[1]

Social Distance Tracker Using Computer Vision (June 2022)

Mohamed Nasr, *Student 201801675, Zewail City*, and Nada Mostafa, *Student 201800225, Zewail City*

***Abstract*—With the spread of the COVID-19 Corona pandemic, maintaining a safe physical distance of two meters between people has become one of the best preventive measures. In this project, we propose a system for live detection and tracking of social distancing to monitor the practice of social distancing in a crowd.**

The first step for building this system will be using an object detection model to identify people in each frame, then applying an image processing and tracking algorithm for people tracking, after that, we use the pixel distance to set the social distancing and get the distances among the crowd, and if the distance is found to be more than 1 meter between 2 persons, the system recognizes this case and it should identify people who violate the social distancing.

***Index Terms*— Keywords: People Detection, Social distancing, and Tensorflow.**

I. Introduction

SOCIAL DISTANCE TRACKER is needed in various fields, recently, it’s required in COVID-19 pandemic handling. However, it is also needed in various fields like surveillance systems and detecting video records.
Social distancing is one such concept that has gained prominence in recent years as a result of COVID-19. People are required to keep a safe distance between themselves in order to avoid the transmission of this fatal sickness. In the middle of this crisis, We decided to create a basic Social Distancing Detector that could track the process of social distancing in a crowd.

In this project we used Deep Learning based COCO (R-CNN Model) Pretrained model for object Detection and the code in this project depends mainly on TensorFlow and OpenCV and Python for image processing and Object Detection and Centroid Tracking Algorithm for object tracking to detect if a crowd is practicing Social Distancing or not and we check that using a sample video.
The code is divided into two main parts:

[1]

- Object detection:

We used a COCO pretrained model which is a machine learning model for object detection to detect all the objects in each frame of the video, then we gave every object a box (which represents a part of the image where a particular object was detected), a score (which represent how the level of confidence for each of the objects), a class (which represents a label defines the object), and a number. So it is used to return the person's prediction probability, bounding box coordinates for the detection, and the centroid of the person.

- Distance calculation:

It is used for finding the physical distance between the detected humans in each frame of the video and checking if they violate the social distance or not. We calculate the centroids of the rectangles that bound the detected human by getting the midpoints in both x and y directions.
Euclidean distance is then computed between all pairs of the returned centroids And based on these pairwise distances, we check to see if any two people are less than a certain distance that is allowed or not, and based on that we decide whether to change the color of the bounding box or not.

II. Methodology

This project is made for video detection and tracking of social distancing. The code in this project uses TensorFlow and OpenCV in Python.

This project is inspired from the works of Landing AI and Airpix.

Our solution code consists of mainly two classes (DetectPeople) and (CalcDistance) and the (Main) function that we will discuss in some detail in the coming part.

First we have a class that we called (DetectPeople) class and it is used for detecting all the objects in an image(each frame of the video) and giving each and every one of them a box (which represents a part of the image where a particular object was detected), a score (which represent how the level of confidence for each of the objects), a class (which represents a label defines the object).

This class consists of only one main function as well as the constructor of the class which is given a trained model as an argument to use for the detecting process. It initiates a TensorFlow session that allows executing graphs or part of graphs as It allocates resources (on one or more machines) for that and holds the actual values of intermediate results and variables. It also sets a box, a class, a score, and a number for each object. It also sets definite input and output tensors for the detection graph.

The main function in this class is called (frames) and it is responsible for the actual detection process. It takes an image as input and it expands its dimensions at first since the trained model expects images to have shape of: [1, None, None, 3], Then it starts the actual detection. It uses operate.run() function to call operations and get (boxes, scores, classes, numbers) for the objects in the image and return them as an output.

The second class is called (CalcDistance) class and it is used for finding the physical distance between the detected humans in each frame of the video and checking if they violate the social distancing or not.

This class consists of three main functions besides the constructor. The constructor initiates some variables like the threshold, the average height of the box, a list for the centroids of the rects, and a list for the boxes in the image. Then it iterates over all the boxes/objects in the image, and if this object was found to be a human (has a class of 1), it appends it to the lists and bounds it with a rectangle of green color.

The first main function we have is called (center) and it is responsible for calculating the centroids of the rectangle, that bounds the detected human, it takes the boundaries of the rect as arguments and does calculations to get the midpoints in both x and y directions and return them as the center of the rect.

The second function is called (distance) and it is responsible for calculating the distance between two rects in pixels. It takes the x coefficients of the boundary rects and the indices of the two rects and finds the euclidean distance between them and returns that distance.

The third function is called (distchecker) and it is responsible for checking if the distance between every detected human and the other detected humans in the frame is larger than the social distance allowed or not by calling the function (dist) between every two rects, and based on that it changes the colors of the boundary boxes to be red if the distance exceeds the allowed distance.

For the main body that will be executed, first, we defined the model path, which is the trained model that we use for detection. Then, we make an object from the (DetectPeople) class and pass that model path to it as an input, then we define the video that we will be working with. After that we loop over each and every frame of the video and verify if physical distancing rules are followed or not and that's by making an object from the (CalcDistance) class and passing the frame to it as an input, and then calling its (distchecker) function, then we show the frame as a final step and go for the next one.

III. Pre-trained Model Selection

It is very hard to train a model from scratch. This needs more resources and hardware. Thus, it is a waste of time.

To avoid this time consuming process, We work with a pre-trained model.

When choosing a pretrained model to fine tune for a given object detection application, there are several factors to consider, including the dataset on which the model was trained, expected input/output, accuracy, computational complexity, architecture, size, and so on. A trade-off between speed and accuracy may seem different depending on the application, and modeling size and complexity are especially crucial in business mobile apps.

We were choosing from these three model out of these three papers:

Model Name Speed (ms) mAP Outputs Released

1) ssd_mobilenet_v2_coco 31 22 Boxes 18-04-02 (Sandler et al., 2018) [1]
2) faster_rcnn_inception_v2_coco 58 28 Boxes 15-12-02 (Szegedy et al., 2015) [2]
3) faster_rcnn_nas_lowproposals_coco 540 43 Boxes 17-07-21 (Zoph et al., 2017) [3]

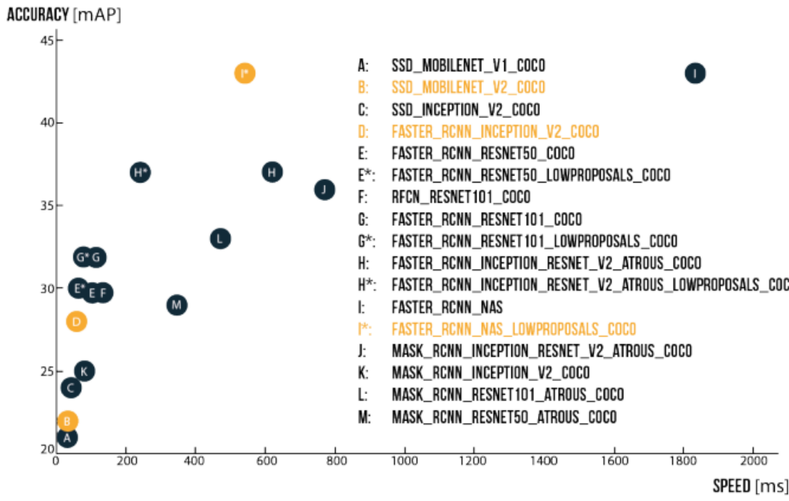


figure (1) - Comparison of available pretrained object detection models in the TensorFlow Object Detection Model Zoo (as of 2018- 05)

According to a master thesis "A Framework for Generative Product Design Powered by Deep Learning and Artificial Intelligence" [4]

it found that faster_rcnn_inception_v2_coco_2018_01_28 is the best model in total amongst them.

So, we chose: faster_rcnn_inception_v2_coco_2018_01_28

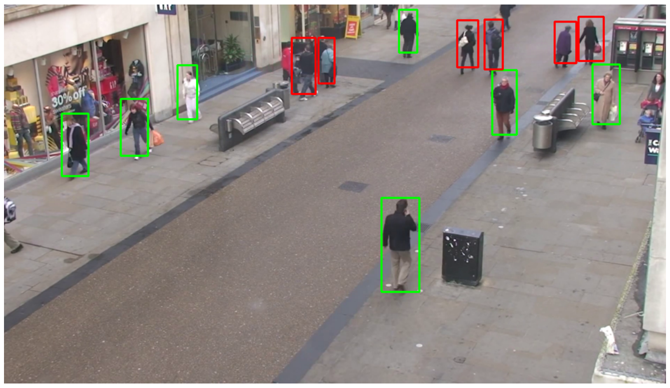
IV. Results

We succeeded in detecting violations less than 1 meter between people.

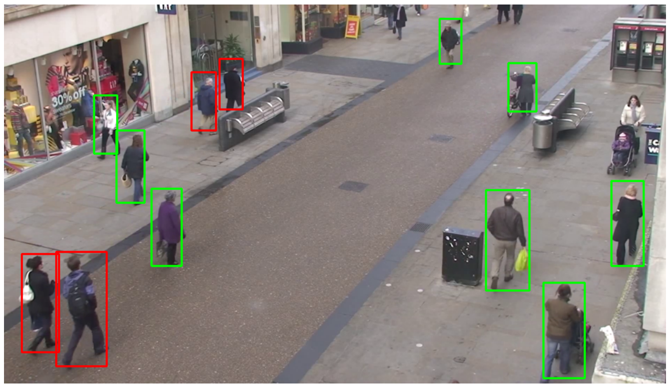
We examined it in 2 videos. We are showing a sample of its results.

First video results

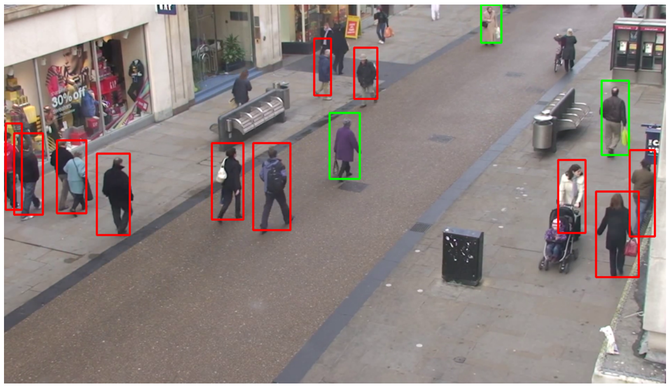
1-



2-

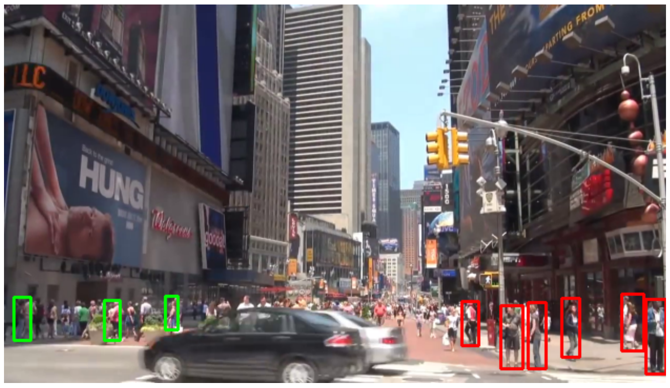


3-

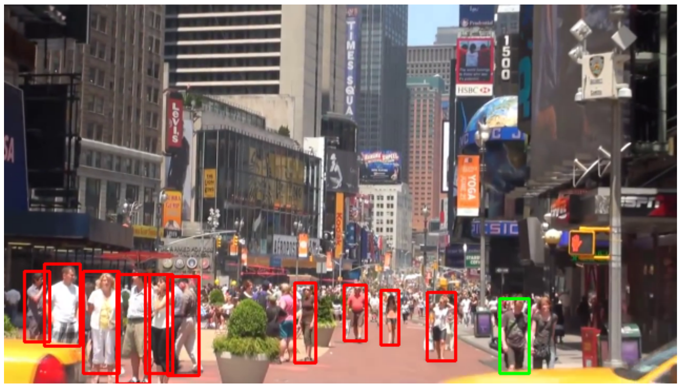


Second video results:

1-



2-



V. Discussion

The results are very satisfying. Although this, There can be some modifications in the future like using transfer learning to build stronger pre-trained models. However, these results can be used directly in live surveillance systems, or recorded videos.

References

[1] Sandler, Mark et al. (2018). “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: CoRR abs/1801.0. arXiv: 1801.04381. URL: https://arxiv.org/pdf/ 1801.04381.pdf.

[2] Szegedy, Christian et al. (2015). “Rethinking the Inception Architecture for Computer Vision”. In: 2016 {IEEE} Conference on Computer Vision and Pattern Recognition, pp. 2818–2826. arXiv: 1512 . 00567. URL: http : / / arxiv . org / abs / 1512 . 00567.

[3] Zoph, Barret et al. (2017). “Learning Transferable Architectures for Scalable Image Recognition”. URL: http://arxiv.org/abs/1707.07012.

[4] Alexander NILSSON, Martin THÖNNERS (2018), A Framework for Generative Product Design Powered by Deep Learning and Artificial Intelligence.