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

Shourove Sutradhar Dip[1], Sajjad Waheed

Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University, Santosh, Tangail 1902, Bangladesh
E-mail: shourovdip@ieee.org[1]

1. Research Area:

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].

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.

3. 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 works faster and it's 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 [2]. 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 matching 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" is 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 [3]. 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 [4] 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 file 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 [2]. This particular configuration file also contains 3 YOLO layers at the last which describes the architecture of YOLO [2]. 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 [2]. For this custom model, where only one class was used, the number of filters were (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:

In order 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 [4] and .mp4 files which were uploaded previously, using a previously mentioned facility with a higher GPU capability.

4. Results and Analysis:

As it can be seen in the image below, the custom YOLOv4 detector had a slight (from 1% to 2%) decrease in accuracy. But the detection time was less with the custom YOLOv4 detector. The original YOLOv4 took 79.030 milliseconds to complete the detection but the custom YOLOv4 detector took 77.392 milliseconds to complete the detection on the same image. Although there was a slight decrease in the accuracy, the detection was done faster in case of the custom YOLOv4 detector. After testing .mp4 files, the outcome was the same with a slight decrease in the accuracy but less time to complete the detection. As the decrease in accuracy is just 1% or 2%, there will be no problem in detecting a person. But as the speed of detection has increased, this will help to detect humans faster in real-time and perform operations faster, which is more beneficial.

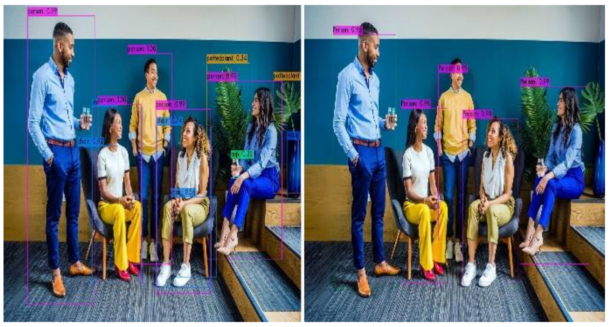

Figure-1: Detecting various objects in an image [4] using the original YOLOv4 object detector (left) and using the custom YOLOv4 object detector (right)

5. References:

[1] P. Mahto, P. Garg, P. Seth and J. Panda, "REFINING YOLOV4 FOR VEHICLE," *International Journal of Advanced Research in Engineering and Technology (IJARET),* vol. 11, no. 5, pp. 409-419, 2020. DOI: 10.34218/IJARET.11.5.2020.043

[2] A. Bochkovskiy, C.-Y. Wang and H.-y. M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020.

[3] AlexeyAB, "darknet," 2020. [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed 30 October 2020].

[4] "Plurale," [Online]. Available: https://www.plurale.com.br/upfiles/fckeditor/1_ muuuuuuuuuuuuu.jpeg. [Accessed 30 October 2020].