Computer Vision Project
Social Distance Detection

1st Bustillos Vila, Daniel Arturo
Ingenieria Mecatronica
Universidad Catolica Boliviana "San Pablo"
La Paz, Bolivia
daniel.bustillos@ucb.edu.bo

2nd Fernandez Testa, Sergio Rodrigo
Ingenieria Mecatronica
Universidad Catolica Boliviana "San Pablo"
La Paz, Bolivia
sergio.fernandez@ucb.edu.bo

3rd Molina Montes, Mariana
Ingenieria Mecatronica
Universidad Catolica Boliviana "San Pablo"
La Paz, Bolivia
mariana.molina@ucb.edu.bo

4th Quinteros Terrazas, Osvaldo Cesar
Ingenieria Mecatronica
Universidad Catolica Boliviana "San Pablo"
La Paz, Bolivia
osvaldo.quinteros@ucb.edu.bo

***Abstract*—COVID-19 pandemic, the ongoing global pandemic of Coronavirus disease (COVID-19) was the origin of hard times for a major share of the entire world population. Many precautions and recommendations were made on account of the global effort to stop the spread of the virus. The practice of social distance is an important recommendation, this paper proposes a technical approach in order to help control a safe distance in public spaces through video-cameras and video recordings. Social Distance Detection Project was developed to control distance between two persons or more, the project first approach was made using YOLO framework, it is actually called the faster algorithm detector. Nonetheless YOLO is not the most accurrate one. The project main developped approach was a combination between MobileNets and Single Shot MultiBox Detecto. The challenge of the project was to develop, train and execute a program that calculates and plots the distance between people in order to prevent COVID contagion in the community.**

***Index Terms*—Distance, COVID-19, Contagion prevention, Social distancing, Combined approach.**

I. Introduction

The coronavirus pandemic is a worldwide problem that has been affecting people of all ages. Currently protocols to prevent infections and the spread of the virus among the population have been developed, one of the most efficient recommendations is social distancing. This project aims to automate the control and supervision of social distance between one person to other in public places. The objective is to develop a program which detects, through video security cameras and video recording whether the distance between people is appropriate or whether if it is breaking Wold Health Organization (WHO) minimum distance recommendation.

For the purpose of developing a great implementation of methods and algorithms it is important to remember the definitions of Convolutional Neuronal Networks (CNN). It is a class of deep neural network, in deep learning, that can take an image and detect features of it. CNN has multiple artificial neural layers, and works just like human neurons. All artificial neurons have its own weight, all of them receive multiple pixel values, make a multiplication of each pixel's color values with the neuron weight and finally sum that values and produce an activation value. With this mathematical process each neuron catch important features that become the detected feature of the processed image.

This project uses techniques of computer vision, a combination of state-of-the-art algorithms, frameworks, methods and architectures that result in a fast and efficient solution for social distance detection, applicable in many systems and places, this project consist in an synthesis of a streamlined architecture and a single-shot detector method for multiple categories faster and more accurate than similar state-of-the-art methods.

II. Related work

A. Background on MobileNets

MobileNets is a project which develop different classes of models to computer vision and embedded systems applications, object detection is its main achieve. The project present an innovate form to apply CNN algorithms, is interesting the results developers obtained.

The objective of creating new, small and efficient neural networks has always been a desire of developers, whose different approaches are generally related to compressing pretrained networks or also training small networks in a straight forward way. MobileNets architecture is focused on having optimized latency and also make small NN, not only focusing in size like other architectures but also considering speed.

Its is primarily build from depthwise separable convolutions as introduced in "Rigid-motion scattering for image classification" by L. Sifre in his Phd thesis. Also used inception models like "Batch normalization: Accelerating deep network training by reducing internal co-variate shift", by S. Ioffe and C. Szegedy. Other methods that have given shape to how MobileNets is nowadays are, the flattened networks of seen in "Flattened convolutional neural networks for feedforward

acceleration" by J. Jin, A. Dundar, and E. Culurciello. Depthwise concepts were based on "Xception: Deep learning with depthwise separable convolutions" by F. Chollet. Compression based on product quantization was first seen in "Quantized convolutional neural networks for mobile devices" by J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Another training method aimed for small networks is "Distilling the knowledge in a neural network" by G. Hinton, O. Vinyals, and J. Dean. and it is in fact complementary to the MobileNets approach.

B. Social Distancing Detector using OpenCV a similar project

Sanju Mehla is the autor of a project in the same area. The steps used to build a social distancing detector in his projects include applying object detection to detect people in a video stream, computing the pairwise distances between all detected people, and finally, based on these distances, check and see if any people are not complying with the social distance restriction, that is to say having a distance less than N pixels apart, being N the desired restriction.
This project implementation relies on pixel distances, and it is said in its documentation that this model is consequently not as accurate as some applications need it to be. It uses YOLO object detector with OpenCV, and as well as the library's dnn module, to load the YOLO net into the memory.

III. Data

Social Media Distance was desired to be robust and get the best possible results. In order to achieve this, COCO dataset was applied to get and train the model with images that contained the features the project required. Common Objects in Context (COCO) is a open source dataset which contains 1.5 million images with about 80 labeled categories for different applications. For this project, a pre-model training with COCO dataset was used, SSD300*, the model was trained for bigger or general applications, however, models pre-trained with that characteristic can be used for specific purposes. This first training got a parameter of object detection accuracy (mPA) of 0.68, to increase this value, a second tuning training phase was applied, with VOC0712 (Visual Object Classes) . Pascal VOC is widely used as a benchmark for object detection, and other tasks. With this second phase of training, the model got a 0.72 of mPA. With this value, it was possible to continue developing the project.

Fig. 1. In the figure is represented an examples of COCO dataset that are useful for the project

IV. Methods

The team's approach was initially based on the method You only Look Once (YOLO), despite the fact that the first stage of the project had already been developed with this method , the partial results obtained were not satisfactory and the team decided to use Single Shot Detectors (SSDs) instead. Both methods are really popular and widely applied in projects of object detection, nonetheless quite a few differences were found among the two. In order to improve the speed of detection and its efficiency as well as its accuracy the team made a deep research in ImageNet and networks architectures, founding that MobileNets is a great approach viable not only for computer systems but also for embedded systems, an interesting application in special for one of our team members future expertise field.

Common Objects in Context (COCO) is a useful dataset meant to be used in object detection, segmentation and captioning. This was the chosen dataset in the project and whereas a variety of factors where changed in the project development, the implementation of this dataset remained untouched.

First approach considered was You Only Look Once v3 (YOLO) an unified real-time object detection, it makes use of an only neural network, this makes the approach easy optimized, YOLO predicts various bounding boxes and the class probabilities of these boxes. As a matter of fact, the model takes the boxing framing process as and regression problem and proceeds accordingly, additionally, this model can be trained directly on full images. Therefore YOLO v3 is ideal for general applications.
In the teams first effort to develop the project, the Euclidean distance between the center or identified boxes was found through a created function, and the the algorithm checked if the distance violated the minimum social distance established. Results were obtained for frames (images) only, but the team realized that these were not correct in some images, and for

some, people was not correctly recognized.
Some seen and studied limitations of YOLO are that it imposes strong spacial constraints, this method is not quite accurate for small objects or in new scenarios not previously trained, furthermore it can be inferred that for the rough implemented features it has a major downsampling, sometimes it even has incorrect localizations.

Single Shot MultiBox Detector (SSD) is a method developed by Google, it also uses a single deep neural network to be performed, SSD scores the presence of objects in the tested image, and samples the output space of the boxes in a discrete way. This method was originally testes in COCO, VOC, PASCAL and other important datasets having a great accuracy, reason why it seemed like a great option for the project. After a research, it was evident that even though YOLO was a bit faster is not as exact as SSD. The team considered that for real applications of the project exactness was a main consideration. For a further explanation, SSD makes use of different feature maps, combined predictions and diferent resolutions and scales. As the implementation of the method was made, it was noticeable that SSD had balance between speed and accurracy. SSD was implemented in the developed pretrained model based on an MobileNet-SSD model with free MIT license.

The table below shows a general comparison between YOLO and SSD, this table was taken from the official SSD repository in github.

System	VOC2007 test *mAP*	FPS (Titan X)	Number of Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO (customized)	63.4	45	98	448 x 448
SSD300* (VGG16)	77.2	46	8732	300 x 300
SSD512* (VGG16)	**79.8**	19	24564	512 x 512

Fig. 2. General SSD and YOLO comparison.

The table below shows is another comparison between general characteristics of the two frameworks, taken from a recently published article in the International Journal of engineering research and technology (IJERT).

Sl.no.	Parameters	YOLO	SSD
1	Model name	You only look once	Single shot multi-box detector
2	Speed	Low	High
3	Accuracy	80.3% High	72.1% Low
4	Time	0.84 - 0.9 sec/frame	0.17 - 0.23 sec/frame
5	Frame per second	45	59
6	Mean averge precision	0.358	0.251

Fig. 3. Characteristics comparison SSD an YOLO.

The team model was trained to detect and then generate boxes colors for each declared class. In order to be able to implement SSD this need to be installed, prepared, trained, evaluated and compiled in a caffe model following the instructions of the official SSD github repository. Fortunatelly the developers of the method offer free access to models already trained with different dataset to help the community reproduce the same results as in the published paper. This is where COCO model "trainval35K SSD300*" was obtained from. The normalization was already done by the referenced authors of the pretrained MobileNet-SSD model with free MIT license. All required code from SSD official repository was pulled in order to generate the MobileNet-SSD model.

MobileNet is an efficient network architecture based of a streamlined architecture, this architecture uses depthwise convolutions in order to build a lightweight neural network. The model is based in first instance on depthwise separable 3x3 convolution layers. This layer is a full convolution, all layers are then followed by a batch normalization and ReLU layers, after a 1x1 pointwise convolution is applied.
In the following table it is shown the nature of the MobileNet body architecture, this table was obtained from the official paper publication of MobileNet.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig. 4. MobileNet body architecture.

The implementation of MobileNet-SSD detection network was made through a implementation in cafe with a VOC "07+12 SSD300*" model obtained from the same place as "$trainval35KSSD300*$". This networks has a theorical 72.7 mAP and the train and deploy caffemodel are available in the free MIT licence MobileNet-SSD implementation.

A key module used in the development of the project was dnn module form OpenCV. This module was used in the building of the team's object detector. Luckily this module is laready part od OpenCV's contemplated library. In order to have a proper understanding of this module and what it does, the official OpenCV tutorial "$LoadCaffeframeworkmodels$" was consulted. It is

shown in this tutorial a clear how to used pretrained models in C++, there is no much difference betwen the login in C++ an python, so key ideas were studies, moreover, some other official OpenCV realted and recommended tutorials were used as reference.

The team did all this implementations in order to calculate the distance between boxes centers and determine whether recognized people are not complying with the social distance regulations recommended to prevent covid according to the World Health Organization (WHO).

Computer vision course lectures and additional material meant a great advantage in understanding the models and their applications. It is remarkable that the knowledge acquired during the semester and specially the last three lectures were some of the reasons the team decided to implement a better approach in the present project, and consequently obtaining much better results a far more complete work in this area.

The team states that in this project has been found a great if not the best combination of methods, architectures and parameters to make a Social Distance Detector. Having done a proper research, explanation and comparison as well as have taken different approaches, the necessary evidence was compiled to infer that this solution is not only theoretically correct but is also practical and applicable in the real situations.

V. Experiments

A. Explain experiment

The team's objective in this project was to obtain a fast image processing, an appropriate number of FPS and a short memory size of the model, because of its relation with the processing. In the paper "Tradeoffs using Binary and Multiclass Neural Network Classification for Medical Multidisease Detection" the team found interesting and relevant data for the application of the project, a CNN with high performance and fast processing that is Mobile Net. In this paper, several CNN are compared such as "VGG16", "Inception", "NASnet", "MobileNet" and others. Main focus is finding out which CNN is better for medical multi-disease detection using fine tunning in order to solve the problem, they experiment and prove that MobileNet has a good number of FPS with their dataset with OvR (One vs Rest) which is used to multi-class classification. In the part of the accuracy , that was previously explained, MobileNet efficiently trade off between latency and accuracy is present. In addition, the developpers experimented with the confussion matrix and ROC, even if some regular results were obtained about this metrics, it was considered that this CNN is the best option for their requirements.

In the project considerations, the team aimed for a CNN that has fast processing in order to predict in real time. For that reason the team considered the possibility to use MobileNet, nonetheless for the project more efficiency was needed, so the best way to improve this parameter is using "SSD" like framework. In addition, it was necessary to evaluate the metrics, however, these were not only evaluated for MobileNet, but also for the SSD combination, meaning that the integration MobileNet-SSD was evaluated.

B. Metrics

1) Metrics table: The team used scikit−learn metrics to obtain the metrics explained in table I, one by one results are shown in the confusion matrix in figure 5.

	precision	recall	f1-scorem	support
0	0.86	0.62	0.72	100
1	0.70	0.90	0.79	100
accuracy			0.76	200
macro avg	0.78	0.76	0.76	200
weighted avg	0.78	0.76	0.76	200

TABLE I
TABLE GENERATED USING SCIKIT-LEARN CLASIFICATION REPOR METHOD.

2) Confusion matrix: Graphic that makes a comparisson between known values and predicted values

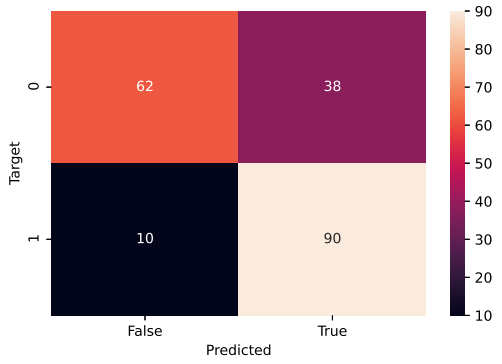

Fig. 5. Confusion matrix made with seaborn graphics

C. Comparison with previous studies

In this part, the team compared the evaluation metrics with some paper which has the same objective as this project has. The evaluation metrics that were compared are:

- **Accuracy**, quantify how frequently the calculation groups an information point effectively.
- **Precision**, is the small amount of applicable cases among the recovered occurrences.
- **Recall**, is the negligible part of pertinent examples that were recovered.

This comparison can be seen in the next figure 6.

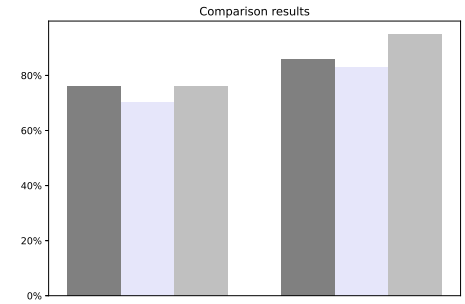

Fig. 6. Comparison the results with the model YOLOv3 with TL.

	Mobile Net-SSD	YOLOv3 with TL
Accuracy	76.0 %	95.0 %
Precision	70.3 %	86.0 %
Recall	76.0 %	83.0 %

TABLE II
TABLE GENERATED FOR COMPARISON THE EVALUATION METRICS.

VI. Conclusions

The team came to the conclusion that an acceptable model was obtained, based on all the metrics applied during the development of the experiment and comparing the accuracy of 76% with professional projects with an accuracy of more than 90%. The experiment was evaluated with a test set created by the same team using images taken from the city of La Paz, Bolivia along with images from the COCO data set. Additionally, relaying in the results of the confusion matrix, it was concluded that the algorithm has greater certainty when predicting positive cases of non-compliance of social distancing than when predicting negative cases of non-compliance.

VII. Bibliography

1) W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick.CoRR,abs/1504.04788, 2015.
2) F. Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357v2, 2016.
3) Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. (2016)
4) Hyodo, K. (2019, 12 november). MobileNet-SSD-RealSense. GitHub. https://github.com/PINTO0309/MobileNet-SSD-RealSense#summary
5) chuanqi305. (2018, 21 april). MobileNet-SSD. GitHub. https://github.com/chuanqi305/MobileNet-SSD/tree/master/voc
6) djmv. (2020, 13 november). MobilNet SSD opencv. GitHub. https://github.com/djmv/MobilNet_SSD_opencv
7) Mehla, S. (2021, 5 abril). Social Distancing Detector using OpenCV. Medium. **medium url**
8) Berstad, Tor and Riegler, Michael and Espeland, Havard and de Lange, Thomas and Smedsrud, Pia and Pogorelov, Konstantin and Stensland, Hakon and Halvorsen, Pal (2018) Tradeoffs Using Binary and Multiclass Neural Network Classification for Medical Multidisease Detection $1 - 8.10.1109/ISM.2018.00009$.
9) National Center for Biotechnology Information, U.S. National Library of Medicine. (2020, 1 noviembre). NCBI - WWW Error Blocked Diagnostic. Nationa Centerfor Biotechnology Information. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7603992/