Social distance for COVID-19 prevention with YOLOV4

Mauricio Ariel Paredes Jauge
Leo Jhoel Alvarez Choque
Belinda Alcon Sullcani
Ana Raquel Martinez Carballo
Carrera, e.g. Carrera de Ingeniería Mecatrónica
Universidad Católica Boliviana "San Pablo"
La Paz, Bolivia
address [at] ucb.edu.bo

***Resumen*—The arrival of covid-19 had a great impact on society, where people must take security measures to prevent a rapid spread. One of those security measures would become the social distancing that people must have to avoid increasing the number of infections. The project will focus on carrying out a program that is capable of detecting the distance between each person to see if this security measure is met. For this, it was decided to use the YOLOv4 method, which will help with the detection of objects in a fast and accurate way. The already trained model was retrained. For this, a dataset of its own containing 780 images was created, this dataset was created in Roboflow. A series of tests was carried out to evaluate the model. The respective metrics were found, where a precision of 78.68 % and an average precision of 0.77 were obtained.**

I. Introduction

With the arrival of COVID-19, daily life as we knew it completely changed, people were forced to take enough security measures, like the governments of different countries, they issued a rigid quarantine for the purpose to avoid further spread. COVID-19 is known to be transmitted from person to person when someone with the virus coughs or sneezes. Therefore, people who maintain close contact with others can contract the virus, that is why the idea of carrying out this project arose, which through artificial vision and using tools to detect objects as it is YOLO (You Only Look Once), an algorithm was made in python that is capable of detecting the minimum distance a person can be with respect to another. According to the World Health Organization (WHO), it is recommended to maintain a distance of at least 1 meter with anyone, all this to avoid the rapid spread of the virus.

Nowadays, social distance seems to have a much greater importance than before and it is also one of the best ways to stop the spread along with masks. Artificial vision has had excellent results in solving complex health problems, showing considerable potential in this area and may be able to contribute to the monitoring of social distance, a neural network (COCO dataset) was used already trained, that same network was retrained again with our own dataset, which contained images of people, all this with the aim of having a better result for the detection of people at the time of monitoring distances.

The tool used for the respective detection was YOLOv4, which uses deep learning and convolutional neural network (CNN), has a fairly efficient speed, which allows it to detect objects in real time in videos up to 30 fps (frame per second). To carry out a detection, it predicts n quantities of bounding boxes and calculates the probability of each of them generating different boxes, once the predictions are obtained, the boxes that are below a limit are eliminated and thus leave the most exactly. There are implementations for YOLO, the one used in the project was the Darknet implementation, it is a whole framework for neural networks, used in different objectives in addition to YOLO, it has the characteristic of supporting an environment with GPU.

COCO dataset was created with the aim of representing a number of objects that we normally find in our day to day. COCO dataset has labels, it provides data to train supervised models that are able to identify common objects in the dataset. Once the model is trained it can be adjusted so that it can learn other tasks with a custom dataset. It should be noted that COCO dataset has 80 different classes and approximately 121,000 images. On the other hand, it has multiple tasks for artificial vision, such as: Object detection, segmentation and keypoints detection.

II. Related Work

It is important to emphasize that the detection of people in sequence of images is one of the most important secondary branches in the field of object detection and computer vision. Many research papers that focused on detecting and recognizing people have quite a few complications with accuracy, plus limited function was always a problem.

Convolutional Neural Networks (CNN) play a very important role in the extraction and classification of complex objects, including the detection of people. The goal of CNN is to learn features of a higher order using the convolution operation, in said convolution, each output pixel is a linear combination of the input pixels. By developing a faster CPU and GPU runtime environment, CNN enables more accurate detection and faster comparison of common models. However, the time it takes to train, the speed of detection and obtaining better precision are methods that need to be improved.

At the time of carrying out this project, information was sought and works related to the subject were found, in a repository made by Wilson Córdova, which likewise seeks to monitor people's distance to comply with the social distancing protocol, uses a previous version of YOLO (version 3) and a previously trained COCO data set, to be able to carry out the respective detection of people, for this he collected video sequences and images of people letters in a photograph to carry out tests with the model, in his case I do not rerun the model. Also, it is worth mentioning that it used the TensorFlow library, which has the objective of building simple compilations of models to perform machine learning. In order to evaluate its, the author of the repository used classification metrics, evaluating the accuracy by means of a confusion matrix, calculating the precision and sensitivity.

Another repository that is related to the project was the one mede by Alexander Espinosa, which had the objective of developing a graphical interface for the detection of social distancing using artificial vision. The author took as options to use different types of methods such as: R-CNN, Fast R-CNN, Faster R-CNN, SSD (Single Shot Detector) and YOLO (You Only Look Once). Among all these methods, the author saw that the most suitable for this type of project was the YOLOv3 method because it has a faster detection speed than the other methods mentioned, in addition to allowing objects to be found in real time. On the other hand, the author used the method of classification by Euclidean distances, the realization test was carried out with the Darknet-53 implementation of ImageNet and the data set with COCO.

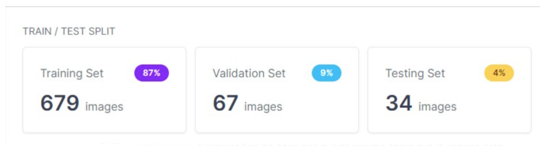
Finally, unlike the related works, for our project we used the YOLOv4 method since it is an improvement compared to the previous version because it improves the detection precision without the need to increase the inference time. In addition, we use our own data set to train the model so that it can have a higher detection, obtaining more efficient results.

III. Data

First, an already trained network was used with COCO dataset (Common Objects in Context), which, as mentioned before, is a dataset that already has labeled objects, provides data to train supervised models, has 80 classes and a total of approximately 120,000 images. It should be noted that a code was introduced in python to be able to separate the images of the class called 'person', some of those images were chosen to transform them and by default they were assigned to a class '0', which refers to the class for people in COCO dataset .

Furthermore, the network was trained once more, in this case with our own dataset. This dataset was created in Roboflow, which included 780 images of a single class restricting the output to a class called "0"with a label called "person". Of the 780 images, 340 are original images with labels made in CVAT, a platform that allows images to be labeled for their respective use in training datasets.

Some images were transformed with a 15º right turn, adding a grayscale filter. Once the images were obtained, they were divided, in the training set we had 679 images (87 %), in the test set with 34 images (4 %) and finally the validation set had 67 images (9 %).


TRAIN / TEST SPLIT

Training Set 87%	Validation Set 9%	Testing Set 4%
679 images	67 images	34 images

Figura 1. Train,Test and Validation

Finally, thanks to the use of Roboflow, a direct download link was obtained, which contained the entire dataset created with the 780 images, the generated link was placed in python colab in order to have a direct download.

IV. Methods

The method that was used to solve the problem raised above was the following:

- 1) Preparation of a data set for subsequent training.
- 2) Training of a convolutional neural network (Yolov4).
- 3) Recognition of people within each frame in a video.
- 4) People tracking with SORT (simple online and realtime.
- 5) Determination of distances between individuals with the help of bounding boxes.
- 6) Determine if a person is in danger of contagion or not.

We carry out each of these steps considering this a reliable and practical option to solve the problem posed.

Clearly to detect if a person is very close or far enough from another or others it is necessary to first detect each person in the scene so a neural network for detecting these is the most reliable option. When we talk about convolutional neural networks for this purpose it is impossible not to think of YOLO (You Only Look Once) which is a network for object detection in general, if we compare this network with some other (ex. SSD) we usually choose YOLO thanks to its level of precision or speed.

Figura 2. YOLO vs SSD

Yolo v4 is the improvement of previous versions since it improves the precision of the detector without increasing the inference time, this has an accuracy from 10 % to 12 % higher than its predecessor.

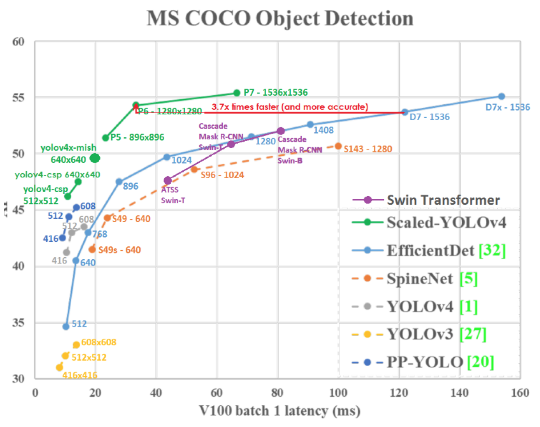


Figura 3. MS COCO Object Detection

On the other hand, we conclude that obtaining a previously trained Yolo v4 network to later train it again with a better selected data set is a very feasible idea since the network will be able to learn faster about the new data set with previous knowledge in detecting people, once all people in each frame are detected, bounding boxes are generated thanks to the OPENCV library which provides us with an easy use to determine safe or dangerous distances.

During the tracking of the people, a Kalman filter was used, this filter works as an estimator making a prediction of the position values and the bounding boxes, this gives us an advantage since without the use of this the tracking would not be so fast if we talk about a real-time implementation of our algorithm.

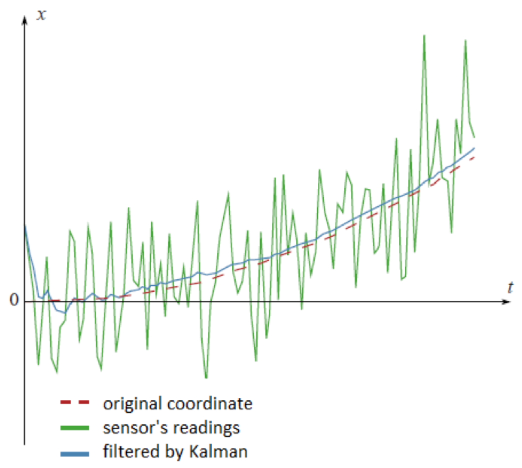


Figura 4. Kallman Filter

On the other hand, there are different alternatives to solve this problem, one of the methods that were analyzed during the process was the use of Pytorch for network training, however the framework used for training was DARKNET, this since it is the most widely used framework for Yolo training in addition to the ease of obtaining it on a platform like Google, we can also mention that Yolo and Darknet complement each other very well since it has a more robust support for CUDA and CUDNN.

V. Experiments

For the training part, a variety of batches will be used in order to see which was the most optimal. Below this paragraph, you can see a comparative table between the number of batches used, number of images, average IoU, detection time and accuracy.

Number of Batches	Number of images	Average IoU	Time	Accuracy
1300	780	47 %	2 seg	55 %
1850	780	51,85 %	2 seg	76,13 %
2000	850	55,85 %	1,7 seg	76,20 %

Tabla I
HYPERPARAMETERS

3 experiments were carried out for the training of the images, it was decided to choose the one that had 1850 batches, compared to the training with 2000 batches, it has a much shorter time at a precision almost similar to that of 2000 batches.

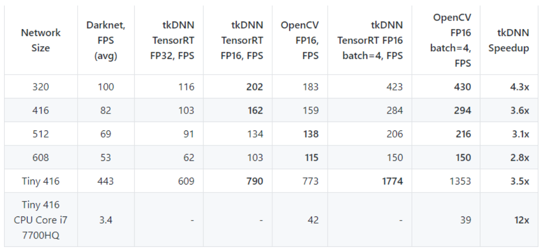
The values obtained in the previous table can be compared with training weights carried out with different datasets from different sources, of which 3 will be mentioned:

- yolov4x-mish.cfg: - 640x640 - 67.9 % mAP@0.5 (49.4 % AP@0.5:0.95) - 23(R) FPS / 50(V) FPS - 221 BFlops

- yolov4-tiny.cfg: - 40.2 % mAP@0.5 - 371(1080Ti) FPS / 330(RTX2070) FPS - 6.9 BFlops

- enet-coco.cfg: (EfficientNetB0-Yolov3) - 45.5 % mAP@0.5 - 55(R) FPS - 3.7 BFlops

To try to optimize the training time of the network, a possible option would be the implementation of a TensorRT. However, for optimal performance of this accelerator a GeForce RTX 2080 Ti card is required. Due to the limitations of google colab, the implementation of TensorRT was not possible.



Network Size	Darknet, FPS (avg)	tkDNN TensorRT FP32, FPS	tkDNN TensorRT FP16, FPS	OpenCV FP16, FPS	tkDNN TensorRT FP16 batch=4, FPS	OpenCV FP16 batch=4, FPS	tkDNN Speedup
320	100	116	**202**	183	423	**430**	**4.3x**
416	82	103	**162**	159	284	**294**	**3.6x**
512	69	91	134	**138**	206	**216**	**3.1x**
608	53	62	103	**115**	150	**150**	**2.8x**
Tiny 416	443	609	**790**	773	**1774**	1353	**3.5x**
Tiny 416 CPU Core i7 7700HQ	3.4	-	-	42	-	39	**12x**

Figura 5. Comparison between TensorRT and Darknet

In addition, tests were carried out with a series of videos, taking into account some parameters in which we can define the minimum distance between people located in front and to the sides.

Number of video	Forward	Behind
video 1	169	45
video 2	160	60
video 3	160	62
video 4	155	45

Tabla II
MINIMUM DISTANCE BETWEEN PEOPLE

- View video 1: Top view from a corner, at an elevation of approximately 4 meters.

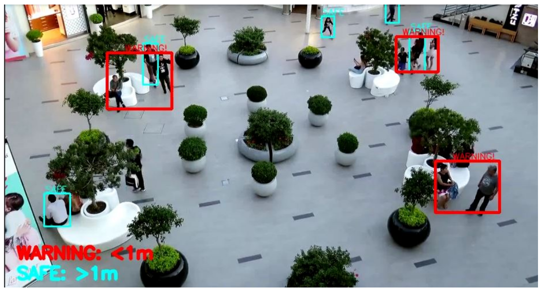


Figura 6. Video 1

- View video 2: Top view to the left located on the third floor.

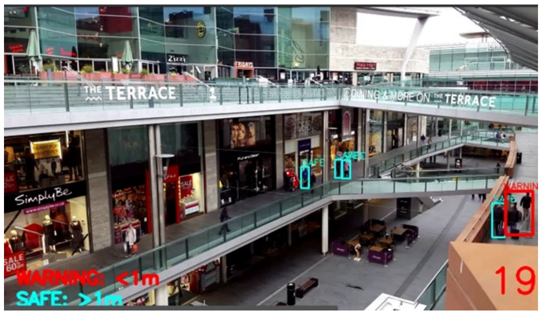


Figura 7. Video 2

- View video 3: Top view to the left, at an elevation of approximately 3 meters above the second floor.

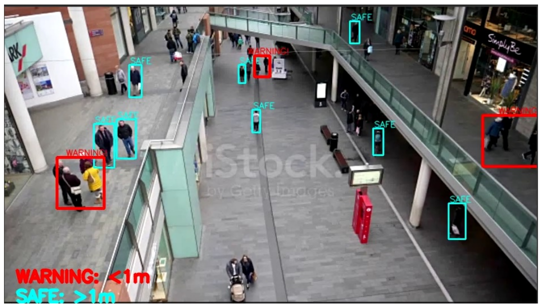


Figura 8. Video 3

- View video 4: top view from a corner, at an elevation of approximately 4 meters.

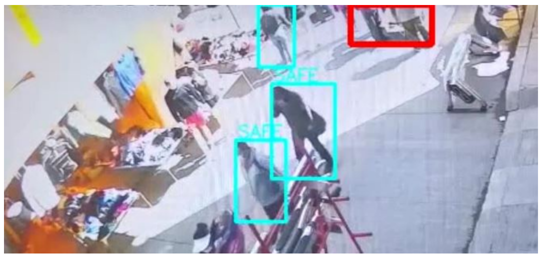
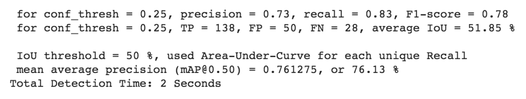


Figura 9. Video 4

Once the training is done, we find some metrics to determine the efficiency of the model. An accuracy of 73 % and a recall equal to 83 % were obtained. The F1 score metric is a function of precision and recall. The closer the F1 value is to 1, it means that there is a better balance between precision and recall. The F1 score obtained was 78 %. In addition, a mean average precision equal to 76.13 % was obtained. For the confusion matrix we have a TP = 138, FP = 50, FN = 28. Finally, the detection time is equal to 2 seconds.



```
 for conf_thresh = 0.25, precision = 0.73, recall = 0.83, F1-score = 0.78
 for conf_thresh = 0.25, TP = 138, FP = 50, FN = 28, average IoU = 51.85 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.761275, or 76.13 %
Total Detection Time: 2 Seconds
```

Figura 10. Metrics

VI. Conclusions

Thanks to the tests carried out at different points throughout the progress of the project, and making a respective evaluation of the metrics obtained, we can conclude that the project carried out with the help of artificial vision and machine learning is too effective for this type of problems such as It is social distancing, to avoid infections due to the health crisis of COVID-19.

Throughout the project, a lot of training was carried out with its own dataset in order to improve the performance and accuracy of the network, in order to obtain an excellent detection when evaluating the monitoring of social distancing. Despite obtaining the expected results, there are other methods that can facilitate the detection of distance between people.

However, the method used which is YOLOv4 has a higher detection speed than other methods, retraining the COCO dataset with your own helped enough for the detection of people.

VII. References

AlekeyAB. (15 de 05 de 2017). Darknet. Obtenido de github.com: https://github.com/AlexeyAB/darknetyolo-v4-in-other-frameworks

Goncharov, I. (27 de 10 de 2019). Datasets: ASAP. Obtenido de github.com: https://github.com/ivangrov/Datasets-ASAP

Laut, J. (17 de 02 de 2019). Obtenido de How to train YOLOv4 on a custom Dataset in Darknet: https://www.youtube.com/watch?v=N-GS8cmDPog

Goncharov, I. (27 de 10 de 2019). Read COCO Dataset for Bounding Boxes (including YOLOv3 format) in Python. Obtenido de youtube.com: https://www.youtube.com/watch?v=uq4F9PIeZB0t=961s

Brasileroth75. (18 de 06 de 2020). COVID-SOCIAL-DISTANCING-DETECTION. Obtenido de https://github.com/basileroth75/covid-social-distancing-detection

Ronchi, M. R. (2020). Custom Object in Context. Obtenido de https://cocodataset.org/

Valdez, A. E. (2021). Interfaz gráfica para la detección de distanciamiento social usando visión artificial en software libre. Colombia: Facultad de Ingeniería Electrónica.

Eras, W. S. (2020). Implementación de un sistema de reconocimiento de distanciamiento social como medida preventiva para COVID-19 usando Deep Learning. Machala: Universidad Técnica de Machala.