**Object Detector for Real-Time and Accurate Monitoring of COVID-19 Patients and Maintaining Social Distance**

Shourove Sutradhar Dip Shuvo Raj Bangshi Sajjad Waheed

Email: shourovdip@ieee.org Email:bangshishuvo@gmail.com Email:swaheed.iu@gmail.com

Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University,

Santosh, Tangail 1902, Bangladesh

***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 paper works with the newly released YOLOv4 object detector to detect humans to use the detections for maintaining social distance.YOLOv4 is one kind of framework for object detection and use in CNN(Convolution Neural Network).YOLO means “You Only Look Once”.YOLO-based CNN family of models have some series for object detection but YOLOv4 is the most recently released model. A large number of features are included in the YOLOv4 model which is said to improve Convolution Neural Network(CNN). YOLOv4 model is faster, more accurate than others object detection models, and also can process the video at 65 FPS. The accuracy level 43.5%AP(65.7% AP₅₀) for the MS COCO dataset with the great achievement of the YOLOv4 model. In this paper, we are working with only one class named human. 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. This system works for basically social distance maintaining among the people in this COVID-19 situation especially some valuable places like as the ticket counter in the railway station, bus station, airport. It is also worked for maintaining social distance among the students and the teachers in sitting positions in the classroom for their safety.

***Keywords*:** Object Detection, Human Tracking, Computer Vision**.**

***1. INTRODUCTION***

Real-time human detection is a technology employed in applications like self-driving cars, surveillance cameras. 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. 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 [1]. The YOLOv4 model is quite faster than the YOLOv3 or others object detection models. Not only high accuracy is not the only factor in object detection anymore but also runs the model smoothly in the edge devices. The processing factor of the input video in real-time with low-cost hardware is also very important for building a better object detection model. The YOLOv4 object detection model fulfills both requirements for better experiments. This YOLOv4 model also makes changes to make the detector more suitable for training on a single GPU. YOLOv4 model is twice faster than the other models and improves YOLOv3’s AP and FPS by 10% and 12%, respectively. The new features of YOLOV4 are ‘Weighted-Residual-Connections(WRC), Cross-Stage-Partial-Connection(CSP), Cross-mini-Batch-Normalization(CmBN).’

Figure 1: Graphical representation of YOLOv4 comparision among the others model.

***1.1 Motivations***

This work will make it easier to detect human movements. It will be useful to ensure maintaining social distance and to keep track of the COVID-19 patients. It also ensures a better and safe working environment in the office, sitting in the classroom in educational institutions, Prayer hall like a mosque, temple, etc.

***1.2 Objectives***

This work aims to improve the newly released, YOLOv4 detector, specifically, for human tracking applications using some existing methods [1]. The goal is to train the YOLOv4 detector on a custom dataset of humans to detect humans faster in real-time and track their movements and use the detections for maintaining social distance.

***2.Related Works:***

P. Mahto et al proposed a refined YOLOv4 for detecting vehicles [1]. They’ve improved the YOLOv4 detector specifically for vehicle tracking applications.

J. Zicong et al proposed an improved object-detection model using YOLOv4-Tiny [2]. 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 [3]. 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 [4]. They used the YOLOv4 model to detect melanoma at multiple scales and orientations despite of skin color variation.

The next one is YOLOv4 for Urban Object Detection: Case of Electronic Inventory in St.Petersburg[5]. This work is 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 in Saint Petersburg in Russia under the concept of Smart City methods and technologies done by Ebrahim Najafi Kajabad , Petr Begen, Boris Nizomutdinov, Sergey Ivanov and using YOLOv4 for object detection such as Windows, Doors, Adv Billboards, Ramps, etc.

***3.*Methodology *And System Architecture***

Choosing the model

Creating a dataset

Converting data into YOLO format

Creating training and test files

Detecting using pre-trained weights

Creating configuration files

Training custom object detector

Running custom object detector

Maintaing Social Distance

Sending an SMS to authority

***System Architecture:***

YOLOv4

MODEL

Creating A dataset with a single class “Person”

Using OIDv4 toolkit

OIDv4 toolkit

converting

YOLO formate

.txt file

A image with

.jpg file

A image with

.jpg file

creating

detection using YOLOv4 pre-trained weights

Test file

(test.txt)

Training file(train.txt)

using google colab

Train custom file using

Yolov4\_train.cfg

Run the custom model and result of .jpg or .mp4 file

Creating .cfg file

For social distance maintaining

Maintaining social distance with DeepSort technique

Sending an SMS to authority if social distance is hampered

**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.

**3.9**

**4. 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:**

**5. 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.

**6. 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] |  | 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. |
| [2] |  | J. Zicong, Z. Liquan, L. Shuaiyang and J. Yanfei, Real-time object detection method based on improved YOLOv4-tiny, 2020. |
| [3] |  | A. Bochkovskiy, C.-Y. Wang and H.-y. M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020. |
| [4] |  | A. "darknet," 2020. [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed 24 November 2020]. |
| [5] |  | "Plurale," [Online]. Available: https://www.plurale.com.br/upfiles/fckeditor/1\_muuuuuuuuuuuuu.jpeg. [Accessed 24 November 2020]. |

|  |  |
| --- | --- |
|  |  |

Real Time and Accurate monitoring of social distance during COVID-19 situaton.