Title: AI-powered Smart Social distancing detector

Project by: Mr. Theivaprakasham H (CB.EN.P2CEN20026)

Project Guide: Mr. Sajith Variyar V. V.

Abstract

In this period of COVID-19 pandemic, the concept of social distancing had become a new normal to limit the spread of coronavirus. In this study, a smart social distance detector is proposed using a real-time/recorded, video from a camera and object detection deep learning architectures. The procedural workflow is as follows. Firstly, the real-time/recorded video is streamed and pre-processed. Secondly, the pre-processed video is fed into a pre-trained single-stage object detection architecture namely Yolov3 and Yolov4 to detect and annotate the persons with bounding box in the video. Finally, the euclidean distance are calculated between every person and are classified as safe, cautious and unsafe depending on the mandated physical distance norms. Yolov4-tiny-288 and yolov4-tiny-416 models had the best performance running at the fastest speed at 7.3 Frames per second followed by the yolov3-tiny 608px model had the next best performance of 4.6FPS. From the results we observe that Jetson Nano B01 Developer board can be an affordable solution for building a smart social distance detector.

Keywords: Social-distancing Detector; Centroid distance calculation; Jetson Nano Social Distancing BOT.

---

1. Introduction

The COVID-19 shifted the perspective of the planet on pandemics that have dire implications for global economy and wellbeing. Beginning in Wuhan, China, 210 countries and territories worldwide recorded over ten million people diagnosed with more than 500,000 deaths within just six months (January-June 2020). Aside from the global health crises (i.e., a potential incidence of 25% in the US, 1 million job losses in Canada in March 2020, 1 million employment losses in Australia and estimated a global decline of 3% of GDP), COVID 19 caused dramatically economic losses, resulting in a global recession as many economists expected.

The World Health Organisation (for Disease Control et al., 2020) believes that physical distancing plays a vital role in the future containment of the present CoVid19 pandemic and points out that such strategies are capable of preventing the production and mutation of influenza epidemics, but only when used in tandem, enabled without delay and sustained for a comparatively long period of time. Due to COVID-19 pandemic, society need to embrace and adopt new norm that includes practising social distance to break the transmission. The onset of the COVID-19 pandemic has led to an increase in the importance of social distance to intervene in the spread of the virus by curbing social interactions and maintaining physical distances. To avoid or decrease the incidence and degree of transmission of the disease within a population, social distancing can be described as a non-pharmaceutical disease prevention and control intervention implemented to curtail interaction between those who are infected with a disease and those who are not. Kelso et al., (Kelso et al., 2009) suggested a critical role of social distancing in the potential control of a future pandemic and indicate that such interventions are capable of arresting influenza epidemic development.

1.1 Literature review

Some of the relevant works on human detection and the previous work done on social distancing using deep learning are highlighted in this section. With advancement in the compute capability of the Graphics cards, high performance computing systems and advanced optimization algorithms, the community of Artificial Intelligence has witnessed a phenomenal growth. The state-of-the-art analysis reflects largely on the latest work of study using machine learning to detect objects. A convolutional neural network (CNN or ConvNet) is a special type of artificial neural network which is predominating pattern detection and has hence proved as a highly efficient algorithm in image recognition and image classification. It had revolutionized the domain of Computer Vision by serving as a basic building block for the Neural Network Architectures. The initial convolutional layers are responsible for feature extraction, whereas extracted features are mapped to the final output layers for classification and regression. Convolution Neural Networks have emerged as one of the game changing algorithm in the field of image classification(Theivaprakasham, 2020), speech recognition, recommender systems, natural language processing, medical image analysis (Vyshnav et al., 2020), Face recognition, text classification, climate zone classification, scene classification (Damodaran et al., 2019), financial time-series data analysis (Veith and Steele, 2018). Keniya and Mehendale (Keniya and Mehendale, 2020) proposed a new architecture named SocialdistancingNet-19 for detecting the frame of a person and displaying labels, they are marked as safe or unsafe with an accuracy of 92.8 %. Rusli et al., (Rusli et al., 2020) proposed a MySD system (My Safe Distance) that leverages smart phone hardware features that typically has Bluetooth transceiver as well GPS to determine safe distance and required level compliance. Ahamad et al., 2020 (Ahamad et al., 2020) suggested a system for the identification of individuals in the areas of interest using the Single Shot Single Multibox Detector of MobileNet (SSD) and the Image processing OpenCV library. The distance tracking system achieved an outdoor and complicated input videos precision of between 56.5 and 68% for monitoring, while the indoor checking setting achieved 100% precision. Hou et al.,(Hou et al., 2020) presents a methodology by employing YOLOv3 algorithm for pedestrian detection and to calculate the distance between them using a video feed.

From the literature survey we understand that there are many similar works on social distancing using Deep Learning techniques implemented using an unaffordable graphic card. But none of the social detection system were tested using low-cost affordable compute capable deployment ready real-time embedded system like Nvidia Jetson Nano B01.

1.2 Problem statement

Continuous Lockdown measures cannot be the solution for sustainability and the economics of the world. There is an urgent need for the alternative affordable monitoring system based on social distancing to prevent the spread of CoVid19 and economic revival. Usage of AI technology on the concept of social distancing on a affordable embedded board will not only be helpful for the government to monitor the people’s behaviour, but also helps small-scale companies and educational institutions to setup such devices for strict monitoring of social distancing norms.

1.3 Objectives

The following are the main objectives of our proposed framework.

- To explore capabilities of NVIDIA Jetson Nano B01 Developer Board for AI
- Exploring different YOLO architectures for Object Detection
- Designing Social Distancing Detector using the YOLO architecture
- Testing the deployablity of our detector in both real-time and recorded videos

1.4 Scope of the work

Social distancing interventions such as complete lockdown, school closure and prohibition of public gatherings are present in this period of pandemic influenza. However, they cannot continue for long as it severely affects the sustainability and the economics of the world. Around the world, citizens are practising physical distancing to help flatten the curve on the COVID-19 pandemic. But with many countries cautiously restarting parts of their economies, concerns have emerged regarding workplace safety in this new environment. Hence an alternative monitoring system or technique to keep them alert about the situation and spread of the disease.

2. Proposed Work

Our research aims to decide whether a person practises the social distancing law. The results are confirmed by using both a live stream and a video feed. We can understand whether a person preserves social distance by calculating the difference of two frames of individuals from the centroids. Finally, the persons are labelled as safe, cautious, and unsafe.

The proposed framework performs five major tasks in the following order:

A) Person Identification from a Video frame (Recorded/Live)

B) Live person Tracking

C) Social distance estimation between persons through Euclidean distance calculation

D) Alerting mechanism when the Social distancing violation are reported through coloured bounding boxes and notification

E) Deploying the Social distance system to real-time using NVIDIA Jetson Nano B01 and Raspberry Pi v2 8MP Camera

2.1 Hardware & Software details

2.1.1 Hardwares

1. NVIDIA Jetson Nano B01

   - 5V 4A Power Supply

   - WiFi Dongle

   - Wireless Mouse and Keyboard

2. Raspberry Pi v2 8MP Camera with CSI Connector

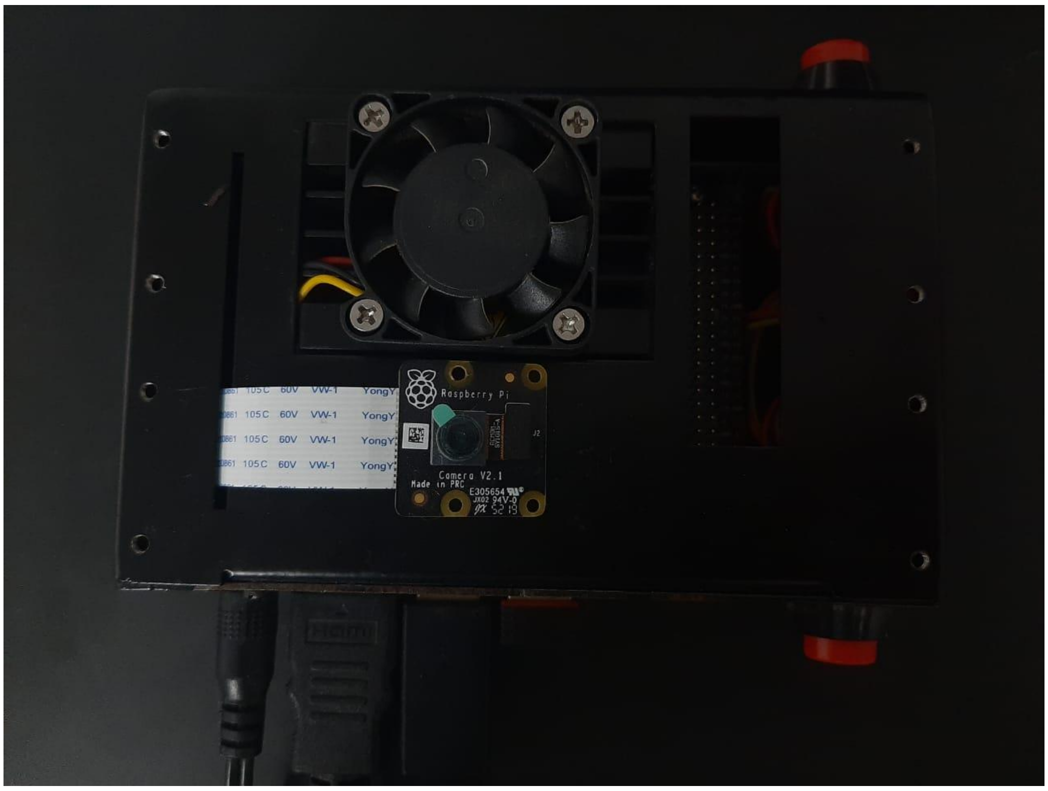


Figure 1: NVIDIA Jetson Nano B01 Developer Board in chasis connected to Raspberry Pi v2 8MP Camera via CSI Connector

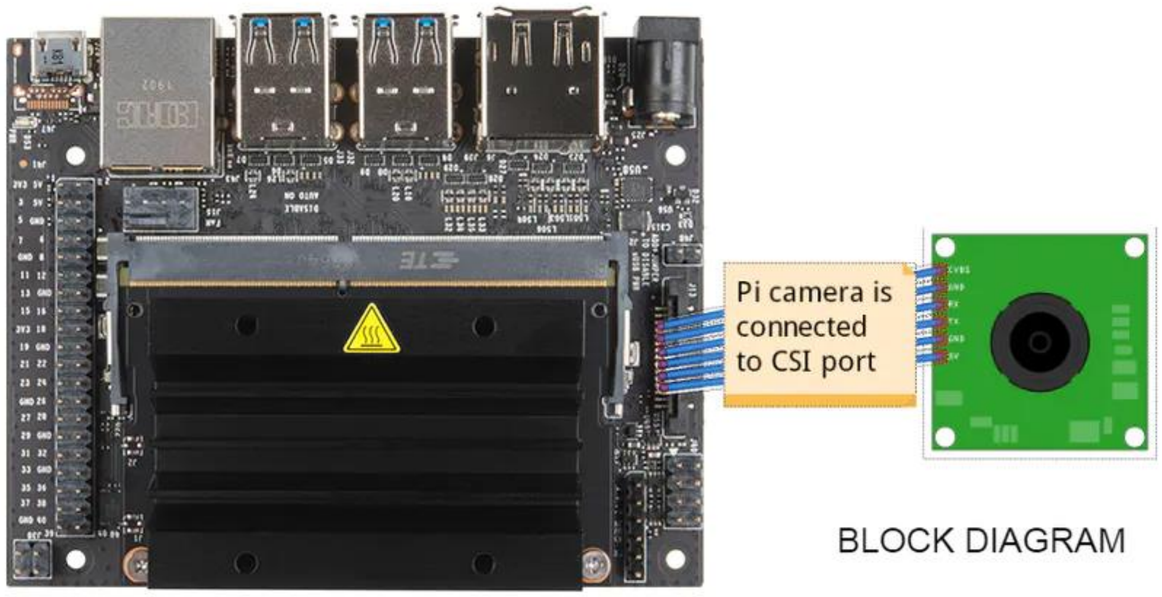


Figure 2: NVIDIA Jetson Nano B01 Developer Board connected to Raspberry Pi v2 8MP Camera via CSI Connector

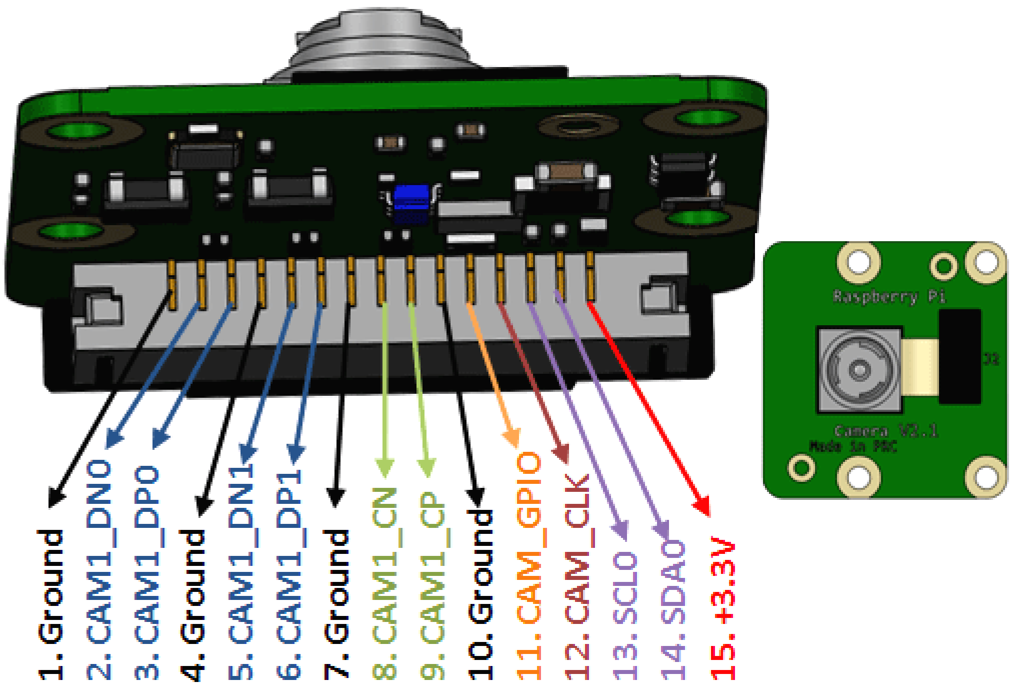


Figure 2: Pinout configuration of Raspberry Pi v2 8MP Camera

2.1.2 Softwares

We used the following dependencies installed on Ubuntu 18.0 LXDE Operating system (supported by Nvidia Jetson Nano B01 Board).

1. OpenCV==4.5 (with GPU Enabled)
2. Numpy==1.16.1
3. Scipy==1.6.0
4. Imutils==0.5.4
5. Argparse==1.4.0
6. PyCuda

We downloaded the pre-trained weights from the following link *https://github.com/artynet/darknet-alexeyAB#pre-trained-models*.

2.1.3 Total Development cost

Total development cost of the Smart Social distancing detector was Rs. 11570.

Components	**Cost**
NVIDIA Jetson Nano B01	Rs.9000
5V 4A Power Supply	Rs.350
WiFi Dongle	Rs.270
Wireless Mouse and Keyboard	Rs.450
Raspberry Pi v2 8MP Camera with CSI Connector	Rs.1500
Total Cost	**Rs. 11570**

2.1.4 Pre-trained models

We used 11 pre-trained models namely yolov3-spp, yolov3-288, yolov3-416, yolov3-608, yolov3-tiny 228px, yolov3-tiny 416px, yolov3-tiny 608px, yolov4-288, yolov4-608, yolov4-tiny-288 and yolov4-tiny-416 to build our Social distancing detector. The Yolov3 and Yolov4 models are depicted as show from Figure 4 and Figure 5. YOLO V3 is an improvement in comparison to previous YOLO networks. The multi-scale detection, stronger network extractor features and some loss functional changes are compared to previous versions. This network will now be able to detect many more targets from big to small. YOLO V3 also runs quite fast, just like other single-shot detectors, and allows GPU devices to be inferred in real time.

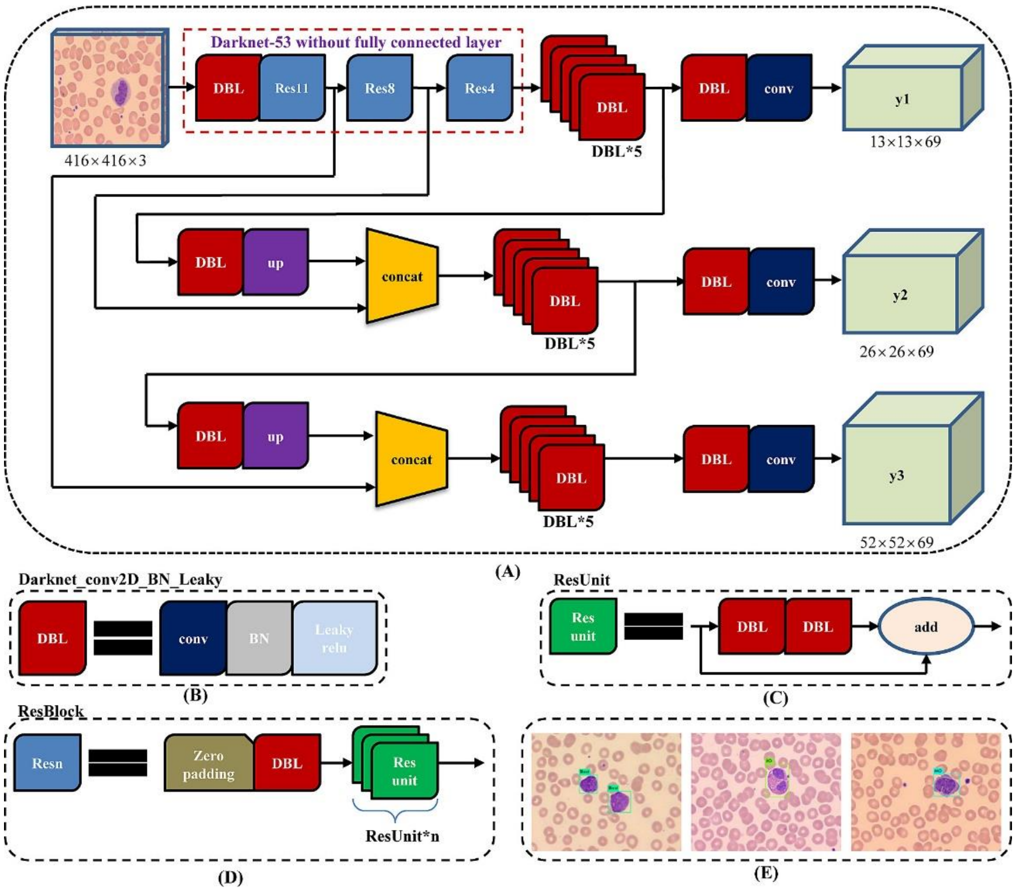


Figure 4: YOLOv3 pipeline with input image size 416×416 and 3 types of feature map (13×13×69, 26×26×69 and 52×52×69) as output

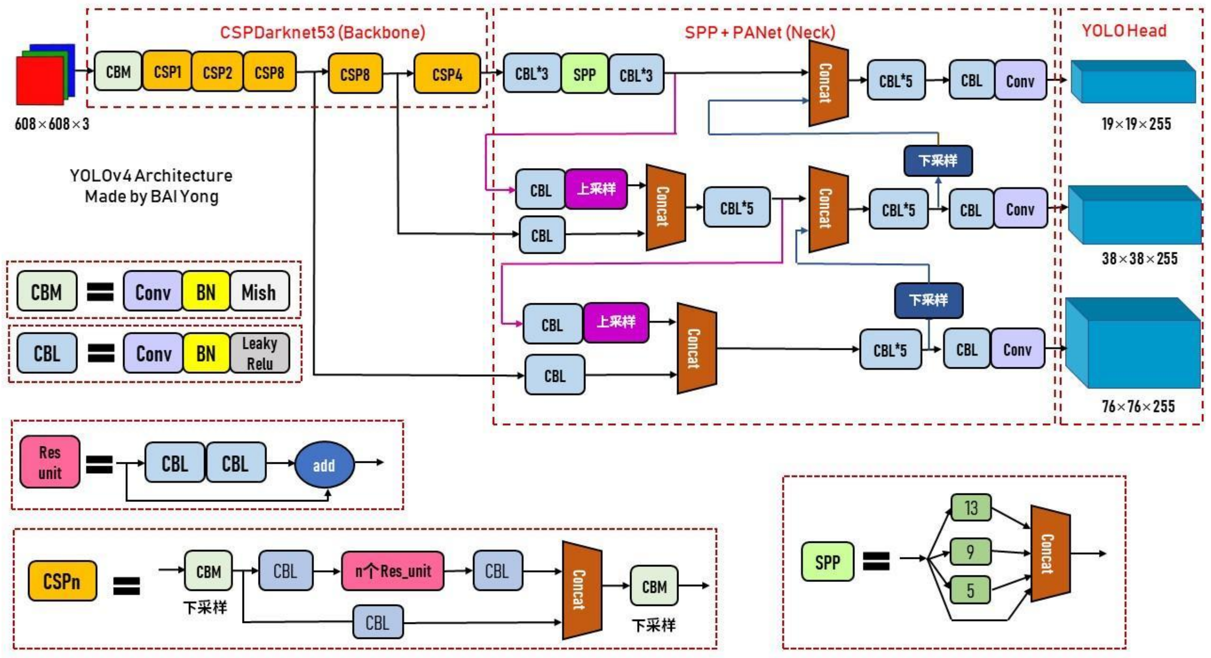


Figure 5: Overview of Yolov4 model

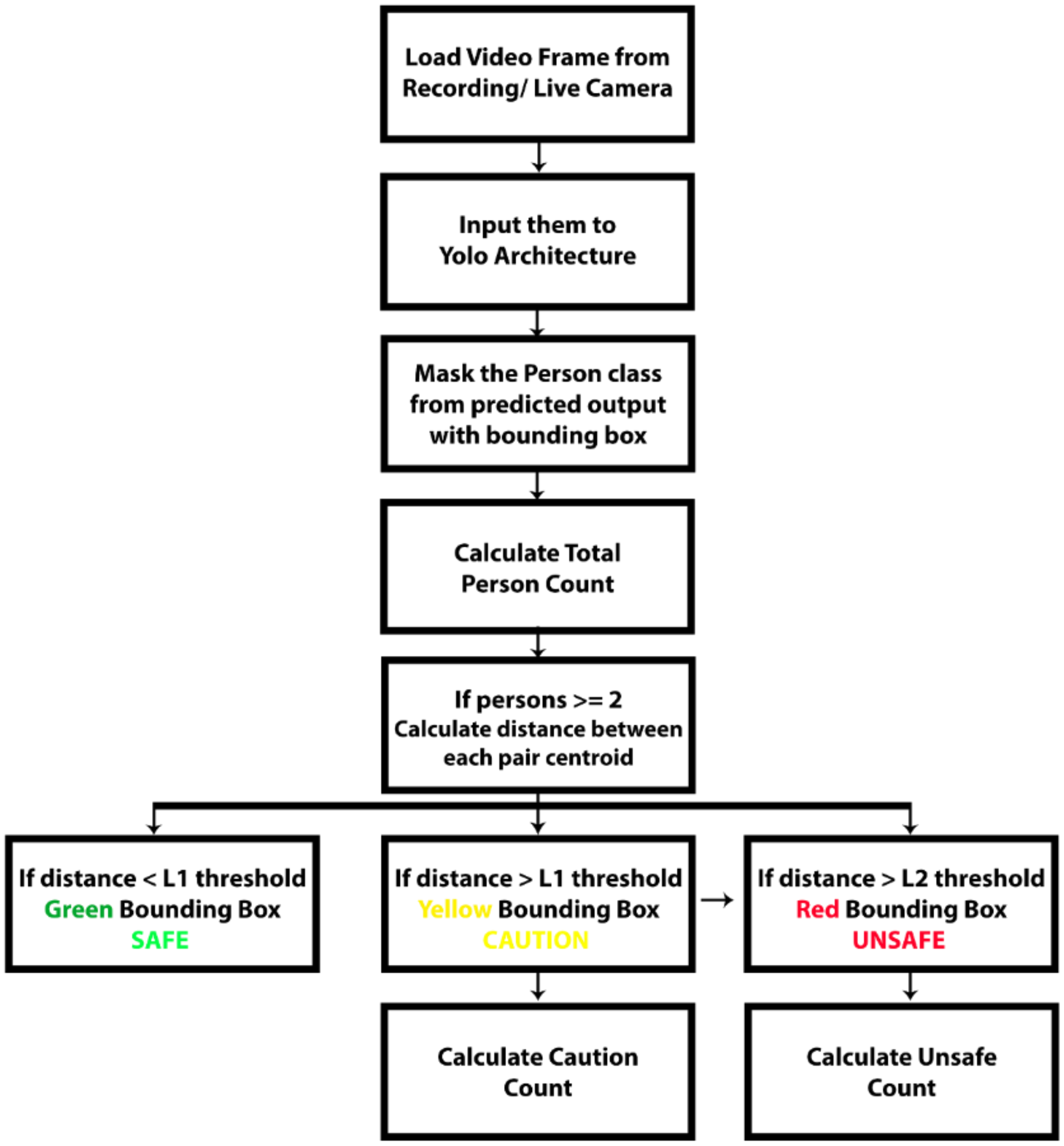
2.2 Workflow

The workflow for this proposed framework is illustrated in Figure 6.

1. We connect the CSI Camera with Jetson Nano B01.
2. We then define the minimum object detection confidence, non-maxima suppression threshold, Caution Threshold(L1) and Unsafe Threshold(L2) in our code.
3. We extract each video frame from the live stream/recorded video.
4. We pass the input image into the pre-trained Yolo models for object detection.
5. The results of the Yolo models that contains the person prediction probability and bounding box coordinates of the detection.
6. Based on the centroid value we then calculate the centroid of the object and store then in an array
7. Pair-wise Euclidean distance is then calculated with respect to each pair of centroids and store in a square matrix.
8. The pairwise matrix is then assessed for the threshold limits. If the calculated Euclidean distance has not crossed L1 threshold, then we change the colour of the bounding box to green meaning that the pair of persons are at safe distance. If the

calculated Euclidean distance has crossed L1 threshold, then we change the colour of the bounding box to yellow meaning that the pair of persons are at cautious distance. If the calculated Euclidean distance has crossed L2 threshold, then we change the colour of the bounding box to red meaning that the pair of persons are at unsafe and have violated the social distancing norms.

9. We then calculate the total number of cautious, unsafe and total number of people and display them on the frame.
10. We also display violation alert when the limit of unsafe people has crossed more than 13.



Load Video Frame from Recording/ Live Camera
Input them to Yolo Architecture
Mask the Person class from predicted output with bounding box
Calculate Total Person Count
If persons >= 2 Calculate distance between each pair centroid
If distance < L1 threshold Green Bounding Box SAFE
If distance > L1 threshold Yellow Bounding Box CAUTION
If distance > L2 threshold Red Bounding Box UNSAFE
Calculate Caution Count
Calculate Unsafe Count

Figure 6: Workflow for this proposed framework

2.2.1 Sample Execution Code:

```
#Importing required libraries
from mylib import config
from mylib.detection import detect_people
from imutils.video import VideoStream, FPS
import time
from scipy.spatial import distance as dist
import numpy as np
import argparse, imutils, cv2, os, time, schedule

#Selecting the model
keyy= "yolov3-tiny-288"

#---------------------------Parse req. arguments----------------------------#
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="",
          help="path to (optional) input video file")
ap.add_argument("-o", "--output", type=str, default="./"+keyy+"_processed.avi",
          help="path to (optional) output video file")
ap.add_argument("-d", "--display", type=int, default=1,
          help="whether or not output frame should be displayed")
args = vars(ap.parse_args())
#---------------------------------------------------------------------------#

#Making use of gstreamer_pipe for live streaming of video from CSI Port of Jetson Nano B01

def gstreamer_pipeline(
    capture_width=1280,
    capture_height=720,
    display_width=1280,
    display_height=720,
    framerate=60,
    flip_method=0,
):
    return (
        "nvarguscamerasrc ! "
        "video/x-raw(memory:NVMM), "
        "width=(int)%d, height=(int)%d, "
        "format=(string)NV12, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )
# load the weights and cfg
weightss = "/yolo/"+keyy +".weights"
cfgg = "/yolo/" + keyy+ ".cfg"

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

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

# load our YOLO object detector trained on COCO dataset (80 classes)
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

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

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

# if a video path was not supplied, grab a reference to the camera
if not args.get("input", False):
	print("[INFO] Starting the live stream..")
	vs = cv2.VideoCapture(gstreamer_pipeline(flip_method=0), cv2.CAP_GSTREAMER)

# otherwise, grab a reference to the video file
else:
	print("[INFO] Starting the video..")
	vs = cv2.VideoCapture(args["input"])

writer = None
# start the FPS counter
fps = FPS().start()

# loop over the frames from the video stream
while True:
	# read the next frame from the file

	(grabbed, frame) = vs.read()
	# if the frame was not grabbed, then we have reached the end of the stream
	if not grabbed:
		break

	# resize the frame and then detect people (and only people) in it
	frame = imutils.resize(frame, width=700)
	start = time.time()

	results = detect_people(frame, net, ln,
		personIdx=LABELS.index("person"))

	end = time.time()

	print("[INFO] YOLO processes at {:.6f} FPS".format((1/(end - start))))

	# initialize the set of indexes that violate the max/min social distance limits
	serious = set()
	abnormal = set()

	# ensure there are at least two people detections (required in
	# order to compute our pairwise distance maps)
	if len(results) >= 2:
		# extract all centroids from the results and compute the
		# Euclidean distances between all pairs of the centroids
```

```
                centroids = np.array([r[2] for r in results])
                D = dist.cdist(centroids, centroids, metric="euclidean")

                # loop over the upper triangular of the distance matrix
                for i in range(0, D.shape[0]):
                        for j in range(i + 1, D.shape[1]):
                                # check to see if the distance between any two
                                # centroid pairs is less than the configured number of pixels
                                if D[i, j] < config.MIN_DISTANCE:
                                        # update our violation set with the indexes of the
centroid pairs
                                        serious.add(i)
                                        serious.add(j)
              # update our abnormal set if the centroid distance is below max distance limit
                                if (D[i, j] < config.MAX_DISTANCE) and not serious:
                                        abnormal.add(i)
                                        abnormal.add(j)

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

                # if the index pair exists within the violation/abnormal sets, then update the
color

                if i in serious:
                        color = (0, 0, 255)
                elif i in abnormal:
                        color = (0, 255, 255) #orange = (0, 165, 255)

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

        # draw some of the parameters
        Safe_Distance = "Safe distance: >{} px".format(config.MAX_DISTANCE)
        cv2.putText(frame, Safe_Distance, (470, frame.shape[0] - 25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255, 0, 0), 2)
        Threshold = "Theivaprakasham H (CB.EN.P2CEN20026)"
        cv2.putText(frame, Threshold, (470, frame.shape[0] - 380),
                cv2.FONT_HERSHEY_SIMPLEX, 0.30, (255, 0, 0), 2)

    # draw the total number of social distancing violations on the output frame
        text = "Total serious violations: {}".format(len(serious))
        cv2.putText(frame, text, (10, frame.shape[0] - 55),
                cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 0, 255), 2)

        text1 = "Total abnormal violations: {}".format(len(abnormal))
        cv2.putText(frame, text1, (10, frame.shape[0] - 25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 255, 255), 2)

        text2 = keyy + " - {:.6f} FPS".format((1/(end - start)))
        cv2.putText(frame, text2, (10, frame.shape[0] - 85),
                cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 255, 255), 2)
#------------------------------Alert function----------------------------------#
        if len(serious) >= config.Threshold:
                cv2.putText(frame, "-ALERT: Violations over limit-", (10, frame.shape[0] - 80),
                        cv2.FONT_HERSHEY_COMPLEX, 0.60, (0, 0, 255), 2)
#----------------------------------------------------------------------------#
        # check to see if the output frame should be displayed to our screen
        if args["display"] > 0:
                # show the output frame
                cv2.imshow("EC Project: Live Social Distancing Detector-  Theivaprakasham H", 
frame)
                key = cv2.waitKey(1) & 0xFF
                # if the `q` key was pressed, break from the loop
                if key == ord("q"):
                        break
    # update the FPS counter
        fps.update()

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

# stop the timer and display FPS information
fps.stop()
print("===========================")
print("[INFO] Elasped time: {:.2f}".format(fps.elapsed()))
print("[INFO] Approx. FPS: {:.2f}".format(fps.fps()))

# close any open windows
cv2.destroyAllWindows()
#==========================================================================#
```

3. Results

The performance overview of the 11 pre-trained models deployed on NVIDIA Jetson Nano B01 Developer Kit in Table 1. Yolov4-tiny-288 and yolov4-tiny-416 models had the best performance running at the fastest speed at 7.3 Frames per second. The yolov3-tiny 608px model had the next best performance of 4.6FPS. Figure 8,9,10,11 shows the implementation of Yolo models on the recorded video.

Table 1: Performance overview of different models on Jetson Nano B01 Developer Board

Models	FPS on Nano	mAP @IoU=0.5
yolov3-spp	1.15	60.6
yolov3-288	1.15	52.1
yolov3-416	1.15	55.3
yolov3-608	1.15	57.9
yolov3-tiny 228px	4.6	15
yolov3-tiny 416px	4.6	20
yolov3-tiny 608px	4.6	22.2
yolov4-288	1.15	62.8
yolov4-608	1.15	65.7
yolov4-tiny-288	**7.3**	**34.1**
yolov4-tiny-416	7.3	38.1

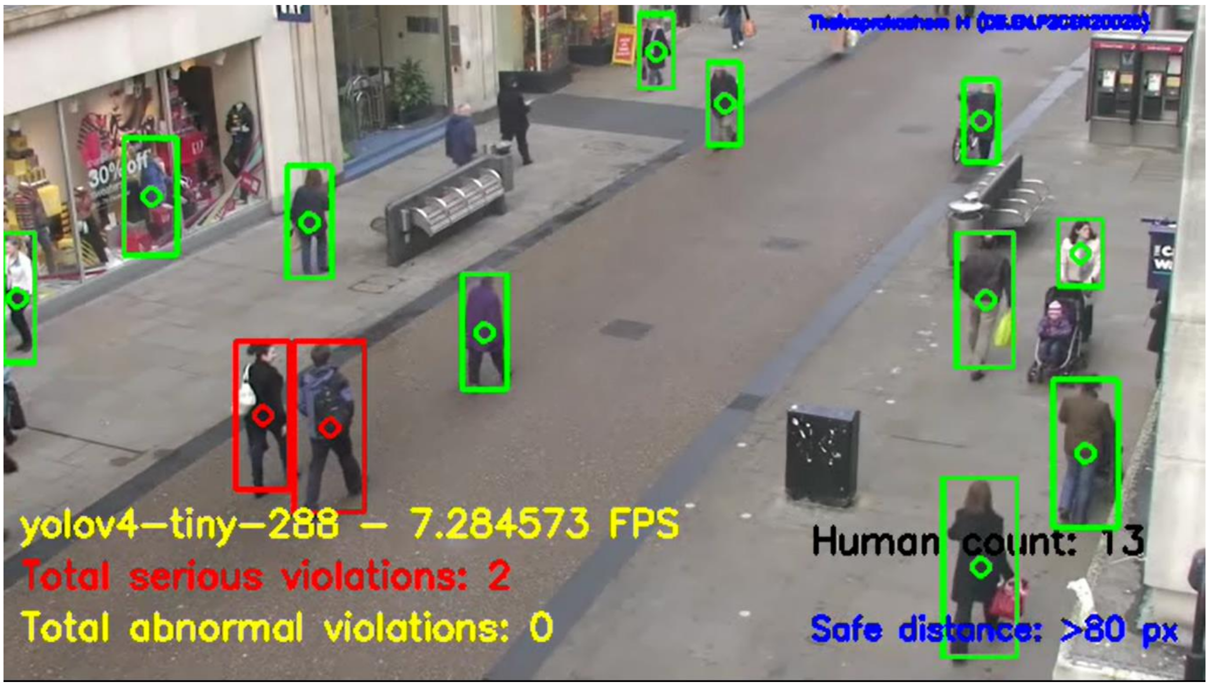

yolov4-tiny-288 - 7.284573 FPS
Total serious violations: 2
Total abnormal violations: 0
Human count: 13
Safe distance: >80 px

Figure 8: Performance of Yolov4-Tiny288 model on a recorded video

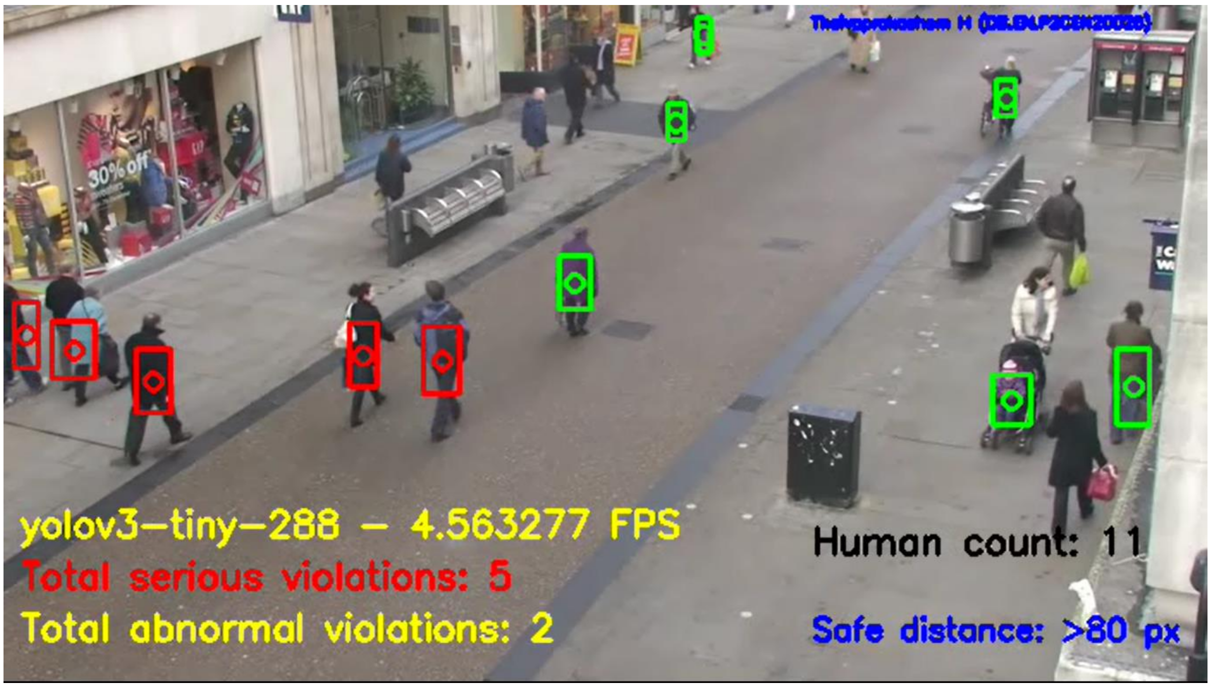

yolov3-tiny-288 - 4.563277 FPS
Total serious violations: 5
Total abnormal violations: 2
Human count: 11
Safe distance: >80 px

Figure 9: Performance of Yolov3-Tiny288 model on a recorded video

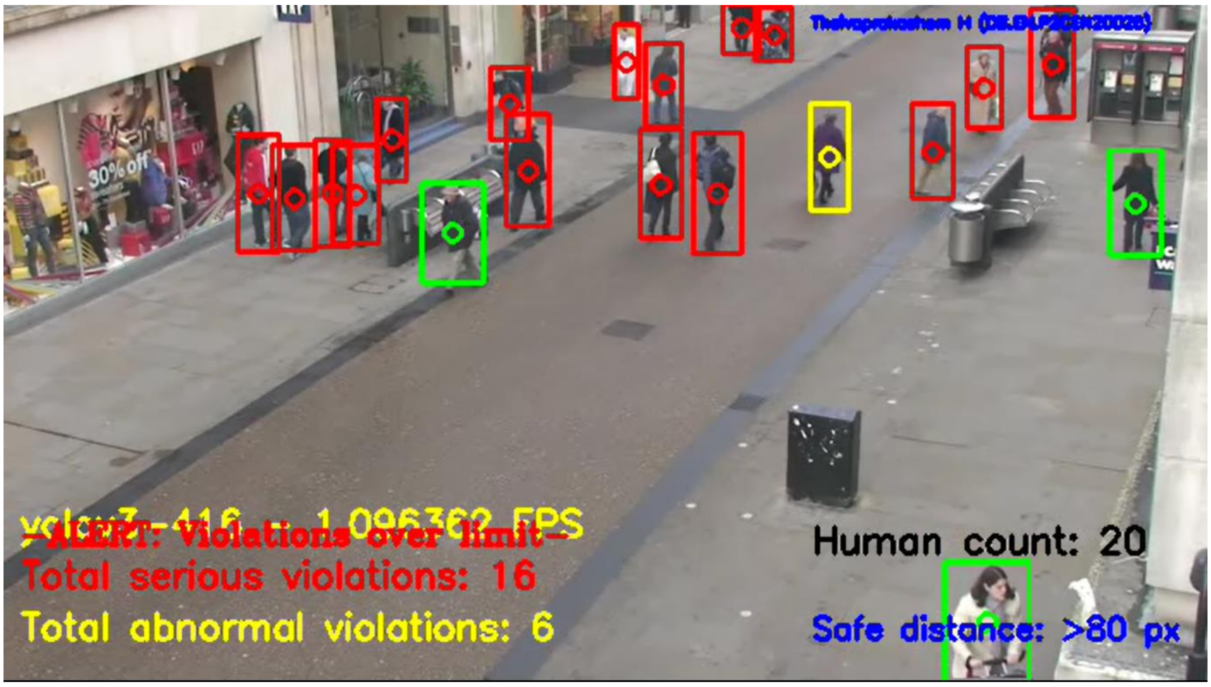

Thalvaprakasham H (DE.EXLP2CEN20026)
yolov3-416 — 1.096362 FPS
-ALERT: Violations over limit-
Total serious violations: 16
Total abnormal violations: 6
Human count: 20
Safe distance: >80 px

Figure 10: Performance of Yolov3-416 model on a recorded video

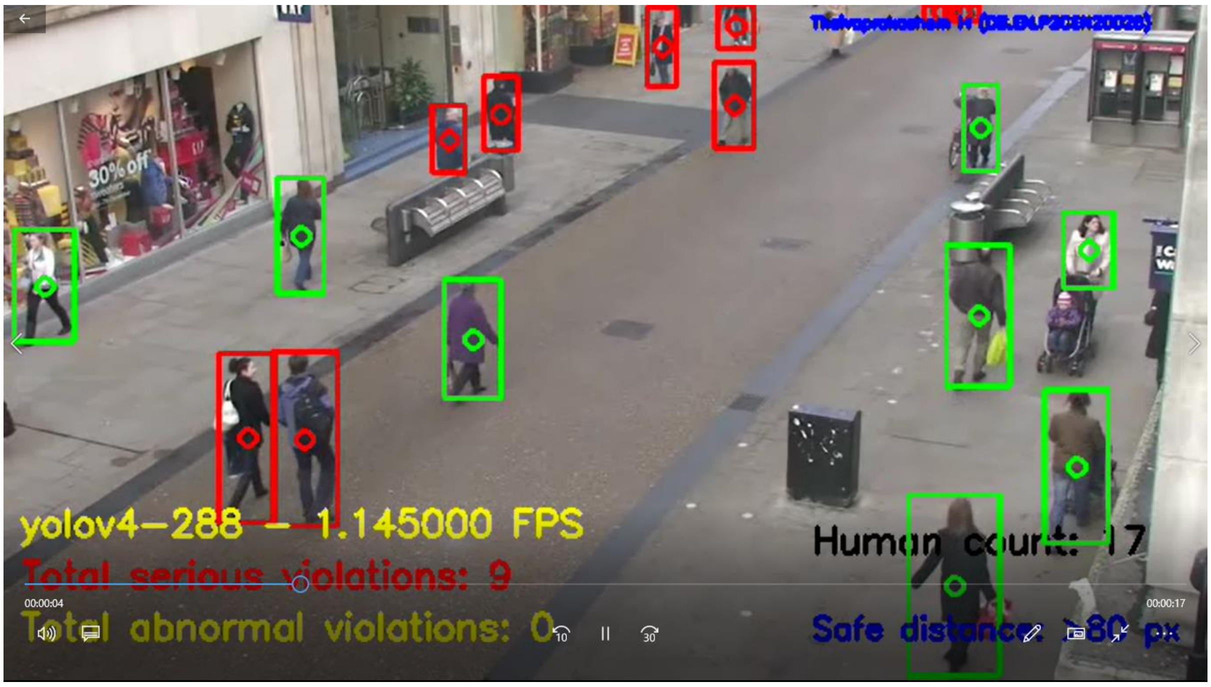

Thalvaprakasham H (DE.EXLP2CEN20026)
yolov4-288 — 1.145000 FPS
Total serious violations: 9
Total abnormal violations: 0
Human count: 17
Safe distance: >80 px
00:00:04
00:00:17

Figure 11: Performance of Yolov3-416 model on a recorded video

4 Conclusion

In this study we have built a low-cost smart social distance detector by using Yolov3 and Yolov4 deep learning architectures on Jetson Nano B01 Developer Board. Our experiments using real-time/recorded, video from a camera revealed that Yolov4-tiny architecture had the top performance of mean average Precision of 40.2% with processing power of 7.2 Frames per second. We firmly believe that our AI-powered smart social distance detector will help people keep track of and remind them to stick to this practise. The direct effect of our application would be a smaller or minimum percentage of COVID-19 cases due to a high degree of social distance conformity. Using this knowledge, it is possible to recognise bottlenecks where social distance cannot be sustained and how people respond when constraints are introduced or removed. This form of data not only demonstrates how physical distances change in real time, but also gives comprehensive insight into long-term behavioural changes.

5 Future Works

Though we have successfully deployed our Social Distancing Detector using Jetson Nano B01 Developer Kit, we had few shortcomings and challenges. These challenges helped us to propose our future works.

Implementation of other Object detection Architectures

The proposed work only focusses on implementing YOLO architecture variants on Jetson Nano B01 Developer Board. In future, we will work to try and implement other Object detection architectures like Single Shot Detectors (SSD), RetinaNet, Faster RCNN and EfficientDet to validate their performance.

Exploring TensorRT

NVIDIA TensorRT is an SDK for high-performance deep learning inference that provides INT8 and FP16 optimizations for production deployments of deep learning inference applications such as video streaming, speech recognition, recommendation, fraud detection, and natural language processing. In future works, we will try to port our existing object detection variants with TensorRT platform to optimize the performance of our model.

Innovative alternative metric to detect social-distancing

The centroid calculation method for determining the safe and un-safe persons is not very precise and accurate as the image in 2D feature space whereas the environment is a 3D feature space. This may lead to frequent misidentifications and cause non-reliability to our system. In future works, we will be working on to develop novel, innovative alternative metric to detect social distancing through various techniques like Stereographic projection/multiple video stream 3D mapping to implement new metrics and determine the safe limits.

Real-time notification alert system

The present study missed many additional notification features. In future we will make value additions like automated message notification through Bots, Email-alerts and SMS alerts using Internet of Things and GSM technology.

References

Ahamad, A.H., Zaini, N., Latip, M.F.A., 2020. Person Detection for Social Distancing and Safety Violation Alert based on Segmented ROI, in: Proceedings - 10th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2020. Institute of Electrical and Electronics Engineers Inc., pp. 113–118. https://doi.org/10.1109/ICCSCE50387.2020.9204934

Damodaran, N., Sowmya, V., Govind, D., Soman, K.P., 2019. Single-Plane Scene Classification Using Deep Convolution Features, in: Wang, J., Reddy, G.R.M., Prasad, V.K., Reddy, V.S. (Eds.), Soft Computing and Signal Processing. Springer Singapore, Singapore, pp. 743–752.

for Disease Control, C., Prevention, others, 2020. Implementation of mitigation strategies for communities with local COVID-19 transmission. fact sheet, 2020a. As March 12.

Hou, Y.C., Baharuddin, M.Z., Yussof, S., Dzulkifly, S., 2020. Social Distancing Detection with Deep Learning Model, in: 2020 8th International Conference on Information Technology and Multimedia, ICIMU 2020. Institute of Electrical and Electronics Engineers Inc., pp. 334–338. https://doi.org/10.1109/ICIMU49871.2020.9243478

Kelso, J.K., Milne, G.J., Kelly, H., 2009. Simulation suggests that rapid activation of social distancing can arrest epidemic development due to a novel strain of influenza. BMC Public Health 9. https://doi.org/10.1186/1471-2458-9-117

Keniya, R., Mehendale, N., 2020. Real-Time Social Distancing Detector Using Socialdistancingnet-19 Deep Learning Network. SSRN Electron. J. https://doi.org/10.2139/ssrn.3669311

Rusli, M.E., Yussof, S., Ali, M., Abobakr Hassan, A.A., 2020. MySD: A Smart Social Distancing Monitoring System, in: 2020 8th International Conference on Information Technology and Multimedia, ICIMU 2020. Institute of Electrical and Electronics Engineers Inc., pp. 399–403. https://doi.org/10.1109/ICIMU49871.2020.9243569

Theivaprakasham, H., 2020. Identification of Indian butterflies using Deep Convolutional Neural Network. J. Asia. Pac. Entomol. https://doi.org/10.1016/j.aspen.2020.11.015

Veith, N., Steele, R., 2018. Machine learning-based prediction of ICU patient mortality at time of admission, in: ACM International Conference Proceeding Series. Association for Computing Machinery, pp. 34–38. https://doi.org/10.1145/3206098.3206116

Vyshnav, M.T., Sowmya, V., Gopalakrishnan, E.A., V V, S. V, Menon, V.K., Soman, K.P., 2020. Deep Learning Based Approach for Multiple Myeloma Detection, in: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–7. https://doi.org/10.1109/ICCCNT49239.2020.9225651