to construct plots. Matplotlib's frontend or API is the set of classes responsible for producing and maintaining figures, text, lines, plots and other elements of data visualization. This is a purely graphical user interface that has no concept of how it will be used. As a last step, the backends are the rendering and/or drawing devices that use the frontend representation to produce hardcopy (such as a JPG or PNG file) or an image for use on a display device (such as a Paint or GD file). Because so much of the rendering code is built in C/C++, it performs exceptionally well [2].

4.3 Input and Training Dataset

4.3.1 YOLOv5

We can use YOLOv5 for fastest object detection in real time. The speed and accuracy of collecting images are controlled by resolution of the image that we collected. In this paper, we describe a layered YOLOv5 architecture with layering. Firstly, we generate the total number of people. The stream's results are then acquired by comparing the distance by 80 pixels [77]. Using a model neck, it builds feature pyramids which let the model conclude effectively on object scaling. Pytorch, SSD MobileNet etc. served as the foundation for the YOLOv5 object detector. In YOLOv5, the mosaic data augmentation enhances the accuracy of locating tiny details by combining four images. Using pre-trained YOLOv5 weights and a scale factor of 0.00392 the YOLOv5 was trained on the video with a convolutional neural network spatial size of 416 x 416 [77]. The computer recognizes persons in the video and uses four coordinates to calculate their location. Afterwards, the data loader does three types of augmentation scaling, color space alterations, and mosaic augmentation. YOLOv5 generates three forecasts for each spatial position in an image. To keep track of each forecast, calculating objects, pedestrian points regressors and classification scores are used [77].

Weights selection

We select a model in the form of weights in YOLOv5. In this model, the very first work that we will be doing is selecting our weights or dataset. We have selected YOLOv5s as our desired weight. The YOLOv5's framework is composed of three portions. They are; backbone network, neck network and detection network. YOLOv5's speed has increased dramatically, reaching 140 frames per second at its fastest. To improve the training efficiency of this social distance analyzing model, we compressed the original video and took it frame by frame via datasets so that they meet the input channel requirements of YOLOv5s. We have also tried this model's implementation on YOLOv4 and found the result of training time being taken much more than that of YOLOv5s. Again, in comparability among all the versions of YOLOv5 such as, YOLOv5x, YOLOv5l, YOLOv5m and YOLOv5s, this YOLOv5s needs less storage which is efficient for any device or system. The next step begins with generalizing the data. Here, this term identifies the data that we

are giving as input is a what source or type of data. Following this we will also need to set the input data in the dataset. Augmenting the classes and agnostic for determining the true coordinates is also one of the essential steps for implementing this algorithm.

Labeling objects

```

class LoadStreams:
def __init__(self, sources='streams.txt', img_size=416):
    self.mode = 'images'
    self.img_size = img_size

    if os.path.isfile(sources):
        with open(sources, 'r') as f:
            sources = [x.strip() for x in f.read().splitlines() if len(x.strip())]
    else:
        sources = [sources]

    n = len(sources)
    self.imgs = [None] * n
    self.sources = sources
    for i, s in enumerate(sources):

        print('%g/%g: %s... ' % (i + 1, n, s), end='')
        cap = cv2.VideoCapture(0 if s == '0' else s)
        assert cap.isOpened(), 'Failed to open %s' % s
        w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = cap.get(cv2.CAP_PROP_FPS) % 100
        _, self.imgs[i] = cap.read()
        thread = Thread(target=self.update, args=([i, cap]), daemon=True)
        print(' success (%gx%g at %.2f FPS).' % (w, h, fps))
        thread.start()
    print('')

    s = np.stack([letterbox(x, new_shape=self.img_size)[0].shape for x in self.imgs], 0)
    self.rect = np.unique(s, axis=0).shape[0] == 1
    if not self.rect:
        print('WARNING: Different stream shapes detected. For optimal performance supply similarly-shaped streams.')

```

Figure 4.6: COCO Dataset in YOLOv5

In figure 4.6, it is shown the use of the COCO dataset in our YOLOv5 model. It helps to get our detector off the ground and for that the first work we need to do is collect training images. Next, we have to be sure that the number of objects in every class is equally distributed. Again, to train the object detector, first we need to fix its training using pedestrian points annotations. Next, splitting of the dataset is required for labeling various objects in frames. Again, to train the YOLOv5 model, we also need to add a ‘.yaml’ file to state the parameters in the dataset.

Measuring distance

Here, we are initially determining the humans and citing them with pedestrian points. Also, determining humans along with their movements. The center and radius are determined to plot the points on humans. Following this when we are doing finding the centroid values, we need to figure out the main distancing between the centroids. For finding out the centroids we need to input the people coordination, the image dataset and the required distance threshold. So, after determining these centroid values (cntr1 and cntr2), we calculate the distance between them.

Again, calculating the distance between the threshold limit we can determine the “Low Risk” when we are getting the threshold limit higher than that of actual value. We label it and call append for the value of the centroids. From this value when we

```

def plot_dots_on_people(x, img):
    thickness = -1;
    color = [0, 255, 0]
    center = ((int(x[2])+int(x[0]))//2, (int(x[3])+int(x[1]))//2)
    radius = 10
    cv2.circle(img, center, radius, color, thickness)

def distancing(people_coords, img, dist_thres_lim=(200,250)):
    already_red = dict()
    centers = []
    high_risk=[]
    low_risk=[]
    for i in people_coords:
        centers.append(((int(i[2])+int(i[0]))//2, (int(i[3])+int(i[1]))//2))
    for j in centers:
        already_red[j] = 0
    x_combs = list(itertools.combinations(people_coords,2))
    radius = 10
    thickness = 5
    for x in x_combs:
        xyxy1, xyxy2 = x[0],x[1]
        cntr1 = ((int(xyxy1[2])+int(xyxy1[0]))//2, (int(xyxy1[3])+int(xyxy1[1]))//2)
        cntr2 = ((int(xyxy2[2])+int(xyxy2[0]))//2, (int(xyxy2[3])+int(xyxy2[1]))//2)
        dist = ((cntr2[0]-cntr1[0])**2 + (cntr2[1]-cntr1[1])**2)**0.5

```

Figure 4.7: Determining pedestrian points and measuring distance

are able to calculate the center of the pedestrian point, we can easily label it and call for the next frame calculation as the position of the pedestrian is changing from frame to frame. Also, used file writer for determining the people in every frame or images and plotting pedestrians on the centroids. And when we are getting the distance less than that of our threshold limit, we are getting the warning labeled as “High Risk”. We have done it the same as done for labeling Low Risk. Lastly, in the later part of the model, we have taken all the values from the distancing class and generated all the values of Frame ID, Date and Time, Total people and People at High and Low Risk. So, the values of the model will be generated in a form of CSV in the desired location as results. We will discuss more about results in your next portion. And also using a video writer we are saving all the frames with the desired pedestrian points and the distance calculator are synced together and an AVI file is being created.

4.3.2 TensorFlow

In our second model which is TensorFlow, first we have to install Anaconda Python 3.8 32/64-bit Graphical Installer. We need to make sure of installing all the paths of the anaconda in the environment variables just to make sure that we have all the same default python distribution. Next step will be creating the anaconda virtual environment with the name TensorFlow. So, now we are ready for our TensorFlow installation process. So, for this installation first we need to install the TensorFlow pip packages and verify them by importing the TensorFlow libraries.

TensorFlow installation

In this figure, we can see the output when we are importing the TensorFlow as tf. So, this desired output verifies the TensorFlow installation. If the device we are working on is a CUDA- enabled GPU, then it is necessary to use CUDA Toolkit


```
(tensorflow) C:\Users\user>python -c "import tensorflow as tf;print(tf.reduce_sum(tf.random_normal([1000, 1000])))"
2022-05-03 21:39:25.293940: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-05-03 21:39:25.294713: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-05-03 21:39:28.918329: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library nvcuda.dll
2022-05-03 21:39:29.342879: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1733] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce 940MX computeCapability: 5.0
coreClock: 1.2415GHz coreCount: 3 deviceMemorySize: 4.00GiB deviceMemoryBandwidth: 14.30GiB/s
2022-05-03 21:39:29.344431: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-05-03 21:39:29.345796: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cublas64_11.dll'; dlerror: cublas64_11.dll not found
2022-05-03 21:39:29.346877: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cublaslt64_11.dll'; dlerror: cublaslt64_11.dll not found
2022-05-03 21:39:29.347992: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cufft64_10.dll'; dlerror: cufft64_10.dll not found
2022-05-03 21:39:29.349033: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'curand64_10.dll'; dlerror: curand64_10.dll not found
2022-05-03 21:39:29.350112: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusolver64_11.dll'; dlerror: cusolver64_11.dll not found
2022-05-03 21:39:29.351309: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusparse64_11.dll'; dlerror: cusparse64_11.dll not found
2022-05-03 21:39:29.353071: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not found
2022-05-03 21:39:29.355324: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1766] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2022-05-03 21:39:29.365077: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-05-03 21:39:29.366214: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1258] Device interconnect StreamExecutor with strength 1 edge matrix.
2022-05-03 21:39:29.366701: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1264] tf.Tensor(937.62634, shape=(), dtype=float32)
```

Figure 4.8: TensorFlow Installation

v11.2 and CuDNN 8.1.0 for better performance and execution. NVIDIA created the CUDA parallel computing platform and programming model. By utilizing the GPUs power, it allows for huge boosts in computing performance. CUDA capable GPUs contain hundreds of cores and can run thousands of computational threads at the same time. A register file and shared memory are among the shared resources available to these cores. Again, CuDNN is a deep neural network primitive library that runs on the GPU. Forward and reverse convolution can be implemented with great precision. A layer of normalizing and stimulation. Many analysts are highly interested in this model for high performance GPU acceleration. Now, after installing the CUDA toolkit the next process will be setting up the installation paths in the system variables on the environment variables of the device. TensorFlow object detection is therefore necessary for this model. This open-source framework built on top of TensorFlow is the finest option for designing, training, and deploying object identification models. Model Zoo is a framework that has a collection of pre-trained models that are used to create this model. An image/frame or video can be detected, located, and traced by using TensorFlow object detection. By detecting items, the method allows us to understand how the models function and gives us a better knowledge of the image/frame or video. Object detection with TensorFlow allows us to train machines to better comprehend human behavior and perform at their best using that information. In this phase of object detection API installation, the model requires the TensorFlow model garden. This model garden is a repository with a total number of different implementations of SOTA and various modeling solutions for users. There has to be a TensorFlow model repository in a folder named TensorFlow. This TensorFlow object detection API now requires Protobuf to configure the model and training parameters. The Protobuf libraries must be installed, setup, and compiled correctly before this TensorFlow model can be used. COCO is an image dataset that may be used for a variety of tasks like item detection, segmentation, and human detection and segmentation, and many others. Using this tool, MATLAB and Python users can load, analyze and display COCO

annotations. The COCO dataset offers demanding, high-quality visual datasets for computer vision, with the majority of the datasets containing SOTA neural networks. In real-time object identification systems, COCO is widely used to compare the performance of different algorithms. Then the object detection API needs to be installed in the folder named research inside the model folder. Now, after these steps the installation of the TensorFlow model needs to be tested.

TensorFlow model activation

Once we get the output, we can verify that our TensorFlow model activation and every library installation is successfully done. Now, we are ready to implement our code for detecting human and measuring social distancing between the human detected pedestrian points. The most important thing in the TensorFlow model is model selection where we configure SSD MobileNet dataset to the model. In order to determine an object's bounding box and category, this MobileNet SSD model employs an input picture. For mobile devices, this SSD object detection methodology may be able to provide speedy and accurate object identification. MobileNetV2 is a convolutional neural network design that is intended to be user-friendly on smartphones and other portable devices. An inverted residual structure with residual connections between bottleneck levels supports the design of this system. Also, restoring checkpoint is necessary for loading each image or frames of the input video in the MobileNet dataset.

Measure pedestrian points

```
def distancing(people_coordinations, img, distance_threshold_limit=(200,250)):
    #Plot lines connecting people
    already_red = dict() #dictionary to identify if there is high risk
    centers = []
    peopleathigh_risk = []
    peopleatlow_risk = []
    for i in people_coordinations:
        centers.append(((int(i[3])+int(i[1]))//2,(int(i[2])+int(i[0]))//2))
    for j in centers:
        already_red[j] = 0
    x_combinations = list(itertools.combinations(people_coordinations,2))
    radius = 5
    thickness = 2
    for x in x_combinations:
        xxyy1, xxyy2 = x[0],x[1]
        centroid1 = ((int(xxyy1[3])+int(xxyy1[1]))//2,(int(xxyy1[2])+int(xxyy1[0]))//2)
        centroid2 = ((int(xxyy2[3])+int(xxyy2[1]))//2,(int(xxyy2[2])+int(xxyy2[0]))//2)
        distance = ((centroid2[0]-centroid1[0])**2 + (centroid2[1]-centroid1[1])**2)**0.5

        #Identifying Low Risk in frames/images
        if distance > distance_threshold_limit[0] and distance < distance_threshold_limit[1]:
            color = (0, 255, 255)
            label = "Low Risk "
            peopleatlow_risk.append(centroid1)
            peopleatlow_risk.append(centroid2)
            cv2.line(img, centroid1, centroid2, color, thickness)
            if already_red[centroid1] == 0:
                cv2.circle(img, centroid1, radius, color, -1)
            if already_red[centroid2] == 0:
                cv2.circle(img, centroid2, radius, color, -1)
```

Figure 4.9: Determining pedestrian points and measuring distance in TensorFlow model

In this figure, we can see initially the work of plotting lines is done. Next to this, the center is identified and the variable of people at low risk and high risk is created in a form of array. Later, using the equation which we mentioned above is implemented and through this we can identify the value of centroid1 and centroid2. Finally, we

can measure the distance between the centroids and identify the value. If the value that we generated from the equation is higher than that of the threshold limit we will be getting the label “Low Risk”, or else if we get the distance less than that of the threshold value we will get “High Risk”.

While generating the output first the expansion of the model is required to determine the formation of the input. Later on, the classes, scores and boxes are defined to store the NumPy values and finally these values are generated in the CSV file. This CSV file will show the contents Frame ID, Date and Time, Total people, People at Low Risk and People at High Risk. Also, this output will generate all the images or frames into a single AVI file using the write video process.

Chapter 5

Result Analysis

While it is impossible to identify a fair characteristic of distinct object detectors, every real-world circumstance may necessitate a different approach. Understanding the other aspects that impact performance (the kind of feature extractor, the steps out of the extractor, the image resolutions, pictures, strategy coincidence and threshold as predictions) is required in order to make an educated decision about accuracy and speed. When calculating loss, it is vital to remove predictions that are not accurate or fast enough. The IOU has no maximum suppression ratio of positives, and the threshold IOU does not exist. Technology is always improving and any comparison might become out of date very fast if not done correctly.

5.1 Detecting pedestrian points and measuring distance

TensorFlow Object Detection API provides a framework including the use of pre-trained Object Detection Models such as YOLO, SSD, RCNN and Fast-RCNN etc. in our paper we used MobileNet SSD, COCO etc as our pre-trained model. It is now the most widely used software library. TensorFlow is popular because of its many real-world applications in deep learning. If anyone looking to predict with high accuracy it will be wiser to go with Faster-RCNN and SSD. TensorFlow object detection API has inbuilt architectures like faster RCNN and SSD. We may utilize this framework to apply transfer learning to pre-trained models that were previously learned on big datasets, allowing us to tailor these models to a specific goal. For example, we may use transfer learning to develop a model that can determine whether or not someone is maintaining a distance or not. According to the theory of transfer learning, an image classification model may be used to represent the whole visual world provided it is trained on a sufficiently big and diverse dataset. These feature maps may then be used instead of starting from scratch by training a huge model using a large dataset. We don't need to use transfer learning in this scenario since we already have several models trained to recognize pedestrians[43]. Speed and accuracy are well-balanced with the Single Shot Detector. A convolutional network is applied to the input picture just once and a feature map is generated. A simple 3 x 3 convolutional kernel is used to anticipate the bounding boxes and classification probability from this feature map, which is quite small. The anchor boxes used by

SSDs such as Faster-RCNN are of different sizes, and instead of learning the box itself, the offset is learned. The SSD predicts bounding boxes after a large number of convolutional layers in order to handle the scale. The fact that each convolutional layer operates on a different scale implies that it is capable of distinguishing between things of diverse sizes and forms. The accuracy that we got in TensorFlow is 93.3% for human detection and 92.5% for the social distancing calculation. The accuracy is calculated here through the formula of determining accuracy which is dividing the total number of correct predictions which are corresponding diagonal to the matrix by the total number of output predictions. When we tried to implement the output, we got a variety of outcomes in terms of CPU and RAM utilization. In the following table, you can see how much CPU and RAM have been used.

5.1.1 Performance of TensorFlow

The TensorFlow Object Detection API provides a framework that allows you to employ pre-trained Object Detection Models like YOLO, SSD, RCNN, and Fast-RCNN, among others. The Single Shot Detector strikes a good mix between speed and accuracy. A feature map is created by applying a neural network to the input image only once. To predict the bounding boxes and classification probability from this little feature map, a basic 3 x 3 convolutional kernel is utilized. The anchor boxes utilized by 36 SSDs, such as Faster-RCNN, are various sizes, thus the offset is learned instead of the box itself. To address the scale, the SSD predicts bounding boxes after a large number of convolutional layers.

Model	Use	Accuracy	Time
TensorFlow	Human detection	93.3%	83 min 33 sec
TensorFlow	Social Distancing	92.5%	83 min 33 sec

Table 5.1: TensorFlow accuracy table

This table 5.1 shows the model's use for human detection and social distancing measurements and their respective accuracy with the desired time taken for generating the output.

Status	CPU usage(percentage)	RAM usage(percentage)
Pre-compiling	1.5	60.4
Compiling phase 1	74.1	74.7
Compiling phase 2	76.8	60.4
Post-compiling 2	2.6	60.6

Table 5.2: Performance of TensorFlow

The table 5.2 shows the usage of CPU and RAM while implementing the TensorFlow model. We got 5 FPS in this model. This model requires a very large amount of space in the device and also high end configured device as it contains various trained libraries and datasets. It took 1 hour 23 minutes 33 seconds to generate 15,456 data from video input.

5.1.2 Data output generated from TensorFlow

Number	Frame ID	Date and Time	Total people	People at Low Risk	People at High Risk
0	Frame 1	8/5/2022 12:25	33	10	20
1	Frame 2	8/5/2022 12:25	34	8	19
2	Frame 3	8/5/2022 12:25	39	8	21
3	Frame 4	8/5/2022 12:25	36	7	20
4	Frame 5	8/5/2022 12:25	37	7	22
...
15452	Frame 15453	8/5/2022 13:48	33	10	20
15453	Frame 15454	8/5/2022 13:48	33	10	20
15454	Frame 15455	8/5/2022 13:48	33	10	20
15455	Frame 15456	8/5/2022 13:48	33	10	20
15456	Frame 15457	8/5/2022 13:48	33	10	20

Table 5.3: Data output from TensorFlow model

The table 5.3 shows the output data which we generated from the input videos. Through this model we were able to generate 15,456 data, which means the TensorFlow model has ran for each of these times to identify the frame and detect humans in each frame and also calculate the distance. It also shows the output that are generated from the input video. We generated the total number of people count in the frame. We have also added a section where the total number of people at High Risk and Low Risk are plotted.

5.2 “You Only Look Once”- YOLO staging

You Only Look Once is the acronym for YOLO. Convolutional neural network-based object detection is used to recognize an object. YOLO has the benefit of speed without sacrificing precision. When using the Darknet implementation (official YOLO), the algorithm is able to anticipate exceptionally quickly. YOLO has a wider use [24]. It outperforms other approaches when applied to various types of images, such as artwork. There is a $S \times S$ grid for each picture that forecasts N bounding boxes as well as their confidence. It is based on the correctness of the bounding box and whether or not it includes an item. Box categorization scores are predicted for each class by YOLO as well. The likelihood of each class being present in a pedestrian point may be calculated by combining both classes.

On the other hand, most of these pedestrian points have low confidence ratings, so by setting a threshold of say 30 percent, we can exclude the majority of them. YOLO is very quick and can be run in real time. Training for the YOLO method consists of two phases: Classifier networks like VGG16 are first trained. We then replace the completely connected layers with a convolution layer and teach the system to recognize objects. Afterwards, 448×448 photos are used for object detection training purposes. As an example, YOLOv2 initially trains the classifier using 448×448 photos, but then retrains it using 224×224 images, which results in the classifier being trained with a significantly lesser number of epochs. Detector training

becomes simpler and mAP rises 4 percent as a result. Furthermore, YOLO views the whole picture at once, while the earlier approaches merely created area recommendations. In other words, having this background knowledge helps to prevent false positives. Even though it can only predict one class in one grid, YOLO has a problem with extremely small objects due to this constraint. But for accuracy YOLO will not be recommended as a detection model. The accuracy that we got from YOLOv5s model is 76.6% for human detection and 78.7% for social distancing measurements. While implementing the outcome, we saw a wide range of results in terms of CPU and RAM utilization. In the following table, you can see how much CPU and RAM have been used.

5.2.1 Performance of YOLO

The performance of image categorization networks has substantially improved because of the implementation of new training techniques. 15–30 frames per second(FPS) is a normal video frame rate. The number of frames per second necessary for real-time object detection varies based on the application’s characteristics, and such rates are comparable to the speed of a normal video that can be verified with human eyes.

Model	Use	Accuracy	Time
YOLOv5s	Human detection	78.7%	63 min 20 sec
YOLOv5s	Social Distancing	76.6%	63 min 20 sec

Table 5.4: YOLOv5 accuracy table

This table 5.4 shows the model’s use for human detection and social distancing measurements and their respective accuracy with the desired time taken for generating the output.

Status	CPU usage	RAM usage
Pre-compiling	1.2%	58.8%
Compiling phase 1	58.1%	62.4%
Compiling phase 2	56.8%	60.5%
Post-compiling 2	1.6%	60.4%

Table 5.5: Performance of YOLO model

The table 5.5 shows the usage of CPU and RAM while implementing the YOLOv5 model. Here we can see YOLOv5 is taking less RAM Usage than TensorFlow. This model requires a very small amount of space in the device and also any device of any configuration can smoothly run this model. As the pre-trained libraries and dataset or weights are of minimum size. It took 1 hour 3 minutes 20 seconds to generate 15,456 data from video input. The table 4 below shows the output data which we generated from the input videos. Through this model we were able to generate 15,456 data, which means the YOLOv5 model has ran for each of these times to identify the frame and detect humans in each frame and also calculate the distance.

5.2.2 Data output generated from YOLO model

Number	Frame ID	Date and Time	Total people	People at Low Risk	People at High Risk
0	Frame 1	12/5/2022 9:56	41	20	13
1	Frame 2	12/5/2022 9:56	42	20	13
2	Frame 3	12/5/2022 9:56	39	19	15
3	Frame 4	12/5/2022 9:56	36	18	17
4	Frame 5	12/5/2022 9:56	37	19	22
...
15452	Frame 15453	12/5/2022 10:58	6	2	3
15453	Frame 15454	12/5/2022 10:58	7	1	4
15454	Frame 15455	12/5/2022 10:58	7	2	4
15455	Frame 15456	12/5/2022 10:59	7	3	4
15456	Frame 15457	12/5/2022 10:59	8	2	5

Table 5.6: Data output from YOLOv5 model

The table 5.6 shows the output data which we generated from the input videos. Through this model we were able to generate 15,456 data, which means the YOLO model has ran for each of these times to identify the frame and detect humans in each frame and also calculate the distance. It also shows the output that are generated from the input video. We generated the total number of people count in the frame. We have also added a section where the total number of people at High Risk and Low Risk are plotted.

5.3 Comparison

This pandemic was not being given the attention it deserves. As a result, this system will be able to recognize humans, and by measuring the distance between them, it will be able to identify and warn people about the dangers of interacting too closely. Using YOLOv5s and TensorFlow, we implemented and demonstrated our findings. An assessment of this distance criteria is done to determine whether two persons are complying to social distancing rules. Because of its high precision and low error rate, the proposed method is straightforward to employ in real-world situations. In our tests, the TensorFlow model outperformed the YOLO model in terms of accuracy and precision. In some circumstances, the YOLO model fails to distinguish and detect humans due to low light in photos, whereas TensorFlow performs better. TensorFlow, on the other hand, is larger than YOLO since it provides large data-sized pre-trained libraries. In addition, the TensorFlow model necessitates high-end configuration devices in order to execute fast and smoothly and produce more accurate results. The TensorFlow model will continue to generate results on low resolution frames because most surveillance cameras do not produce acceptable quality output. The YOLOv5 model, on the other hand, will be unable to detect humans from the same data. Again, we have also shown the usage of RAM and usage of CPU for both the models. Where we gained results from the TensorFlow that it occupies higher RAM and higher CPU usage as it trains a large number of data with a large number of pre-trained libraries and dataset. Again, the usage of GPU libraries mostly makes the device compatible and the TensorFlow system gains more power for simulation.

While YOLO with numerous data might malfunction and also the results may not be as precious when compared to TensorFlow during low light.

MODEL	Accuracy	Avg. CPU usage	Avg. RAM usage	Time	FPS
TensorFlow	92.90%	38.75%	54.02%	83 min 33 sec	6
YOLOv5	77.65%	29.42%	60.52%	63 min 20 sec	5

Table 5.7: Comparison of YOLOv5 and TensorFlow

In the above table, we can see the comparison between TensorFlow and the YOLOv5 model. Here, the accuracy of TensorFlow is much higher than the YOLOv5 model. Also, RAM usage of TensorFlow is 54% whereas YOLOv5 RAM usage is 60% which is lower than YOLOv5. Again, the CPU usage of YOLOv5 is 29.42% and TensorFlow CPU usage is 38.75%. Furthermore, TensorFlow is taking 83 minutes 33 seconds to compute. On the other hand, YOLOv5 is taking 63 minutes 20 seconds to compute. YOLOv5 is giving 5 frames per second and TensorFlow is taking 6 frames per second which is more efficient.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We just got out of a deleterious pandemic, it is critical for us to maintain social distance and be wary of large crowds. Social isolation is one of the most important precautions in avoiding physical contact that could contribute to the spread of coronavirus. Noncompliance with these rules will result in increased viral transmission rates [56].

We proposed an image processing based human detection with monitoring via fine-tuned deep learning and computer vision for social distance measurement purposes. This pandemic is not being taken as seriously as it should be. As a result, this system will be able to identify and alert people to the dangers of socializing too close to each other. In addition, we've employed social distancing algorithms to gauge the level of social distance between individuals. We have implemented and shown our results through YOLOv5s and TensorFlow. To determine whether two people are adhering to social distancing norms, an evaluation of this distancing criteria is made. Using the proposed method in real-world applications is simple because of its high precision and low error rate. One of the most efficient methods of preventing the development of this pandemic is to establish social space between individuals. There is a fixed range of transmission for coughing, sneezing, and forced speaking droplets. By maintaining this distance, we can lessen the spread of the virus. Other information, such as the centroid coordinates, can also be found in the detection model's pedestrian points. Distance is derived using the pairwise centroid distances between the spots on the ground where pedestrians have been observed. An estimate of the physical distance between the person and the pixel is employed in order to identify social distance violations between persons, and a threshold is established for each violation. Those who practice social distance believe that by staying at home, avoiding large groups of people, and not interacting with one another, virus spread will be prevented. In conclusion, the research is likely to yield new information that will be useful in the study of social distancing and implementation. Using some algorithms, the paper investigates the significance and interactions of many factors on the formation, implementation, and compliance of social distance. The study also provided information on critical problems during the pandemic and steps to prevent the spread of the coronavirus and other future viruses. Our paper has the potential to uncover effective strategies for making social distance policy

measures more acceptable and comprehensible, resulting in actionable knowledge in the fight against the coronavirus and future viruses. Numerous recipients worldwide will benefit from the project's outcome.

6.2 Future Work

Coming from the news a new virus has been detected this month called monkeypox. Although it is not as deadly as COVID was, it spreads the same way as COVID did. This virus also requires social distancing protocols. The United Kingdom (UK) reported an imported case of monkeypox (MPX) in a traveler from Nigeria on May 7, 2022. On April 29, 2022, the subject developed a rash-like sickness and traveled from Lagos to London on May 3-4 [72]. The diagnosis was verified on May 6 by the UK Health Security Agency's Rare and Imported Pathogens Laboratory using monkeypox virus (MPXV) PCR on a vesicular swab [72]. Two more instances (both MSM) were recorded on May 18, 2022, one in London and the other in the South-East of England [72]. Monkeypox is a contagious disease that is difficult to spread between individuals but still during direct and prolonged face-to-face contact, the virus can be transferred between humans via respiratory droplets [72]. So, still we are not safe to gather at public places. So, thinking about the future use of such model the work could be improved for diverse indoor and outdoor conditions. To track the person or people who are violating or breaching the social distance threshold.

In future works, we intend to further enhance the performance of our algorithms by manually executing TensorFlow on the GPU. In addition, we would like to include more information about social distance in the loss function or directly in the model, and assign an appropriate risk level. In order to assess the risk of infection more accurately, it would be interesting to detect human gaze more thoroughly. Lastly, it would be fascinating to monitor these crowds with mobile cameras.

Bibliography

- [1] P. Viola, M. Jones, *et al.*, “Robust real-time object detection,” *International journal of computer vision*, vol. 4, no. 34-47, p. 4, 2001.
- [2] P. Barrett, J. Hunter, J. T. Miller, J.-C. Hsu, and P. Greenfield, “Matplotlib—a portable python plotting package,” in *Astronomical data analysis software and systems XIV*, vol. 347, 2005, p. 91.
- [3] D. Kirk, B. S. Center, *et al.*, “Nvidia cuda software and gpu parallel computing architecture,” 2008.
- [4] M. Ockay, M. Liska, *et al.*, “Compute unified device architecture (cuda) gpu programming model and possible integration to the parallel environment,” *Science & Military Journal*, vol. 3, no. 2, p. 64, 2008.
- [5] A. E. Zein, E. McCreath, A. Rendell, and A. Smola, “Performance evaluation of the nvidia geforce 8800 gtx gpu for machine learning,” in *International Conference on Computational Science*, Springer, 2008, pp. 466–475.
- [6] D. Chatterjee and V. Bertacco, “Equipe: Parallel equivalence checking with gp-gpus,” in *2010 IEEE International Conference on Computer Design*, IEEE, 2010, pp. 486–493.
- [7] P. Huang, A. Hilton, and J. Starck, “Shape similarity for 3d video sequences of people,” *International Journal of Computer Vision*, vol. 89, no. 2, pp. 362–381, 2010.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
- [9] F.-C. Hsu, J. Gubbi, and M. Palaniswami, “Human head detection using histograms of oriented optical flow in low quality videos with occlusion,” in *2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, IEEE, 2013, pp. 1–6.
- [10] A. Alahi, M. Bierlaire, and P. Vanderghenst, “Robust real-time pedestrians detection in urban environments with low-resolution cameras,” *Transportation research part C: emerging technologies*, vol. 39, pp. 113–128, 2014.
- [11] M. Manfredi, R. Vezzani, S. Calderara, and R. Cucchiara, “Detection of static groups and crowds gathered in open spaces by texture classification,” *Pattern Recognition Letters*, vol. 44, pp. 39–48, 2014.
- [12] 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.

- [13] M. Abadi, A. Agarwal, P. Barham, *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2016.
- [14] M. Abadi, P. Barham, J. Chen, *et al.*, “{Tensorflow}: A system for {large-scale} machine learning,” in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.
- [15] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, “Mean absolute percentage error for regression models,” *Neurocomputing*, vol. 192, pp. 38–48, 2016.
- [16] 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.
- [17] D. Rolon-Mérette, M. Ross, T. Rolon-Mérette, and K. Church, “Introduction to anaconda and python: Installation and setup,” *Python for research in psychology*, vol. 16, no. 5, S5–S11, 2016.
- [18] D. K. Singh and D. S. Kushwaha, “Tracking movements of humans in a real-time surveillance scene,” in *Proceedings of fifth international conference on soft computing for problem solving*, Springer, 2016, pp. 491–500.
- [19] M. Moniruzzaman, S. M. S. Islam, M. Bennamoun, and P. Lavery, “Deep learning on underwater marine object detection: A survey,” in *International Conference on Advanced Concepts for Intelligent Vision Systems*, Springer, 2017, pp. 150–160.
- [20] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, “Fast yolo: A fast you only look once system for real-time embedded object detection in video,” *arXiv preprint arXiv:1709.05943*, 2017.
- [21] X. Wang, H. W. Ng, and J. Liang, “Lapped convolutional neural networks for embedded systems,” in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2017, pp. 1135–1139.
- [22] F. Al-Azzo, A. M. Taqi, and M. Milanova, “Human related-health actions detection using android camera based on tensorflow object detection api,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, pp. 9–23, 2018.
- [23] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, “Computer vision and deep learning techniques for pedestrian detection and tracking: A survey,” *Neurocomputing*, vol. 300, pp. 17–33, 2018.
- [24] A. Kathuri, “How to implement a yolo (v3) object detector from scratch in pytorch: Part 1,” *PaperspaceBlog*, 2018.
- [25] P. Mustamo, “Object detection in sports: Tensorflow object detection api case study,” *University of Oulu*, 2018.
- [26] V. Nguyen, T. Dang, and F. Jin, “Predict saturated thickness using tensorboard visualization,” *Visualization in Environmental Sciences 2018*, 2018.
- [27] T. Xin, B. Guo, Z. Wang, *et al.*, “Freesense: A robust approach for indoor human detection using wi-fi signals,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–23, 2018.

- [28] E. N. Kajabad and S. V. Ivanov, “People detection and finding attractive areas by the use of movement detection analysis and deep learning approach,” *Procedia Computer Science*, vol. 156, pp. 327–337, 2019.
- [29] A. Talele, A. Patil, and B. Barse, “Detection of real time objects using tensorflow and opencv,” *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146*, 2019.
- [30] M. Venkata and S. Nishant, *Detecting and tracking of humans in an underwater environment using deep learning algorithms*, 2019.
- [31] F. Yin, X. Li, H. Peng, F. Li, K. Yang, and W. Yuan, “A highly sensitive, multifunctional, and wearable mechanical sensor based on rgo/synergetic fiber bundles for monitoring human actions and physiological signals,” *Sensors and Actuators B: Chemical*, vol. 285, pp. 179–185, 2019.
- [32] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.
- [33] K. Abughalieh and S. Alawneh, “Pedestrian orientation estimation using cnn and depth camera,” Tech. Rep., 2020.
- [34] P. Adarsh, P. Rathi, and M. Kumar, “Yolo v3-tiny: Object detection and recognition using one stage improved model,” in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, IEEE, 2020, pp. 687–694.
- [35] A. H. Ahamad, N. Zaini, and M. F. A. Latip, “Person detection for social distancing and safety violation alert based on segmented roi,” in *2020 10th IEEE international conference on control system, computing and engineering (ICCSC)*, IEEE, 2020, pp. 113–118.
- [36] N. Ahmed, R. J. Rony, and K. T. Zaman, “Social distancing challenges for marginal communities during covid-19 pandemic in bangladesh,” *Journal of Biomedical Analytics*, vol. 3, no. 2, pp. 5–14, 2020.
- [37] K. E. Ainslie, C. E. Walters, H. Fu, *et al.*, “Evidence of initial success for china exiting covid-19 social distancing policy after achieving containment,” *Wellcome Open Research*, vol. 5, 2020.
- [38] S. Anwar, M. Nasrullah, and M. J. Hosen, “Covid-19 and bangladesh: Challenges and how to address them,” *Frontiers in public health*, vol. 8, p. 154, 2020.
- [39] H. Bahri, M. Chouchene, F. E. Sayadi, and M. Atri, “Real-time moving human detection using hog and fourier descriptor based on cuda implementation,” *Journal of Real-Time Image Processing*, vol. 17, no. 6, pp. 1841–1856, 2020.
- [40] Y. C. Hou, M. Z. Baharuddin, S. Yussof, and S. Dzulkifly, “Social distancing detection with deep learning model,” in *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, IEEE, 2020, pp. 334–338.
- [41] F. Mahmud, “Coronavirus: In dense bangladesh, social distancing a tough task,” *Bangladesh News– Al-Jazeera*, 2020.

- [42] S. S. Padmanabula, R. C. Puvvada, V. Sistla, and V. K. K. Kolli, “Object detection using stacked yolov3.,” *Ingénierie des Systèmes d Inf.*, vol. 25, no. 5, pp. 691–697, 2020.
- [43] M. Painter and T. Qiu, “Political beliefs affect compliance with covid-19 social distancing orders,” *Covid Economics*, vol. 4, pp. 103–123, 2020.
- [44] 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.
- [45] —, “Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsort techniques,” *arXiv preprint arXiv:2005.01385*, 2020.
- [46] 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.
- [47] —, “Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic,” *Applied Sciences*, vol. 10, no. 21, p. 7514, 2020.
- [48] B. Roth, *Social distancing detector using a tensorflow object detection model, python and opencv, towards data science, 2020*, 2020.
- [49] I. Shete, “Social distancing and face mask detection using deep learning and computer vision,” Ph.D. dissertation, Dublin, National College of Ireland, 2020.
- [50] K. Sikali, “The dangers of social distancing: How covid-19 can reshape our social experience,” *Journal of community psychology*, 2020.
- [51] R. Szczepanek, “Analysis of pedestrian activity before and during covid-19 lockdown, using webcam time-lapse from cracow and machine learning,” *PeerJ*, vol. 8, e10132, 2020.
- [52] M. Wulandari, E. Syamsudin, *et al.*, “Recognition of pedestrian traffic light using tensorflow and ssd mobilenet v2,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1007, 2020, p. 012022.
- [53] G. Yang, W. Feng, J. Jin, *et al.*, “Face mask recognition system with yolov5 based on image recognition,” in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, IEEE, 2020, pp. 1398–1404.
- [54] A. Younis, L. Shixin, S. Jn, and Z. Hai, “Real-time object detection using pre-trained deep learning models mobilenet-ssd,” in *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, 2020, pp. 44–48.
- [55] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, “Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once,” 2021.
- [56] 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. 102571, 2021.

- [57] M. Ansari, D. K. Singh, *et al.*, “Monitoring social distancing through human detection for preventing/reducing covid spread,” *International Journal of Information Technology*, vol. 13, no. 3, pp. 1255–1264, 2021.
- [58] W. Bernasco, E. Hoeben, D. Koelma, *et al.*, “Promise into practice: Application of computer vision in empirical research on social distancing,” 2021.
- [59] B. Furundzic and F. Mathisson, *Dataset evaluation method for vehicle detection using tensorflow object detection api*, 2021.
- [60] A. Malta, M. Mendes, and T. Farinha, “Augmented reality maintenance assistant using yolov5,” *Applied Sciences*, vol. 11, no. 11, p. 4758, 2021.
- [61] A. Rahim, A. Maqbool, and T. Rana, “Monitoring social distancing under various low light conditions with deep learning and a single motionless time of flight camera,” *Plos one*, vol. 16, no. 2, e0247440, 2021.
- [62] —, “Monitoring social distancing under various low light conditions with deep learning and a single motionless time of flight camera,” *Plos one*, vol. 16, no. 2, e0247440, 2021.
- [63] W. Rahmaniari and A. Hernawan, “Real-time human detection using deep learning on embedded platforms: A review,” *Journal of Robotics and Control (JRC)*, vol. 2, no. 6, pp. 462–468, 2021.
- [64] P. Shukla, R. Kundu, A. Arivarasi, G. Alagiri, *et al.*, “A social distance monitoring system to ensure social distancing in public areas,” in *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, IEEE, 2021, pp. 96–101.
- [65] 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.
- [66] Z. Wang, Y. Wu, L. Yang, A. Thirunavukarasu, C. Evison, and Y. Zhao, “Fast personal protective equipment detection for real construction sites using deep learning approaches,” *Sensors*, vol. 21, no. 10, p. 3478, 2021.
- [67] Q. Xu, Z. Zhu, H. Ge, Z. Zhang, and X. Zang, “Effective face detector based on yolov5 and superresolution reconstruction,” *Computational and Mathematical Methods in Medicine*, vol. 2021, 2021.
- [68] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, “A forest fire detection system based on ensemble learning,” *Forests*, vol. 12, no. 2, p. 217, 2021.
- [69] D. Yang, E. Yurtsever, V. Renganathan, K. A. Redmill, and Ü. Özgüner, “A vision-based social distancing and critical density detection system for covid-19,” *Sensors*, vol. 21, no. 13, p. 4608, 2021.
- [70] I. H. Al Amin, F. H. Arby, *et al.*, “Implementation of yolo-v5 for a real time social distancing detection,” *Journal of Applied Informatics and Computing*, vol. 6, no. 1, pp. 01–06, 2022.
- [71] —, “Implementation of yolo-v5 for a real time social distancing detection,” *Journal of Applied Informatics and Computing*, vol. 6, no. 1, pp. 01–06, 2022.
- [72] R. R. ASSESSMENT, “Monkeypox multi-country outbreak,” 2022.

- [73] A. R. Cubeta Caballo and C. J. Aliac, “Yolo-based tricycle counting in aid of traffic analysis,” in *2022 4th Asia Pacific Information Technology Conference*, 2022, pp. 150–154.
- [74] K. Guo, C. He, M. Yang, and S. Wang, “A pavement distresses identification method optimized for yolov5s,” *Scientific Reports*, vol. 12, no. 1, pp. 1–15, 2022.
- [75] S. Saponara, A. Elhanashi, and Q. Zheng, “Developing a real-time social distancing detection system based on yolov4-tiny and bird-eye view for covid-19,” *Journal of Real-Time Image Processing*, pp. 1–13, 2022.
- [76] T. Sharma, B. Debaque, N. Duclos, A. Chehri, B. Kinder, and P. Fortier, “Deep learning-based object detection and scene perception under bad weather conditions,” *Electronics*, vol. 11, no. 4, p. 563, 2022.
- [77] A. Sonavane and R. Kohar, “Dental cavity detection using yolo,” in *Proceedings of Data Analytics and Management*, Springer, 2022, pp. 141–152.
- [78] V. Tiwari, A. Singhal, and N. Dhankhar, “Detecting covid-19 opacity in x-ray images using yolo and retinanet ensemble,” in *2022 IEEE Delhi Section Conference (DELCON)*, IEEE, 2022, pp. 1–5.
- [79] P. S. Jadhao and D. N. Duche, “Social distancing monitoring system for covid-19,”