Real-Time and Accurate Monitoring of Social Distance during COVID-19 Situation

Shourove Sutradhar Dip[1], Shuvo Raj Bangshi[2], Sajjad Waheed[3]
Department of Information and Communication Technology
Mawlana Bhashani Science and Technology University
Santosh, Tangail-1902, Bangladesh
Email: shourovdip@ieee.org[1], bangshishuvo@gmail.com[2], swaheed,iu@gmail.com[3]

***Abstract*—In recent years, a lot of research works have been done on object detection using various machine learning models. However, not many works have been done on detecting humans in particular. This study works with the newly released YOLOv4 object detector to detect humans to use the detections for maintaining social distance. A large number of features are included in the YOLOv4 model which makes it faster, more accurate than other object detection models, and also can process the video at 65 FPS. The model is originally trained on the MS-COCO dataset which has 80 different classes. But in this study, only one class named "Person" has been used. The YOLOv4 is fine-tuned to improve the speed of detecting humans with satisfying accuracy. These detections are then used to build a system for maintaining social distance and alerting the authority if a breach in the social distance is detected. This system can be applied at ticket counters, hospitals, offices, factories etc. It can also be used for maintaining social distance among the students and the teachers in the classroom for their safety.**

Keywords—Object Detection, Human Tracking, Computer Vision.

I. INTRODUCTION

In recent years, a lot of research works have been done on object detection. Every year we see better and updated object detectors, but as those are trained on general-purpose datasets (like MS COCO), we miss out on targeted model improvements for human-related data. Detecting human beings from surveillance system videos is attracting more attention as it has a vast range of applications such as abnormal activity detection, human movement analysis, people counting, personal identification. These detections are then used in various applications such as self-driving cars, security measurements, identifying particular humans, gender classification etc.

Videos from surveillance cameras usually have low resolution. The scenes captured by a static camera mostly don't have any change in the background. If the video surveillance system relies on human observers for detecting specific activities, there is a high probability of error as humans have limitations to monitor simultaneous events in surveillance displays. So, detecting humans and analyzing their activities in automated video surveillance has become a very popular and attractive research topic in the area of Computer Vision [1].

There are two approaches to object detection. They are: 1) Machine Learning based approach and 2) Deep Learning based approach.

For the Machine Learning based approach, it is needed to define features and then use techniques for classification. For the Deep Learning based approach, it is not very important to define the features as they perform end-to-end object detection. Convolutional Neural Network (CNN) is a class of Deep Neural Networks (DNN) that are vastly used in image visualization applications. However, a standard CNN can not be used for detecting objects due to the variable length of the output layer, as there can be multiple occurrences of the same object in an image [2].

An object detector architecture generally consists of four components: the input, the backbone, the head and the neck. The backbone is the pre-trained network that takes the input images and does the feature extraction related works. The head predicts the classes and draws bounding boxes around detected objects. The neck collects feature maps from intermediate stages to increase robustness.

Such algorithms can be trained and optimized for human detection and tracking. These detections and tracking information can further be used for applications like maintaining social distance. In this study, the YOLOv4 detector will be fine-tuned to detect humans faster and the fine-tuned detector will be used for tracking human movements and maintaining social distance.

The processing factor of the input video with low-cost hardware is very important for building a better object detection model. The YOLOv4 object detection model fulfills these requirements for better experiments. The new features of YOLOV4 are Weighted-Residual-Connections (WRC), Cross-Stage-Partial-Connection (CSP), Cross-mini-Batch-Normalization (CmBN), Self-Adversarial-Training (SAT) and Mish-activation. The YOLOv4 model is also suitable for training on a single GPU. This model is twice faster than various other models and improves YOLOv3's AP and FPS by 10% and 12%, respectively [3].

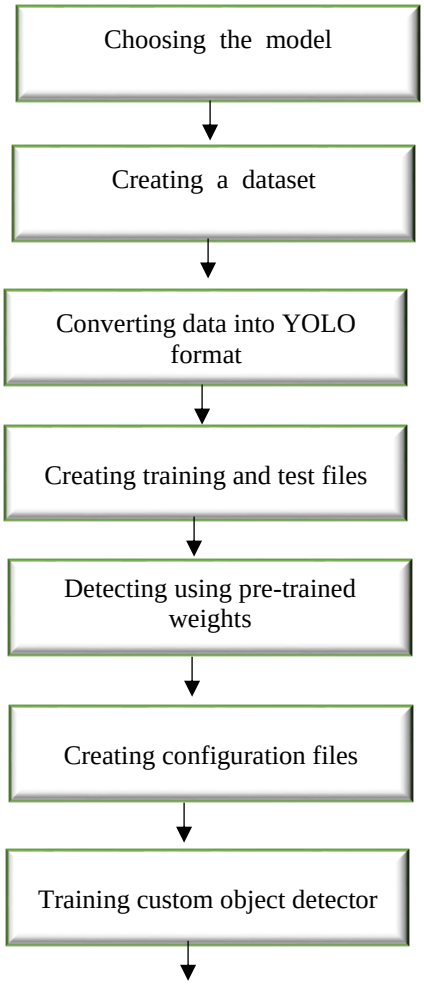
Not maintaining social distance properly is one of t0he main reasons for the widespread of the COVID-19 virus. An analysis by the experts at Emory University over 10,000 cases of SARS-CoV-2 coronavirus infection showed that one-fifth of all infections examined came from just 2% of infected patients [4]. The results of this research confirm the importance of maintaining social distance. This was the main motivation behind this study. This study aimed to build an intelligent system that can detect humans close to each other and alert the authority about the breach in social distance.

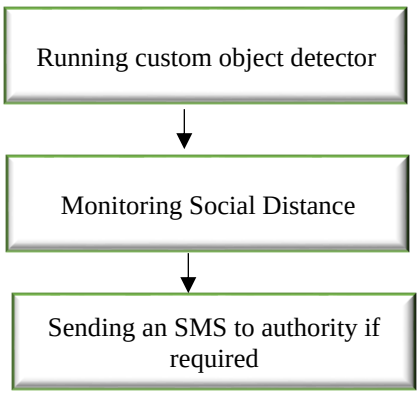
II. RELATED WORKS

P. Mahto et al proposed a refined YOLOv4 for detecting vehicles [2]. They've improved the YOLOv4 detector specifically for vehicle tracking applications. But in this study, the YOLOv4 has been improved to be used in human tracking applications. J. Zicong et al proposed an improved object-detection model using YOLOv4-Tiny [5]. They've aimed to faster detection of various objects. But the YOLOv4-tiny was detected with less accuracy than the

YOLOv4. Bo. Gong et al worked on detecting wheat heads using the YOLOv4 model [6]. They've modified the structure of the YOLOv4 model to improve the accuracy and detection time and used YOLOv3's head structures to predict the bounding box of objects. Another work by Saleh Albahli et al used a fine-tuned YOLOv4 model for detecting melanoma, which is the skin cancer caused by UV radiation from the sun [7]. They used the YOLOv4 model to detect melanoma at multiple scales and orientations despite of skin color variation. In this study, a fine-tuned YOLOv4 model is used for detecting and tracking humans. Another study by Ebrahim Najafi et al [8] was about preparing a labeled dataset from open sources for 11 object classes and the analysis of two well-known object detection methods in the task of urban electronic inventory. They collected data from the environment and used data augmentation techniques to generate more data. In our study, the data was collected from open source platform for one object class and data augmentation techniques were applied to generate more data. Another study by P. Steinkellner and K. Schöffmann used recording videos of specific moments from customer's skilling experiences from modern ski resorts for tracking skiers [9]. In our study, the Deep-Sort tracking algorithm was used to track humans. In another study by Schütz, A.K. et al, YOLOv4 was used to detect foxes and monitor their activities [10]. In our study, a similar technique was used to detect humans and track their movements.

III. METHODOLOGY





3.1. Choosing the model

The study was started with a comparison of the default YOLOv3 model with the default YOLOv4 model. Based on the results, it was decided that the YOLOv4 model worked faster and it was more accurate for real-time detection.

3.2. Creating a dataset

For creating the dataset of only humans, "Open Images Dataset V6" was used, which is an open-source program by Google. A dataset with a single class "Person" was created here. The OIDv4 toolkit was used for downloading the dataset.

3.3. Converting image data into YOLO format

Google Colab was used for writing the codes to convert the annotations into YOLO format. The YOLO supported format is: Location Number, center x, center y, width, height [3]. The previously downloaded Dataset and csv_folder were uploaded to the google drive. A new Google Colab file was then created and mounted to drive. After that only required columns from annotations csv file was collected and not required columns were omitted. Then only records with 0matching classes were collected. Then new columns, required for YOLO format, were added to the csv file. Next, an iteration through all the class strings was performed to create a list and a class number was assigned according to the order that they appeared on the list. Then, center x, center y, width and height were calculated. Next, the dataframe with YOLO required values was generated. Then, a text file containing the required data in the YOLO format was created. Finally, another text file was created in the same directory with the name "classes.txt" containing only "Person".

3.4. Creating training and test files for YOLOv4

For training and testing, "train.txt" and "test.txt" files were created respectively each containing fully qualified paths of images on which the custom YOLOv4 model was trained. With the help of another Google Colab Notebook, a train.txt file was created containing 80% data (lines) inside it. Similarly, the test.txt file was created with 20% data (lines) inside it. After that, two other files named "image_data.data" and "classes.names" were created which were used for training purposes in the YOLO framework. The prior file

contained details such as number of classes, the fully qualified path of train.txt, test.txt, the path to the file classes.txt and the folder name that installed the trained weights. The later file contained different classes of the images, in this case, only one class "Person" was present.

3.5. Object detection using YOLOv4 pre-trained weights

A new Google Colab Notebook was created which was used to clone the Darknet repository [4]. Then some configurations were changed inside the file named "Makefile". The following values were changed: GPU = 1, CUDNN = 1, OPENCV = 1. This will make sure that the GPU, CUDNN (a GPU-accelerated library of primitives for deep neural networks) and OPENCV (a library of programming functions mainly aimed at real-time computer vision) are enabled. After making the changes and saving the file, the Darknet framework was compiled with the help of the Google Colab Notebook to use related files for training the object detection model. Then the YOLOv4 weights were downloaded and some .jpg [5] and .mp4 files were uploaded to the "data" folder inside the "darknet" repository's cloned folder for testing. Then the object detection was performed on the images and video files that were just uploaded.

3.6. Creating configuration files in YOLO object detection

Configuration files (.cfg files) contain information such as learning rate, saturation level, changing the brightness of images, information about rotating images as well as configurations related to CNN layers such as activation function, size and number of filters, strides [3]. This particular configuration file also contained 3 YOLO layers at the last which described the architecture of YOLO [3]. The following information was updated in the yolov4.cfg file: (a) Number of classes, (b) Convolutional layer filters. For YOLO architecture, Number of filters = (Number of classes + 5) × B (here B is the number of boxes predicted by YOLOv4 for every cell of the feature maps). The value of B was suggested as 3 [3]. For this custom model, where only one class was used, the number of filters was (1+5) × 3 = 18. Now, two separate .cfg files were created, each for train and test data, in the "cfg" folder inside "darknet" repository's cloned folder, named as "yolov4_train.cfg" and "yolov4_test.cfg". The contents of these files were same as yolov4.cfg except the following parameters of yolov4_train.cfg needed to be changed as the following values: batch = 32, subdivisions = 16, max_batches = 2000, steps = 1600, 1800, classes = 1, filters = 18. Next, some parameters inside the "yolov4_test.cfg" were changed as follows: batch = 1, subdivisions = 1, max_batches = 2000, steps = 1600, 1800, classes = 1, filters = 17.

3.7. Training custom object detector

To train the custom object detection model using the YOLOv4 framework on this custom dataset, first, the pre-trained weights for convolutional layers were downloaded. The pre-trained weights were being used as transfer learning to decrease the training time on the previously downloaded dataset. After downloading the pre-trained weights, the custom object detector was trained in the Google Colab Notebook, with the GPU selected as the runtime type, using the "yolov4_train.cfg" file which was created previously.

3.8. Running custom object detector

After the training was complete, the custom object detector was run on the .jpg [5] and .mp4 files which were uploaded previously, using a previously mentioned facility with a higher GPU capability. An individual ID is set two each human detected and every detection is tracked using the Deep-Sort algorithm. So one particular human will have only one ID even if the detection is lost for some moments. This makes it easier to identify the detections properly.

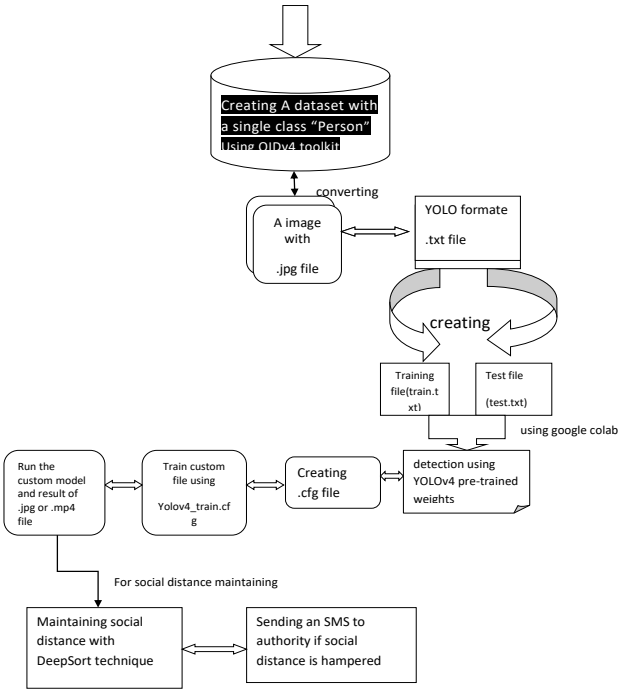
3.9. Monitoring Social Distance

The required functionalities for maintaining social distance were applied. A threshold value for the minimum distance between two humans was pre-defined. This value may differ according to the camera position and distance and angle between camera and humans. If two humans are detected at minimum or more than the threshold distance, then green bounding box is drawn surrounding them indicating that they are at safe distance. If two humans are detected at less distance than the threshold distance, then a red bounding box is drawn surrounding them indicating that they are not at safe distance.

3.10. Sending an SMS to authority if social distance is hampered

If the object detector detects at least two humans are at a distance less than the safe distance, then an alarming text is sent immediately to an authority phone number. The "Twilio" API was used for sending the text messages.

SYSTEM ARCHITECTURE:



IV. RESULTS

As can be seen in the image below, the custom YOLOv4 detector had a slight (from 1% to 3%) decrease in accuracy but took less time to complete the detection. The original YOLOv4 took 79.030 milliseconds to complete the detection on an image [5] but the custom YOLOv4 detector took 77.392 milliseconds to complete the detection on the same image. The outcome was the same for .mp4 files with a slight decrease in accuracy but faster detection.

For Object Detection:

Figure 2. Detecting various objects in an image using the original YOLOv4 object detector (on left) and detecting only persons using the custom YOLOv4 object detector (on right)

For social distance maintaining:

V. DISCUSSION

As the decrease in accuracy is just from 1% to 3%, there is no problem in detecting a person. But as the speed of detection has increased, the detector detects humans faster in real-time and performs operations faster, which is more beneficial.

VI. CONCLUSION

The functionality for maintaining the social-distance needs to be applied. The detector can be trained on more images to improve the accuracy. This detector can be integrated into a website or can be used in a mobile app for being used at hospitals, schools, shops etc.

REFERENCES

[1] Paul, M., Haque, S.M.E. & Chakraborty, S. Human detection in surveillance videos and its applications - a review. EURASIP J. Adv. Signal Process. 2013, 176 (2013). https://doi.org/10.1186/1687-6180-2013-176

[2] P. Mahto, P. Garg, P. Seth and J. Panda, "REFINING YOLOV4 FOR VEHICLE DETECTION," International Journal of Advanced Research in Engineering and Technology (IJARET), vol. 11, no. 5, pp. 409-419, 2020.

[3] A. Bochkovskiy, C.-Y. Wang and H.-y. M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020. arXiv:2004.10934. [Online]. Available: http://arxiv.org/abs/2004.10934.

[4] Lau, Max S. Y., Bryan Grenfell, Michael Thomas, Michael Bryan, Kristin Nelson, and Ben Lopman. 2020. Characterizing superspreading events and age-specific infectiousness of SARS-CoV-2 transmission in Georgia, USA. Proceedings of the National Academy of Sciences117: 22430–35.

[5] J. Zicong, Z. Liquan, L. Shuaiyang and J. Yanfei, Real-time object detection method based on improved YOLOv4-tiny, 2020.

[6] Gong, Bo & Ergu, Daji & Cai, Ying & Ma, Bo. (2020). A Method for Wheat Head Detection Based on Yolov4. 10.21203/rs.3.rs-86158/v1

[7] Albahli, Saleh & Nida, Nudrat & Irtaza, Aun & Yousaf, Muhammad Haroon & Tariq, Muhammad & Mahmood, Muhammad. (2020). Melanoma Lesion Detection and Segmentation Using YOLOv4-DarkNet and Active Contour. 10.1109/ACCESS.2020.3035345.

[8] Kajabad, Ebrahim & Begen, Petr & Nizomutdinov, Boris & Ivanov, Sergey. (2021). YOLOv4 for Urban Object Detection: Case of Electronic Inventory in St. Petersburg. 316-321. 10.23919/FRUCT50888.2021.9347622.

[9] P. Steinkellner and K. Schöffmann, "Evaluation of Object Detection Systems and Video Tracking in Skiing Videos," 2021 International Conference on Content-Based Multimedia Indexing (CBMI), 2021, pp. 1-6, doi: 10.1109/CBMI50038.2021.9461905.

[10] Schütz, A.K.; Schöler , V.; Krause , E.T.; Fischer , M.; Müller , T.; Freuling, C.M.; Conraths , F.J.; Stanke, M.; Homeier-Bachmann, T.; Lentz, H.H.K. Application of YOLOv4 for Detection and Motion Monitoring of Red Foxes. Animals 2021, 11, 1723. https://doi.org/10.3390/ani11061723