POLITECNICO DI TORINO

Master's Degree in Computer Science Engineering - Data Science

PUC
RIO

Master's Degree Thesis

Monitoring COVID-19 prevention measures on CCTV cameras using Deep Learning

Supervisors

Prof. Hélio Côrtes Vieira LOPES[a]

Prof. Paolo GARZA[b]

Candidate

Davide Antonio Maria COTA

[a]PUC Rio de Janeiro
[b]Politecnico di Torino

October 2020

Summary

With the unfortunate rise of COVID-19 disease the whole world is in search of a way to stop the spreading of the virus. New restrictions in everyday life came up, like quarantine, social distancing, wearing masks, washing hands more frequently, limited number of people in closed places and more. This project is dedicated to monitor, through CCTV cameras videos, three of major prevention measures: social distancing, wearing masks, counting the number of people in a closed place. Besides being three different problems distant from each others, they share the possibility of being analyzed through camera images. The need of having some indicators in real time that these measures are being respected in an area of interest is more current than ever. Moreover, in these days CCTV cameras can be found anywhere, from public places such as airports, hospitals, schools, museums to shops, retail stores, houses. It makes the perfect instrument to reliably have real time images with no further installations. With the improvements in the last years in the field of GPU computing, Machine Learning algorithms became the first candidates to address this kind of problems. Different kinds of models in this work are used and presented to offer a reliable instrument to counteract the spreading of the virus.

Acknowledgements

This thesis marks the end of my studies at Politecnico di Torino, to whom I need to say thanks to let me overcome obstacles that I thought being bigger than me, like this work.
Since day one, I've been welcomed and made to feel at home from this beautiful city, that I immediately felt in love with. During this long journey I met wonderful people: Danny, Maps, Giovanni, Sara, Luca, Dario, Nicola and many others. We had unforgettable moments during this path, even if sometimes the path was lost.
I wish to say thankw to the awesome people met during my exchange in Rio de Janeiro. Even if our trip was short, I have lots of memories with each of you: Giacomo, Marta, Julia, Natalie and my three brazilian mothers Ana, Nadja, and Angela.
I would like to thank Marco, Ruben and all my Social Club basketball teammates, that taught me how to rejoice in defeats (lots).
A big thank also goes to my PolitOcean friends, we faced lots of challenges together.
Thanks to Renato and Pier, I'm looking forward for the next falfest.
Finally, I need to say thanks to my family. My uncles Licio and Francesco, my aunt Patrizia, my grandmothers Emma and Marisa. Thanks to my mom and dad that always supported me, they always believed in my successes and never in my failures. Thanks to my sister, Carlotta, and my brother, Andrea, we have so many great moments ahead.
Thanks to Chiara, we were just two lost lovers when the universe reconnected the points, you've always been at my side.
Last but not least, Davide, this is for you.

"And I knew exactly what to do. But in a much more real sense, I had no idea what to do."
- Michael Scott

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial Intelligence

AP
Average Precision

CCTV
Closed Circuit TeleVision

COVID-19
COrona Virus Infection Disease, 2019

FPS
Frames Per Second

GPU
Graphics Processing Unit

ML
Machine Learning

PUC
Pontifícia Universidade Católica (do Rio de Janeiro)

SVM
Support Vector Machine

WHO

World Health Organization

Chapter 1

Introduction

The project presented in this thesis is intended to build a robust system to monitor the COVID-19 prevention measures using Deep Learning algorithm, analysing data coming from surveillance camera. One challenging goal of this work consists in analysing low resolution CCTV camera frames and giving in output accurate predictions. Most of the data collected comes from *Pontifícia Universidade Católica do Rio de Janeiro* surveillance camera.

The following chapter is organised in three parts: the first one presents a description of the COVID-19 outbreak, the second one explains the data source and the problem specification and the the last one describes the project structure.

1.1 COVID-19

COVID-19 is the acronym for *COronaVirus Disease 19*, it is an infective disease caused from **SARS-CoV-2** virus, belonging to coronavirus family. The citizens of *Wuhan, Hubei, China*, in the late 2019, were the first to face the new virus, that resulted in an ongoing pandemic. As of 9 September 2020, more than *27.4 million cases*, with nearly *18.4 million people recovered* and more than *896,000 deaths* have been reported across the globe.

General symptoms can be *cough*, *fever*, *breathing difficulties*, *fatigue*,

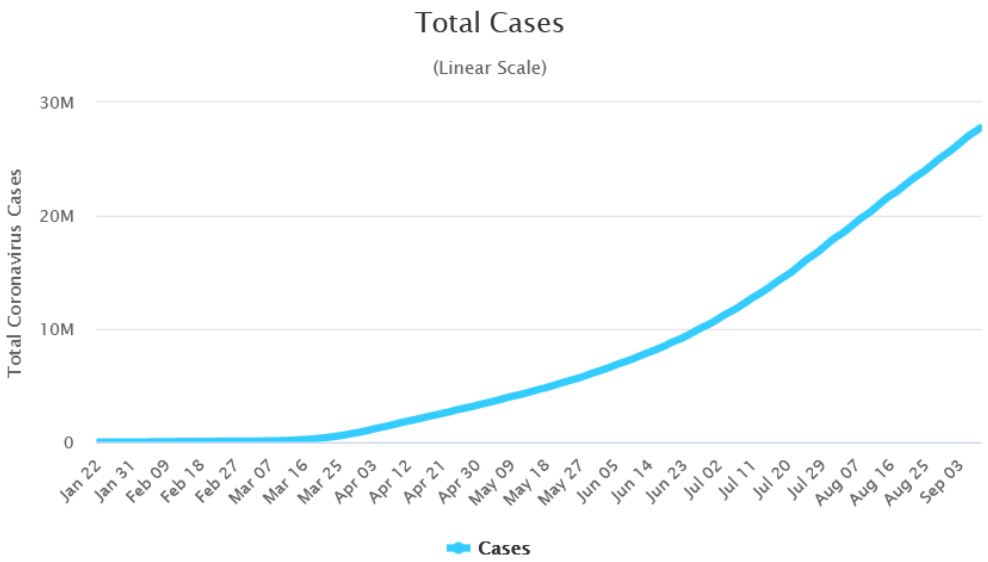


Figure 1.1: Number of total cases at 03/09/2020, source *Worldometer*.

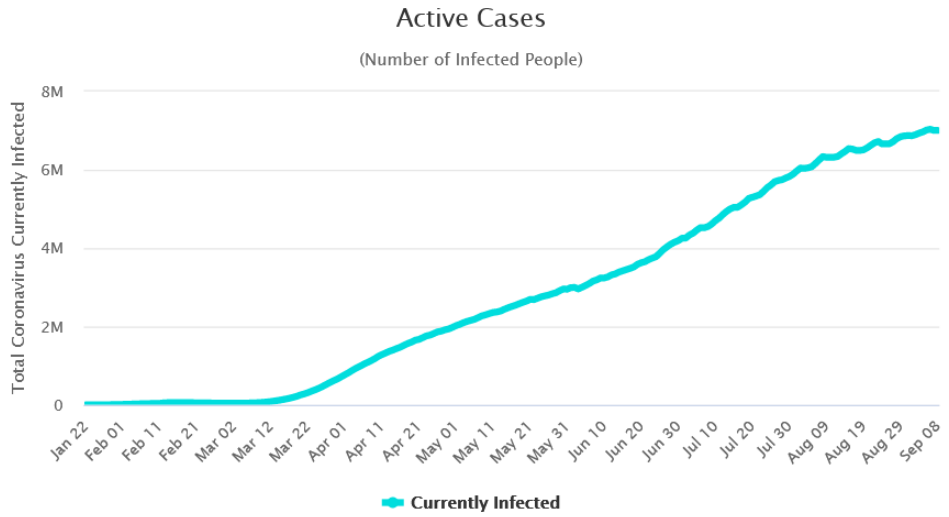


Figure 1.2: Number of active cases at 08/09/2020, source *Worldometer*.

and *loss of smell and taste*. In any severe case, some individuals develop *acute respiratory distress syndrome* (ARDS). In these particular cases,

longer-term damage to organs has been observed and some patients who have recovered continue to experience a range of effects including *muscle weakness*, *memory loss* and other symptoms for months afterwards. The virus transmits mostly via respiratory tract through small droplets from talking, coughing, sneezing with an higher transmission factor in closed and poorly ventilated places. People may also become infected by touching contaminated objects and then touching their face.

1.1.1 Prevention measures

The prevention strategies suggested by WHO consists in washing hands frequently, avoiding touching the face with unwashed hands, and coughing or sneezing into a tissue. Wearing masks is also suggested in public and crowded place such as bus, trains or hospitals. Physical distancing measures are also recommended to slow the disease transmission.

1.1.2 Social distancing

Social distancing (also known as *physical distancing*) includes infection control actions aiming to slow the spread of the virus by minimising close contact between people. Social distancing, is a set of measures adopted to prevent the spread of a contagious disease by reducing the occurrences people come into contact with each other and maintaining a physical distance between individuals. Some examples to be considered as examples of social distancing can be: *isolation*, *closing of public places* such as schools, stadiums, cinemas, *remote work* and keeping a distance from other people higher than **1 meter**. Different nations adopted an higher distance threshold. In South Korea the prescribed physical distance is 2 meters, the same recommended by the USA, Canada and United Kingdom. According tho WHO, the right distance to respect is *"at least one meter"* [1]. This indication relies on the fact that air droplets, when talking, can't fall at a distance higher than 1 meter. Instead, if coughing or sneezing, the indicated distance isn't enough anymore. In any case, the prescribed distance can vary with different scenarios, as shown by a study from *Qureshi et al.*[2]

summarized in Fig. 1.3

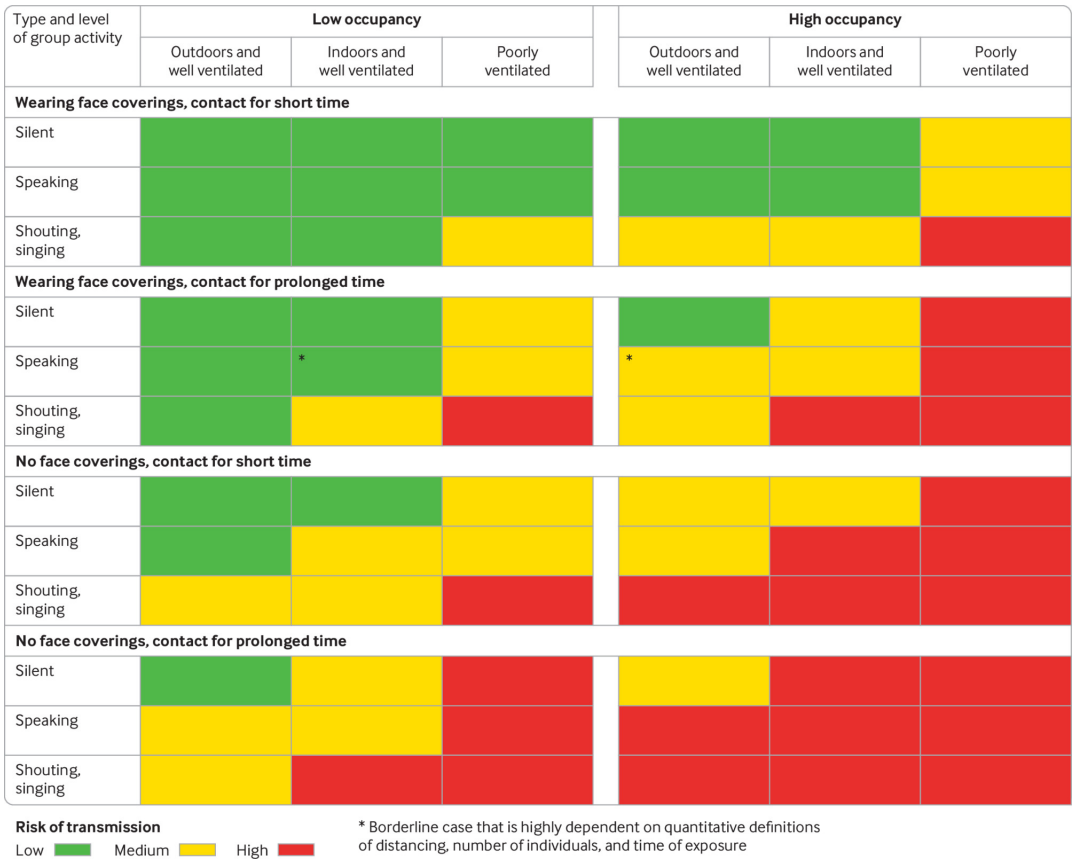


Figure 1.3: Risk of SARS-CoV-2 transmission

Fig. 1.3 shows how transmission risk may vary with different kind of situations. In high risk situations (especially in poorly ventilated areas) physical distancing beyond **2 meters** should be considered. Less stringent distancing is likely to be adequate in low risk scenarios. People with symptoms (who should in any case be self-isolating) tend to have high viral load and more frequent violent respiratory exhalations [2].

1.1.3 Face masks

The WHO encourages people to wear face coverings as it can protect people wearing it from getting infected, as well as can prevent those who have symptoms from spreading the disease. *Medical masks* are

recommended for different kind of people such as health workers, anyone with symptoms of COVID-19, at-risk people (people aged 60 or over or people with underlying health conditions) when they are in crowded areas. *Fabric masks* are not so effective as medical masks and WHO does not recommend their use to contrast the virus outbreak. However, in places when physical distancing is not possible, WHO encourage the general public to use non medical fabric masks. To effectively counteract the spreading of the virus, other types of medical devices are suggested to use. They are named *Respirators*, also known as *Filtering Facepiece Respirator - FFP* with different filtering capacities such as `FFP2`, `FFP3`, `N95`, `N99`. Those are indicated for healthcare workers that are at strict contact with COVID-19 patients.

Figure 1.4: People wearing medical masks and a doctor with a N95 mask on.

1.2 Input data and problem specification

The aim of this project can be subdivided into three macro-areas: counting the number of people, monitoring if a person is respecting the prescribed distance threshold from each individual and detecting people wearing or not face masks. Our input data are the surveillance camera footage, in particular our dataset is composed of videos coming from CCTV camera installed at PUC and online real-time web cameras

situated on the streets. The final dataset consists of a wide variety of angulation and perspectives, crowd places, occlusions and different illumination spots. During the designing process dealing with this particular kind of images had a vital importance in building a suited model for CCTV camera images, especially because of their low-resolution characteristic and because of the presence of individual far away from the camera. No other public dataset has the afore mentioned features.

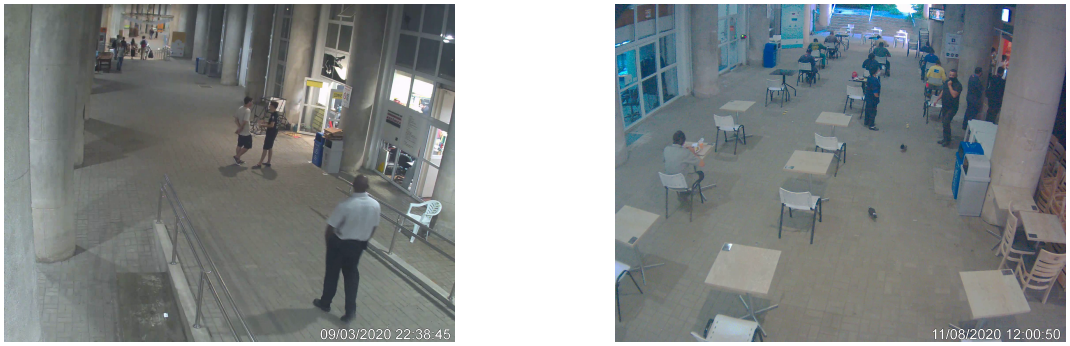


Figure 1.5: Examples of images coming from PUC CCTV cameras.

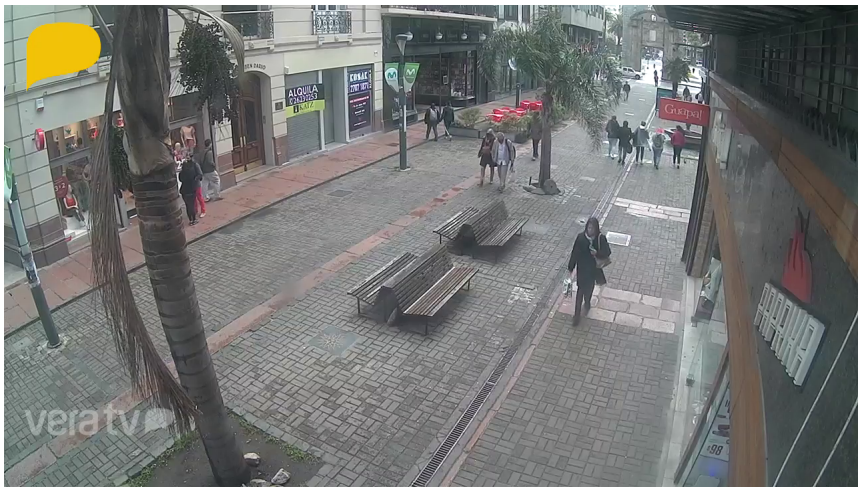


Figure 1.6: Frame coming from online webcam

The process to obtain the faces to train the face mask detector is detailed described in section 3.3. Here we will show just a few training samples. We collected 7543 training samples, 3588 with masks and

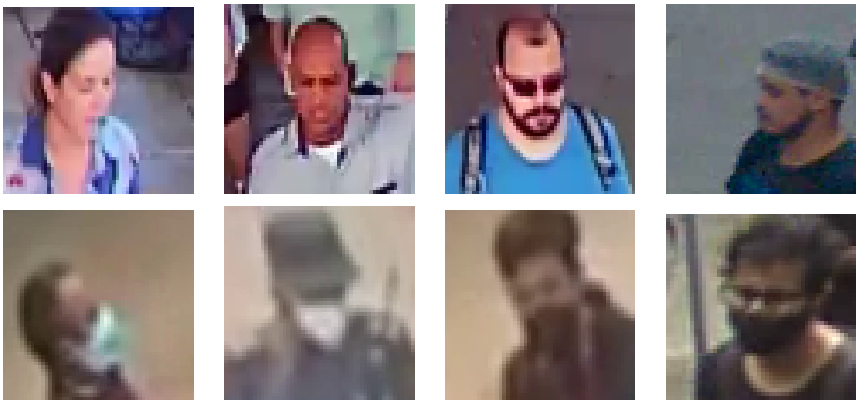


Figure 1.7: No mask and mask sample images

3955 faces without mask.

1.3 Project structure

The project structure is the following:

- **Background and Related Works:** here are presented all the works addressing problems related to proximity monitoring, mask detection and people counting. In particular for social distancing monitoring, being a task never accounted before, it is described solutions originally thought for different kind of problems that can be adopted and re-adjusted for our purposes, such as human pose estimation.

- **Proposed Methodology:** in this chapter it is detailed described the model pipeline and its core parts. It is explained the motivation behind each block and in which manner it affects the system. Some demo images of the final outputs are also showed.

- **Experimental Results:** for each type of tasks are presented the results obtained with our approach and how it performs compared to previous ones.

- **Conclusions and Future Works**

Chapter 2

Background and Related Works

In this chapter, after a first presentation about Machine Learning themes, are reported all the previous works dealing with the three main problems treated in this project: `counting the number of people`, `monitoring the social distancing` and `detecting people wearing masks`.
Please keep in mind that most of the solutions presented in the next sections were originally thought for addressing different type of problems, such as *human pose estimation*; this can be explained by the fact that it is the first time, in modern history, that the world has to face a global pandemic.

The next section will give a general introduction about what Machine Learning and Deep Learning is. After that, for each of the three task, will be reported all the related works.

2.1 Machine Learning

In the last ten years we assisted at a revolution in the theme of Machine Learning, that occurred for different reasons; first, the volume of data and computing power exploded thanks to the developing of Big

Data frameworks and the studies regarding parallel GPU computing. **Artificial Intelligence (AI)** is the set of theory and algorithms that

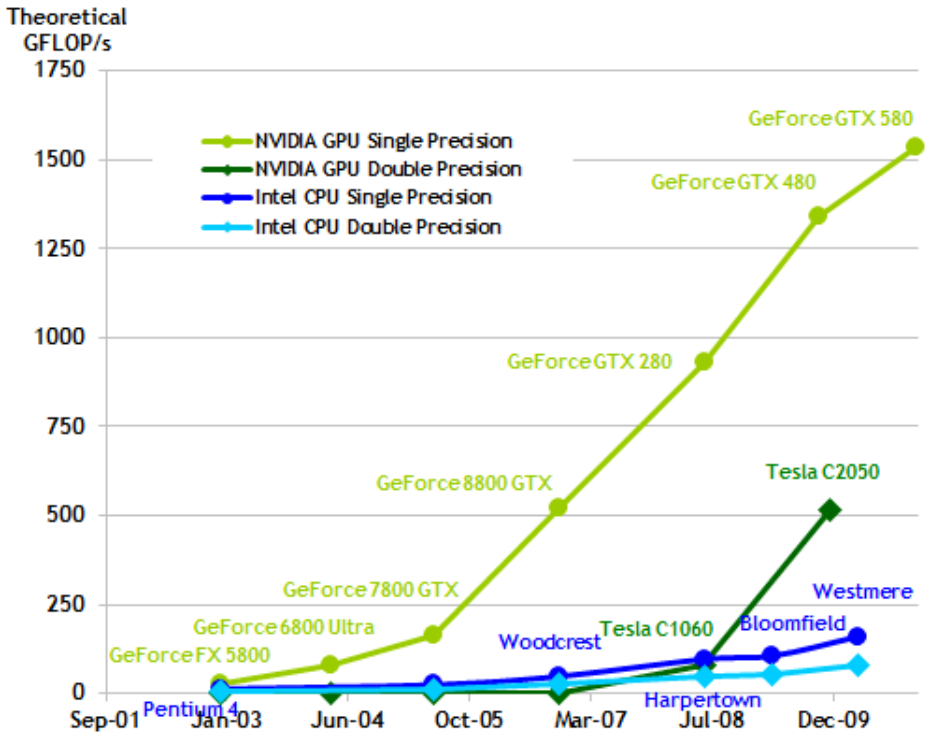


Figure 2.1: GPU vs CPU comparison [3]

enable computers to mimic human intelligence. **Machine Learning** is a subset of AI that includes statistical techniques enabling machines to improve at tasks with experience. Machine learning algorithms exploits some sort of sample data, known as *training set*, in order to make decisions or predictions without being explicitly programmed to perform the task. Machine Learning problems can be divided into two main branches: *supervised learning* and *unsupervised learning*.

2.1.1 Supervised learning

Supervised learning is a way of learning that aims to calculate a function that maps an input x, which is the *sample data*, to an output y, which is the *label*, based on example input-output pairs. It infers a

function f from labeled training data, consisting of a set of training examples. Types of problems that can be solved through supervised learning are:

- **binary classification**, such as detection problems: given x find $y \in 0{,}1$
- **multiclass classification**, such as animal classification: given x find $y \in 1, ..., k$ with $k \in \mathbf{N}$
- **regression**, that aims to estimate a continuous function: given x find $y \in \mathbf{R}$

2.1.2 Unsupervised learning

Unsupervised learning is a type of machine learning procedure that doesn't need data that has been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning searches for similarities in the data and gives a feedback respect to the the presence or absence of such similarities in each new item of data.
Problem prototypes solved with unsupervised learning are:

- **Clustering**: is the problem of grouping points by similarity, detecting similar patterns in the data distribution. Patterns on a two-dimensional dot plot are generally fairly easy to see, but we often deal with higher-dimensional data that humans cannot effectively visualize.
- **Principal Component Analysis (PCA)**: is a technique for reducing the dimensionality of the dataset. The goal is finding a subspace representing the data, in which the data can be projected.

2.1.3 Machine Learning approaches

In literature, several type of approaches has been proposed to solve different type of problems. There isn't a shotgun model, since each kind of algorithm can be best suited for particular type of data or

problems. Here we briefly list and describe the principal Machine Learning approaches and their pros and cons:

- **Nearest Neighbor**: the *Nearest Neighbor Search* is the optimization problem of finding the point in a given set that is closest, or most similar, to a given point. The algorithm computes the distance from all the training samples and take the class of the most frequent *k-Nearest Neighbors*. This approach does not need many parameters and it is easy to implement for small amounts of data. The problem is that, for a huge training set, it requires much memory and computational resources.

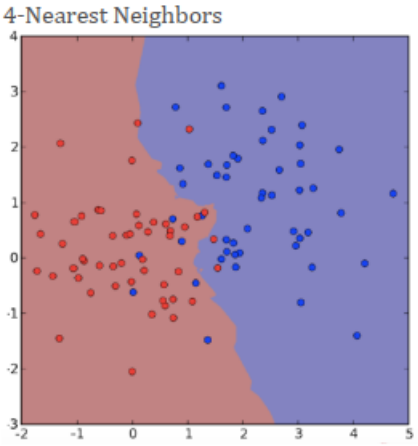


Figure 2.2: 4-Nearest Neighbors

- **Support Vector Machine (SVM)**: *Support Vector Machines* are a way to build non-linear classifiers. The purpose of this algorithm is to find the hyperplane best separating points with two classes of labels, by seeking maximum margin linear separators between the two classes. This is due to that the larger the margin, the farther any of the training points are from being misclassified. Despite being an accurate technique, non-linear SVMs does not scale over millions of data and they aren't the most interpretable model.

- **Decision Trees**: A *decision tree* is a binary branching structure used to classify an arbitrary input vector X. Each node in the

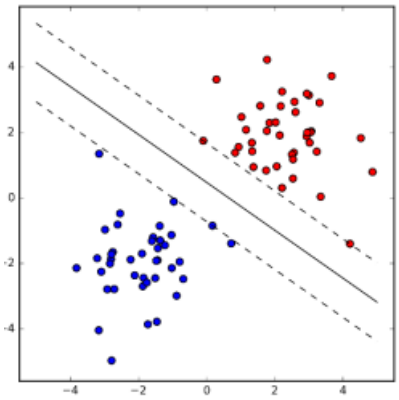


Figure 2.3: SVM classifier separates the two classes by the largest margin.

tree contains a simple feature comparison against some field. The result of each such comparison is either true or false, determining whether we should proceed along to the left or right child of the given node. This type of approach is suited when using a large number of categorical features and when intrepretabilty is asked.

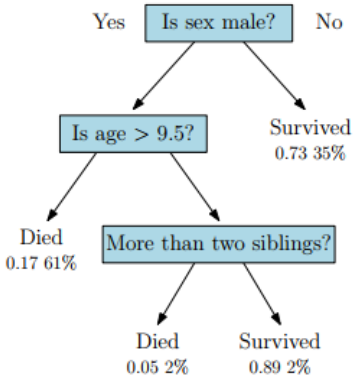


Figure 2.4: Decision tree for predicting mortality on the Titanic.

- **Naive Bayes**: a *Naive Bayes classifier* relies on the probabilistic *Bayes' Theorem* with *strong naive independence assumptions* between the features, assuming that any of the features X_i are independent given the class label Y. This model is highly scalable,

since it requires a number of parameters linear in the number of features and it is easy to update with new data (*online model*). In contrary, the accuracy decreases if there are correlations among features.

2.1.4 Deep Learning

The afore mentioned Machine Learning algorithms do not scale well to big datasets, generally because they have few parameters and have difficulties to deal with a huge number of training samples.
Deep Learning is part of a broader family of Machine Learning methods based on learning data representations. Deep Learning architectures, such as *neural networks*, have been applied to fields including computer vision, natural language processing, speech recognition and more. The *"deep"* term refers to the number of layers through which the data is transformed. In Deep Learning, each level learns to transform its input data into a more abstract representation. For example, in an image recognition task, the input is the pixel matrix: the first layer may encode borders and edges, the second layer may represent nose and eyes and the third one may recognize a whole face. There isn't

Method	**Accuracy**	**Interpretability**	**Training Speed**	**Prediction** Speed
Nearest Neighbor	5	9	10	2
Naive Bayes	4	8	9	8
Decision Trees	7	8	7	8
Support Vector Machines	8	6	6	7
Deep Learning	10	3	3	7

Table 2.1: Score from 1 to 10 (the higher the better) for the presented Machine Learning models.

only one type of *neural network*, since for different tasks exist different architectures. For image understanding and computer vision tasks, *Convolutional Neural Network (CNN)* are the most appropriate, while *Recurrent Neural Network* are the ones used for Natural Language Processing. Here we will briefly describe a standard neural network and its basic concepts. Fig. 2.5 shows how a neural network it's made. Each node of the graph represents a neuron, that calculates the value

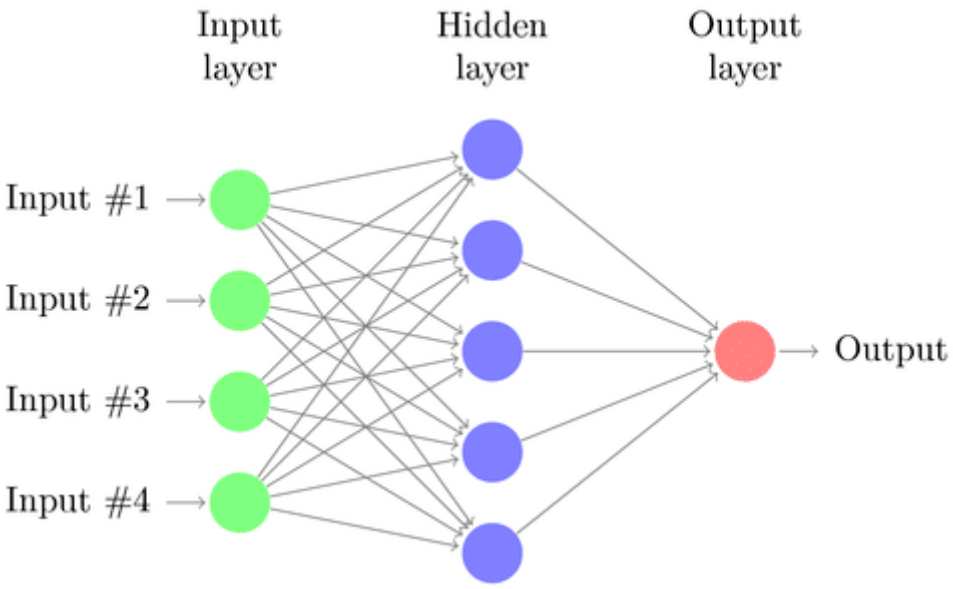


Figure 2.5: Basic neural network

of a function $f(x)$ with given inputs. Each edge (x, y) connects the output of node x to the input of a node y in a further layer in the network. Moreover, each edge has an associated multiplier coefficient $w_{x,y}$ called *weight*. The input of y is the $w_{x,y} \cdot f(x)$, meaning node y calculates a weighted sum of its inputs. The first layer is the *input layer* that is made by a number of nodes equals to the input features. The last layer is the *output layer*, where the prediction score will show up. For a binary classification problem the network will have only one output node, representing the True or False prediction; for a multiclass classification problem, instead, the number of output nodes will be the same as the number of different classes that the we aim to predict (*e.g.* a neural network that predicts the sex of a person has only one output node, while a classifier that predicts the breeds of dogs will have a N outputs nodes, with N equal to the number of dog breeds present in the training set). Between these input and output layers there are some *hidden layers* of nodes that compute the values of the lowest level in the network, propagate them forward, and repeat from the next layer until the last one. The learning process consists in setting the weights

of the coefficient parameters $w_{x,y}$. The more edges there are, the more parameters the net has to learn and adjust to get an output similar to y_i when fed input x_i. The disadvantages of deeper networks is that they become harder to train the larger and deeper they get since it increases the number of parameters to estimate. Moreover, networks get more computationally expensive to make predictions as the depth increases, since the computation takes time linear in the number of edges in the network.
To take advantage of depth we need non-linear node *activation functions* $\phi(\upsilon)$, they operates on a weighted sum of the inputs x where

$$\upsilon_i = \beta + \sum_i w_i x_i$$

β is the *bias* of the node because it defines the function in the absence of other inputs and has to be learned during training. During past few years some interesting non-linear activation functions have been used in building networks. The simplest and widely used non-linear activation function is the so called **ReLU** *Rectified Linear Unit*, which just takes the maximum between zero and the input value. Beyond its simplicity, the advantages are that the *ReLU* function remains differentiable, increasing monotonically and being unbounded on one side (this is alleviates the problem of the *vanishing gradient*).

ReLU	$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
Softmax	$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} i = 1, ..., J$
tanh	$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$

Table 2.2: Examples of activation functions

The goal during the training stage is to minimize the training error, or *loss*. The *loss* function states how costly each action taken is, and,

by designing this function, we guide the model evolving in the desired direction during training. *Backpropagation* is a method used to train neural networks by calculating a *gradient* that is needed in order to update the weights of the neural network: the error (loss) is calculated at the final nodes and distributed backwards through the network's layers.

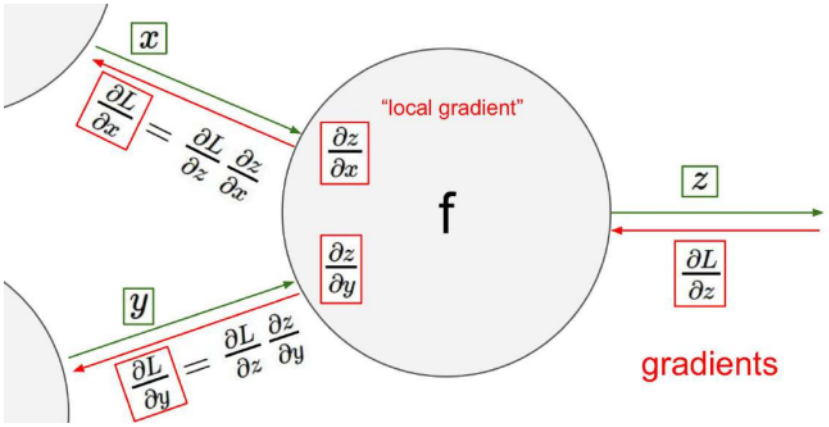


Figure 2.6: Gradients computation for each of the the three nodes

2.2 Related Works

In this section it will be illustrated the works proposed in literature and the state-of-art techniques for each of the main problems that this project is intend to solve: *counting the numbers of people*, *monitoring social distancing* and *detecting people wearing masks*.

2.2.1 People counting

The problem of People Counting, or Crowd Counting, is a task to count people in image. It is mainly used in real-life for automated public monitoring such as surveillance and traffic control. Different from object detection, Crowd Counting puts its goal at recognizing arbitrarily sized targets in various situations including cluttering and sparse scenes at the

same time. Like any other computer vision problem, people counting has to face many non-trivial challenges such as non-uniform distribution of people, non-uniform illumination, occlusions, different perspective making the problem not easy solve and pushing the researchers, in the last few years, to propose several works addressing this kind of problem. Crowd analysis can be applied in many field of critical importance, such as *safety monitoring* in places such as museums, stadiums, airports to perform related tasks like *congestion analysis* or *anomaly detection*. Another important application can be found during the *design of public spaces*, especially the ones predicted to be areas of public gathering with high risk of congestion. Last but not least, this kind of solutions can be really helpful in *forensic search* and *disaster management* in order to search for missing people or victims in events such as bombing, shooting or large accidents.
In literature exist several approaches facing people counting tasks that can be divided into three main branches: *Detection-based approaches*, *Regression-based approaches* and *Density estimation-based approaches*.

Detection-based approaches In *Detection-based approaches* the algorithms relies on a backbone network that is a *detector* which purpose is to detect people in the scene [4] and its output represents the number of people [5]. The classifier can be a monolithic one [6], *i.e.* extract features from a full body, or a parts-based classifier [7] where the aim is finding specific body parts such as head and shoulder to predict the people count in a given area. These methods works well in low density crowd scenes, since high density areas disturbs the prediction and the model cannot discriminate anymore from the different body parts of each person in the image.

Here we present a wide adopted model in our project and the state-of-the-art one-stage object detector: **YOLO (You Only Look Once)**, originally from *Redmon et al.*[8]. The basic idea behind YOLO is to divide the image into an SxS grid. Each grid cell predicts only one object and a fixed number of boundary boxes. For each grid cell:

1. it predicts B boundary boxes with a confidence score

2. it detects one object only

3. it predicts C conditional class probabilities, one per class. The highest score represents the predict class.

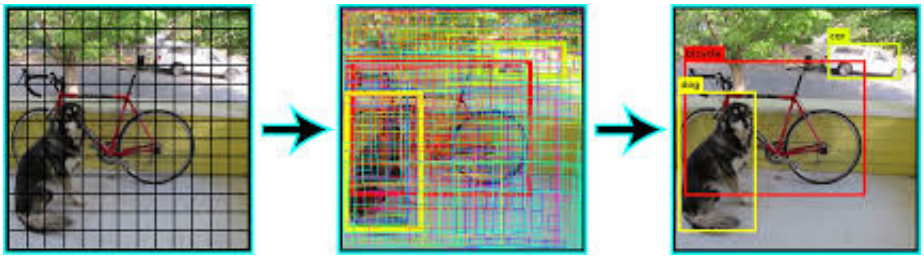


Figure 2.7: How YOLO detects and estimates the class object

Today this algorithm evolved to its 4th version with **YOLOv4** [9] that achieves state-of-the-art results at a real time speed on the *MS COCO dataset* [10]. As we can see from Fig. 2.8, *YOLOv4* improves

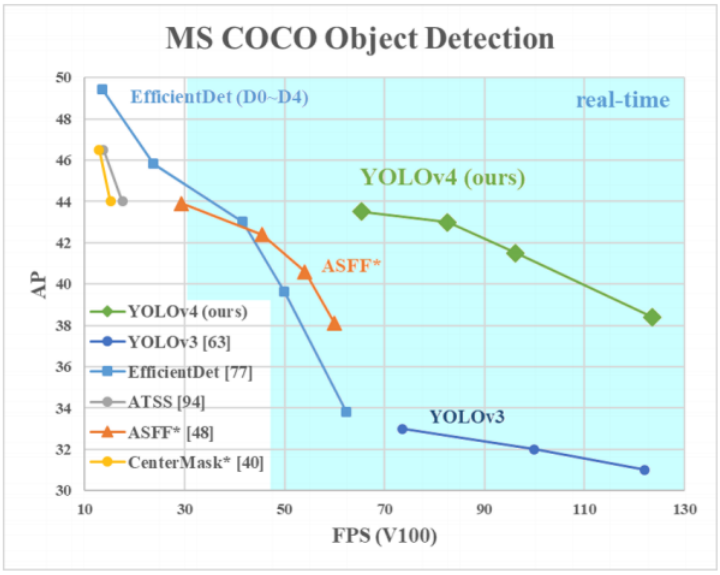


Figure 2.8: YOLOv4 performances against other popular object detectors

YOLOv3 AP and *FPS* by 10% and 12%, respectively. Even if *Efficient-Det D4-D3* [11] achieves better *AP* than *YOLOv4* model, it runs at a

lower speed, less than 30 *FPS*, crucial point if we want to adopt this kind of analysis in a real-time system.
YOLOv4 isn't a model made up from scratch, but it includes existing implementations with new features. A modern detector is generally made up of two parts, a *backbone* which is pre-trained on ImageNet [12] and a *head* which is used to predict the output labels and estimate the bounding boxes. The head part usually can be of two kinds, *i.e.* a two-stage object detector, such as R-CNN [13], or a one-stage object detector such as YOLO [8], SSD [14] or RetinaNet [15]. In the last few years modern object detectors often insert between backbone and head some layers generally used to collect feature maps from different stages, and we can call them the *neck* of the network. To go more in the specific *YOLOv4* consists of:

- **Backbone**: CSPDarknet53 [16]
- **Neck**: SPP [17], PAN [18]
- **Head**: YOLOv3 [19]

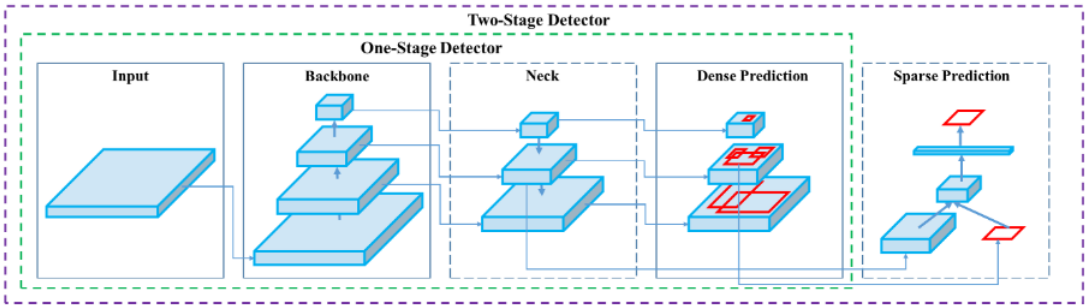


Figure 2.9: Architecture of an object detector [9]

As it can be seen from Fig. 2.10, YOLOv4 shows outstanding performances both for accuracy and speed, making it the perfect candidate for tasks dealing with real-time people counting in restricted areas.

Regression-based approaches The afore mentioned detection-based approaches suffered problems of occlusions and extremely dense crowds.

Method	Backbone	Size	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	38 (M)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	31 (M)	**43.0%**	**64.9%**	**46.5%**	**24.3%**	**46.1%**	**55.2%**
YOLOv4	CSPDarknet-53	608	23 (M)	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
Learning Rich Features at High-Speed for Single-Shot Object Detection [84]									
LRF	VGG-16	300	76.9 (M)	32.0%	51.5%	33.8%	12.6%	34.9%	47.0%
LRF	ResNet-101	300	52.6 (M)	34.3%	54.1%	36.6%	13.2%	38.2%	50.7%
LRF	VGG-16	512	38.5 (M)	36.2%	56.6%	38.7%	19.0%	39.9%	48.8%
LRF	ResNet-101	512	31.3 (M)	37.3%	58.5%	39.7%	19.7%	42.8%	50.1%
Receptive Field Block Net for Accurate and Fast Object Detection [47]									
RFBNet	VGG-16	300	66.7 (M)	30.3%	49.3%	31.8%	11.8%	31.9%	45.9%
RFBNet	VGG-16	512	33.3 (M)	33.8%	54.2%	35.9%	16.2%	37.1%	47.4%
RFBNet-E	VGG-16	512	30.3 (M)	34.4%	55.7%	36.4%	17.6%	37.0%	47.6%
YOLOv3: An incremental improvement [63]									
YOLOv3	Darknet-53	320	45 (M)	28.2%	51.5%	29.7%	11.9%	30.6%	43.4%
YOLOv3	Darknet-53	416	35 (M)	31.0%	55.3%	32.3%	15.2%	33.2%	42.8%
YOLOv3	Darknet-53	608	20 (M)	33.0%	57.9%	34.4%	18.3%	35.4%	41.9%
YOLOv3-SPP	Darknet-53	608	20 (M)	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
SSD: Single shot multibox detector [50]									
SSD	VGG-16	300	43 (M)	25.1%	43.1%	25.8%	6.6%	25.9%	41.4%
SSD	VGG-16	512	22 (M)	28.8%	48.5%	30.3%	10.9%	31.8%	43.5%

Figure 2.10: Comparison of the speed and accuracy of different object detectors on the MS COCO dataset (test-dev 2017). (Real-time detectors with FPS 30 or higher are highlighted) [9]

To overcome these issues, a new strategy was born that consisted in counting by regression with the aim of learning a mapping between features extracted from the input image [20]. This method has the big advantage of being independent from a detector, that adds complexity to the model. Instead, this model is made up of two major components: low-level feature extraction and regression modelling.
Foreground and local features are extracted from the input image in a video using standard background subtraction techniques (*e.g.* histogram oriented gradients (HOG)). Once these features are extracted, several regression techniques such as linear regression [21, 22, 23] or ridge regression [24] are used to learn the mapping low-level feature to the people count.
In particular, in the work proposed by *Massa et al.* [23], the goal is estimating, from video surveillance camera frames, the *real-time* people count in small indoor environments, and particularly in retail

stores, where there is a mix of moving and stationary people and several occlusions from furniture and walls. They employ a supervised learning approach based on a Long-term Recurrent Convolutional Network (LRCN) regression model combined with a foreground detection method, by taking advantage of the spatio-temporal coherence between the scenes. They compared their approach with a YOLOv3 model pre-trained with COCO dataset, with the standard threshold of 25% for detection confidence. The number of people for each image is the number of objects classified as a person from YOLO. We can see from table 2.3 that LRCN-RetailNet accomplishes its task with a very low Mean Absolute Error (MAE)[1]. They visually examinated some predicted images and found cases where YOLO missed many visible people and cases where it predicted non-existing ones, Fig. 2.11. This can be due to the fact that, first, YOLO is not trained to cope with non-standard poses and occluded people, which is quite common in a shop scenario. Second, YOLO utilizes only one image to make its prediction, and doesn't have the spatio-temporal knowledge that LRCN-RetailNet got. On the other hand, YOLOv3 makes a huge gap in terms of prediction

network	MAE	prediction time
LRCN-RetailNet	**0.384**	159.73ms
YOLOv3	6.241	**52.1ms**

Table 2.3: LRCN-RetailNet vs YOLOv3

time. If the final model has to be used in real-time applications, this feature that cannot be excluded and it has to be considered to sacrifice accuracy for a faster response time.

Density estimation-based approaches If the regression methods resulted good in addressing the problems of occlusion and clutter, most of them ignored the spatial information. A first work proposed by

[1]MAE $= \frac{1}{n}\sum_i |y_i - \hat{y_i}|$ with y_i representing the groundtruth and $\hat{y_i}$ the predicted output

(a) Example with many false-negatives. (b) Example with many false-positives.

Figure 2.11: Inaccurate predictions from YOLO [23]

Lempitsky et al. [25], suggested to learn a linear mapping between local patch features and corresponding object density maps, by including spatial information in the learning process. In this way, it isn't necessary anymore detect and localize individual object or parts of them, since it requires only to estimate image density whose integral over any region in the density map gives the people counting. The entire problem is posed as a convex optimization task. *Wang and Zou* [26], identified that despite this method being effective, it was slow from computational complexity point of view. They proposed a fast method for density estimation based on subspace learning. *Xu and Qiu* in 2016 [27], noticed that the past crowd density estimation methods used a smaller set of features that limited their accuracy. They proposed to enhance the performances by using a richer set of features, retrieving them through random forest as the regression model, being faster and more scalable respect to previous methods (based on ridge regression or Gaussian process regression).

2.2.2 Monitoring social distancing

Monitoring social distancing is a problem as timely as ever, there aren't many the publications, in the Computer Vision field, regarding the issue of monitoring the distance between people on video frames. The majority researches for this type of issue fall into past works of *object detection* (in particular *pedestrian detection*) and *human pose*

estimation.

Direct approaches Specific social distancing monitoring works are the one proposed that make use of different sensor, such as LiDAR [28] or stereo camera systems [29]. Another solution making use of sensors, that doesn't utilize Machine Learning algorithms, is the one proposed by *Bian et al.* [30]. They built a wearable, oscillating magnetic field-based proximity sensing system to monitor social distancing. The wearable system owns a detection range of beyond 2 meters and it could efficiently track the individual's social distancing in real-time. Another approach is the one from *Sathyamoorthy et al.* [31], whom exploit a mobile robot with commodity sensors, an RGB camera and a 2-D lidar to perform collision-free navigation and estimate the distance between all detected people. For tracking and detecting pedestrians they use a YOLOv3 implementation. They presented a novel real-time method to estimate distances between people by using a homography transformation, transforming the camera's angled view of the ground plane by applying a homography transformation to four points manually selected on the ground plane in the angled view. Other works [32, 33] used the homography transformation to obtain an estimate of the 3D coordinates of the people in the space. In particular *Yang et al.* [33] confronted the performances of the two main object detectors, YOLOv4 and Faster R-CNN [34], to detect the individuals inside the scene. Once predicted the bounding boxes, they adopted a well-known inverse homography transformation called *Bird's-Eye-View (BEV)*:

$$p^{bev} = M^{-1}p^{im}$$

where $M \in \mathbf{R}^{3x3}$ is a transformation matrix describing the rotation and traslation from world coordinates to image coordinates, $p^{im} = [p'_x, p'_y, 1]$ is the homogeneous representation of $p' = [p'_x, p'_y]$ in image coordinates, and $p^{bev} = [p_x^{bev}, p_y^{bev}, 1]$ is the homogeneous representation of the mapped pose vector. The world pose vector p is derived from p^{bev} with $p = [p_x^{bev}, p_y^{bev}]$. Computing the distance between each pair of people is trivial, since the distance $d_{i,j}$ for person i and person j is

calculated by taking the Euclidean distance between their pose vectors:

$$d_{i,j} = ||p_i - p_j||$$

The disadvantage of adopting this particular transformation is that it requires the user calibration to set the area of interest or ROI (Region Of Interest). The polygon shaped ROI is then warped into a rectangle which becomes the so called Bird's-Eye-View. The BEV has the property of points being equidistant independently of their position. Homography transformation appears to be a wide used technique to

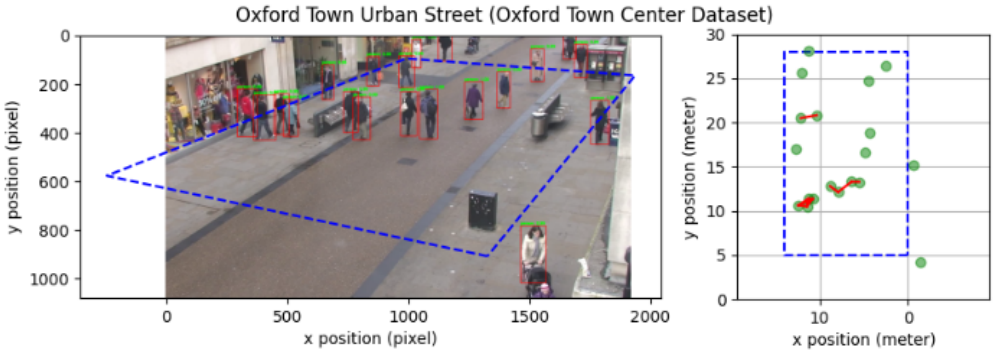


Figure 2.12: On the left, video frames output with ROI. On the right, how the BEV looks like. [33]

estimate the individual coordinates, *Khandelwal et al.* [35] uses the same approach on CCTV cameras monitoring the workplaces to send real-time voice alerts to the workers not respecting the distancing. Using an object detection algorithm as backbone revealed to be a great choice addressing this kind of problem, infact *Punn et al.* [36] inserted in their pipeline a fine-tuned YOLOv3 detector, that showed up to be the best trade off in terms of speed and accuracy compared against other classifier, alongside with a tracking algorithm. To summarize their workflow, it consisted in:

1. Fine-tune YOLOv3 on the Google Open Image Dataset (OID), consisting in images containing or not people.

2. Perform the inference on the frame to retrieve the bounding boxes

of each detected person and a unique person ID from the tracking algorithm.

3. Calculate the 3D coordinates (x, y, d) for each individual, where (x, y) are the centroid coordinates of the bounding box and d defines the depth of the person from the camera computed as follows:

$$d = (2 \cdot 3.14 \cdot 180)/(w + h \cdot 360) \cdot 1000 + 3$$

where w represents the width of the box and h its height.

4. Compute the Euclidean distance for each pair of 3D coordinates.

This approach has on its pros the fact of being calibration independent, since no BEV transformation is computed. On the other hand the distance d computed represent a general estimate of the real depth, surely further away from the groundtruth than the ones obtained using an homography transformation.

Human pose estimation Human pose estimation (HPE) aims to obtain posture of the human body from input images or video sequences. In recent years, with the rise of new GPU parallelization techniques, HPE also achieved remarkable improvements by employing deep learning technology. HPE methods can have different structures and characteristics:

- **Human body model-based (generative) vs human body model-free (discriminative)**: the difference is whether a method uses human body models or not. Discriminative methods learn a mapping between input sources and human pose space, without the needs of a human body model, contrary to the generative methods. Human body model-free methods are usually faster but less accurate then generative ones.

- **Top-down vs bottom-up**: top-down approach means that the human pose estimator requires as input the person location in bounding boxes. This usually translates in having a first stage detector separated from the model estimating the pose. In contrast,

bottom-up methods first predict all body parts of every individual and then group them either by human body model fitting.

- **Regression-based vs detection-based**: the regression-based approach tries to directly map the input image to the coordinates of body joints. The detection-based methods aims to detect person based on two main representations: joint location heatmaps or image patches. Direct mapping from images to joint coordinates isn't an easy task due to the fact that it's a highly nonlinear problem.

- **One-stage vs multi-stage**: one-stage methods uses a monolithic network or end-to-end network with the goal of directly map the input image to human poses, while multi-stage methods takes intermediate steps to process and predict human pose.

- **3D vs 2D**: also the goal can be different. The majority of the models can predict only the 2D coordinates of the human joints, while others can obtain also the depth in the space of each body part, that is also the most challenging task when using monocular images.

Figure 2.13: Example of 2D pose estimation

2D multi-person pose estimation In multi-person pose estimation tasks the goal isn't exclusively estimating the human pose, but it is required to detect and localize the individuals inside the scene. For this kind of problem the models are divided mainly according if they use a top-down or bottom-up approach.

The top-down models are made up of a robust person detector and a single person pose estimator. The most adopted human detector in literature for this kind of problem are Faster R-CNN, YOLO, Detectron [37], Mask R-CNN [38]. *Fang et al.* [39] used spatial transformer network [40], Hourglass network and Non Maximum Suppression (NMS) to retrieve pose estimation in presence of innacurate bounding boxes. *Xiao et al.* [41] implemented different deconvolutional layers before the last convolutional layer of ResNet [42] to generate heatmaps. *Chen et al.* [43] first introduced a cascade pyramid network (CPN) by using multi scale feature maps from different layers. A recent work from *Sun et al.* [44] had great success in HPE field. They proposed a new network as backbone, called *HRNet*, that learns high-resolution representations through the whole training process by connecting high-to-low resolution convolutions in parallel. Top-down models are the easier to implement and customize since it requires only to connect a human detector to a human pose single estimator. The majority of top-down approaches achieved state-of-the-art performance in almost all benchmark datasets.

For what concerns the bottom-up HPE methods the main components usually are a *body joint detection* and *joint candidate grouping*. *DeepCut* from *Pishchulin et al.* [45] used a Fast R-CNN network as body part detector to detect all the joint candidates and then assembled these parts with integer linear programming (ILP) to the complete skeleton. A more advanced implementation of DeepCut is *DeeperCut* [46], that used a stronger and more robust part detector based on ResNet. A wide used human pose detector in the computer vision field is *OpenPose* [47], that is the first both 2D and 3D open-source real-time system for multi-person detection. The proposed implementation uses a non-parametric representation called Part Affinity Fields to learn to

associate joints with person in the image. Currently, the bottom-up methods reach important speeds and some can run in real-time (like OpenPose), while top-down speed performance is limited by the number of detected people.

3D multi-person pose estimation 3D human pose estimation means estimate the coordinates in the space for each person body part from images or video. Despite commercial products have been used for 3D pose estimation, such as Kinect with depth sensor, they work in constrained areas and only with the equipment of extra hardware or sensors. It is important to find a way to predict human pose from monocular images, like the ones coming from CCTV cameras. Deep neural network nowadays has the possibility to predict joint depths from single images, even though this research field is pretty recent and only few methods has been proposed. *LCR-Net* [48] (Localization-Classification-Regression Network) divided the process in three stages: the first one is the detection stage where Faster R-CNN is employed to retrieve the bounding box locations. Second, each pose proposal is assigned with the closest anchor-pose scored by a classifier. Third, the pose are adjusted with a regressor. *Mehta et al.* [49] proposed a novel single-shot method for multi-person 3D pose estimation. Their approach uses novel occlusion-robust pose-maps (ORPM) in order to have the whole body prediction even under occlusions by other people or objects in the scene. Their work cames with MuCo-3DHP, the the first open-source large scale training data set showing real images of complex multi-person interactions and occlusions. Recently published, *PoseNet* from *Moon et al.* [50] represents the state-of-the-art 3D multi-person pose estimation on the MuCo-3DHP dataset. Their work consists in a top-down approach in three stages: first, a detector estimates the bounding boxes. Then, a deep neural network, called *RootNet*, gives in output the 3D center coordinates for each person detected. Finally, *PoseNet* estimates the root-relative 3D pose.

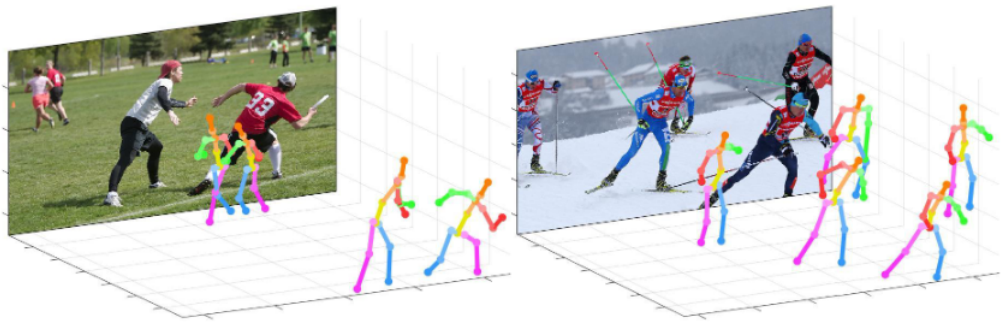


Figure 2.14: Example of 3D human pose estimation obtained with PoseNet [50].

2.2.3 Detecting people wearing masks

The problem of detecting face masks, contrary for the social distancing task, is a problem well studied in the past with different aims. Face mask detection systems came alongside automatic face recognition technologies. The first publication [51] (2005) dealing with this problem had in its intention the goal of preventing bank robberies at ATMs, since criminals that wanted to withdraw illegal money from ATM usually hide their face with masks or helmets. They proposed a model for mask detection based upon automatic face recognition methods (such as Gabor filters to generate facial features and geometric analysis for mask detection, due to the lack of valid machine learning algorithms). In 2015 *Nieto et al.* [52] proposed a system to be used in operating rooms to detect the presence of medical masks on the operators. The whole model was composed by two detectors: one for the face itself, and the other one for the medical mask. The system was reliable with detections performed up to 5 meters from the camera. In 2019, *Sabbir et al.* [53] used the Principal Component Analysis (PCA) on the masked and unmasked images to recognize a person. They found out that, when using PCA, the recognition rate is higher for unmasked face compared to masked ones because of missing features.
With the COVID-19 outbreak, we assisted at a rise of researches in the theme of mask detection on video surveillance cameras. *Loey et al.* [54] proposed a model made up of a first component (ResNet50)

to perform the features extraction from the input image and a second component to perform the classification composed by one of three machine learning models such as decision tree, SVM and ensemble. They tested their model on three different datasets: the *Real-World Masked Face Dataset* (RMFD) that consists of 5000 masked faces and 90000 unmasked faces, the *Simulated Masked Face Dataset* (SMFD) that consists of 785 unmasked faces and 785 simulated masked faces and the *Labeled Faces in the Wild* (LFW) made up of 13000 simulated masked faces of celebrities. The SVM classifier outperformed the other two

Figure 2.15: Examples images from SMFD dataset.

approaches by scoring a testing accuracy of 99.64%, 99.49% and 100% on RMFD, SMFD and LFW respectively. *Mingjie et al.* [55] proposed a one-stage detector, called *RetinaFaceMask*, which consists of a feature pyramid network to fuse high-level semantic information with multiple feature maps, and a novel context attention module to focus on detecting face masks. Parts of the network are pretrained on a larger dataset, WiderFace, consisting of 32203 images and 393703 annotated faces or on Imagenet. The whole network is then trained on FaceMaskDataset (with 7959 images) and tested on a combination of FaceMaskDataset + WiderFace + MAskedFAces dataset (MAFA), for a total of nearly 70000 images. They also used ResNet and MobileNet as different backbones for comparison, the last one is used to have faster predictions sacrificing

accuracy. MobileNet has been used as light backbone also in a work

Backbone	Transfer Learning	Attention	Face		Mask	
			Precision	Recall	Precision	Recall
MobileNet	Imagenet	✓	80.5%	93.0%	82.8%	89.0%
		✗	79.0%	92.8%	78.9%	89.1%
	Wider Face	✓	83.0%	95.6%	82.3%	89.1%
		✗	82.5%	95.4%	82.4%	89.3%
ResNet	Imagenet	✓	91.0%	95.8%	93.2%	94.4%
		✗	91.5%	95.6%	93.3%	94.4%
	Wider Face	✓	91.9%	96.3%	93.4%	94.5%
		✗	91.9%	95.7%	92.9%	94.8%

Figure 2.16: RetinaFaceMask results

regarding the safety distancing in workplaces from *Khandelwal et al.* [35]. They trained a MobileNetV2 architecture on a custom dataset made of 4225 annotated images with 1900 labeled as 'mask' and 2300 as 'no mask' obtaining 97% score for both recall and precision. An interesting work from *Bosheng and Dongxiao* [56] implemented a system capable of classifying three types of classes: no mask wore, mask wore correctly and mask wore incorrectly. The data was collected from the open-source dataset called *"Medical Masks Dataset"*, made up of 3835 images with 671 images of no mask, 134 images of wrong mask-wearing, and 3030 images of people wearing masks. They proposed a novel facemask identification and classification algorithm, with a classification network called *SRCNet*, divided in four steps: image pre-processing, face detection and crop, image super-resolution, and facemask-wearing conditions identification. Finally, SRCNet obtained 98.7% accuracy and outperformed traditional classification methods by over 1.5%.

Figure 2.17: SRCNet outputs, each class is represented. CFW = correct facemask-wearing (green), IFW = incorrect facemask-wearing (yellow), NFW = no facemask-wearing (red) [56].

Chapter 3

Proposed Methodology

Our approach isn't made of a single network, but it is a top-down approach that can be subdivided in four main blocks:

1. **Detector**: the whole system relies on the YOLOv4 [9] algorithm for the human detection and localization. Thanks to its performances in terms of speed and accuracy represents the best candidate for our purposes.

2. **Tracker**: our system needs to store some statistics about each person during the whole time it appears in the video sequence. To do so we choose the state-of-the-art tracking algorithm, *DeepSort* [57].

3. **Human root estimator**: monitoring social distancing is possible thanks to RootNet [50] that takes in input the YOLOv4 bounding boxes and gives in output the human root 3D coordinates. RootNet is a neural network coming from PoseNet project [50] , whose we cut out the last block since we are not interested in the space coordinates of each human joint.

4. **Face mask detector and classifier**: we implements a novel approach to detect human faces bounding boxes by using a 2D human pose estimator, HRNet [44]. The image inside the box is predicted by a binary classifier having as backbone ResNet.

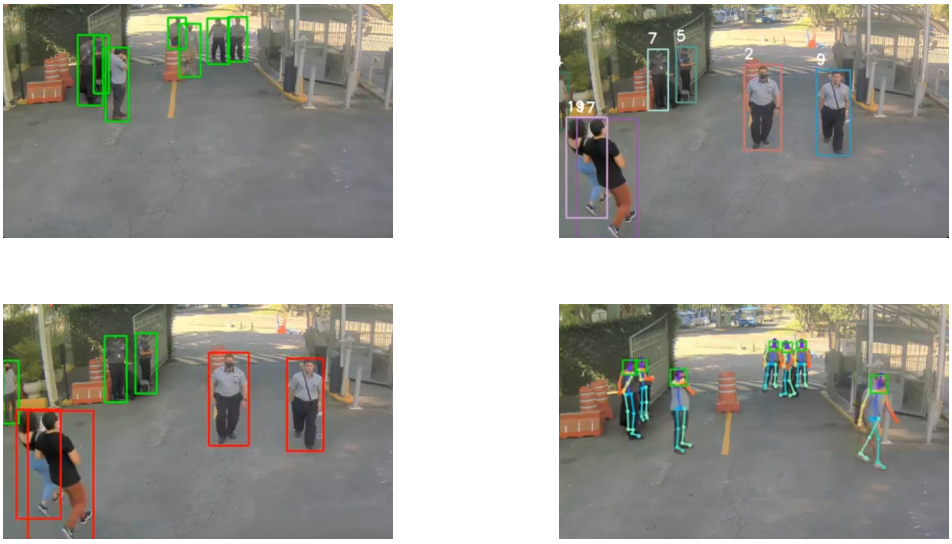


Figure 3.1: Four different stages of our approach

3.1 People counting

For our approach we choose YOLOv4 as human detection algorithm. The model has been trained on the MS COCO dataset, which is a benchmark dataset for object detection/image segmentation. The data we will use contains 117000 images containing objects belonging to 80 classes. The number of people in the scene is simply obtained by counting the number of bounding boxes detected in the image.

Network size	FPS	AP
320	100	40.8%
416	82	41.2%
512	69	43.0%
608	53	43.5%

Table 3.1: YOLOv4 performance varying the network size

3.2 Monitoring social distancing

The intermediate step between RootNet and the human detector, is the tracking algorithm. The basic idea is storing the last 3 positions for each detected person to compute the mean, in order to mitigate the error done by RootNet during a single shot human root estimation. To do this, we need a robust and reliable tracker such as DeepSort. DeepSort adopt a conventional single hypothesis tracking methodology with the so called Kalman filter.

Tracking problem The tracking algorithm takes into account 8 variables of interest: $(u, v, \gamma, h, \dot{x}, \dot{x}, \dot{\gamma}, \dot{h})$. In order, respectively, (u, v) represents the bounding box center coordinates, γ is the aspect ratio, h is the object height, and $\dot{x}, \dot{x}, \dot{\gamma}, \dot{h}$ represent their respective velocities. The Kalman filter assumes that position and velocities are random and Gaussian distributed. A linear Kalman filter with constant velocity is adopted and the observations of the object state are represented by the variables (u, v, γ, h). The algorithm looks at the number of frames of each track k since the last successful measurement association a_k, that represents also the moment when this counter is set to 0. Tracks older than a threshold age A_{max} will be marked as disappeared from the image and are erased from the track list.
The Hungarian algorithm help the algorithm to associate the predicted Kalman states and newly arrived tracks, by solving a so called assignment problem. Finally, to make the tracking algorithm more robust to switching and occlusions, it is applied a convolutional neural network that has been trained to discriminate pedestrians on a large scale person re-identification dataset.

RootNet The RootNet part estimates the camera-centered coordinates of the human root $R = (x_R, y_R, Z_R)$, *i.e.* its center of gravity, from a cropped human image. RootNet first estimates the 2D image coordinates (x_R, y_R) and then predicts the depth value (*i.e.* the distance from the camera Z_R) of the human root. Localizing the position (x_R, y_R) it's the easiest task for RootNet, but when it takes to estimate

Figure 3.2: DeepSort tracks the people during the video assigning them a unique ID

the depth value Z_R the model is required to do some assumptions in order to have a reliable prediction. In particular, we introduce a new distance measure, k, which is defined as follows:

$$k = \sqrt{\alpha_x \alpha_y \frac{A_{real}}{A_{img}}}$$

, where $\alpha_x, \alpha_y, A_{real}, A_{img}$ are focal lengths divided by the per-pixel distance factors (pixel) of x and y-axes, area of the human in real space (mm^2), and image space ($pixel^2$), respectively. The actual distance d (mm) between the camera and object can be computed in the following way:

$$d = \alpha_x \frac{l_{x,real}}{l_{x,img}} = \alpha_y \frac{l_{y,real}}{l_{y,img}}$$

, where $l_{x,real}, l_{x,img}, l_{y,real}, l_{y,img}$ are the lengths of an object in real space (mm) and in image space (pixel), on the x and y-axes, respectively.
For what concerns the network architecture of RootNet, it consists of a backbone (ResNet) to retrieve the global feature of the input image and some deconvolutional layers to produce a 2D heatmap of the root.

The final component is the depth estimation part, which outputs a single scalar value σ. The final absolute depth value Z_R is obtained by multiplying k with $\frac{1}{\sqrt{\sigma}}$. It is then straightforward calculating the

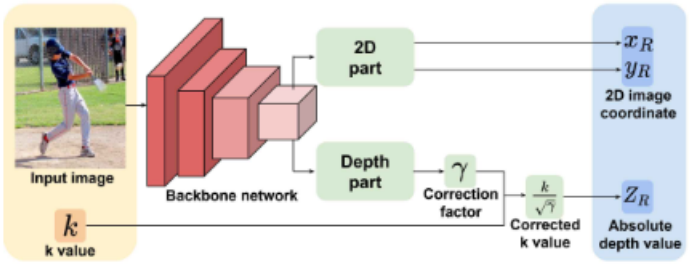


Figure 3.3: RootNet architecture

distance between each pair of person by computing the Euclidean distance between each pair of 3D root coordinates p_i and q_j:

$$d_{i,j} = ||p_i - p_j||$$

A big advantage of RootNet is that it estimates the 3D coordinates without needing any user calibration, contrary to homography transformation approaches seen before.
At this point we propose a slightly different methodology that consists in joining the tracking algorithm with RootNet, in the following way:

1. for each person p_i detected by the tracking algorithm we calculate through RootNet his 3D coordinates $C_i(t)$ at time t.

2. for each person p_i, we store the coordinates into a circular buffer B_i of size N, representing the person position during the last N frames.

3. at time T, when the buffer B_i is full, *i.e.* contains exactly N coordinates, we compute the arithmetic mean of the $C_i(T-N), ..., C_i(T)$ coordinates. This value represents the estimated position at time T.

This kind of methodology allows us to alleviate the error made by the net when it predicts the estimated position using only a single frame.

We obtained good results with circular buffers of length 3, *i.e.* taking the mean of the last 3 frames. The risk of setting larger buffer size is losing the actual person position by considering frames too distant in time from the moment we need to predict his 3D coordinates.

3.3 Detecting people wearing masks

We propose a novel approach, similar to the one proposed here [58], for what regards the face detection of each individual inside the scene: the 2D human pose estimator HRNet, trained on COCO Keypoints dataset estimate the posture for each person. We detect a valid face in

Figure 3.4: COCO Keypoints joints set

the following manner:

1. we calculate the euclidean distance from right hip to right shoulder (points 11 - 5 in Fig. 3.4. Please note that there isn't any particular reason by choosing the right part).

2. the 80% of the previous calculated distance represents the length of each side constructing the box that includes the face

3. we set the center of the box in the same coordinates of the nose (point 0 in Fig. 3.4)

4. we consider as valid faces only the ones in which HRNet predicts both eyes (points 14 - 15) with a confidence higher than 80% and the one which dimension is higher or equal than 20x20 pixels (in this way we don't try to predict faces that are too far from the camera).

A visual demo of the afore mentioned process can be seen in Fig. 3.5. In this way is also easy collect a dataset in order to train the classifier.

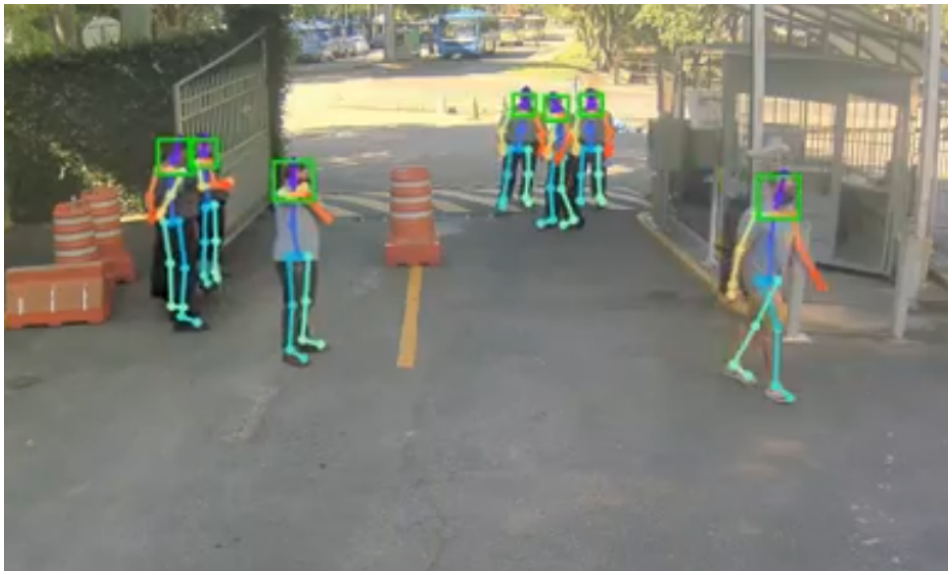


Figure 3.5: Face detection using HPE

In particular we saved all the valid faces by skipping 10 frames at a time, in order to have a wider variety of expressions and positions.
Now we can train our network on the custom dataset; the classifier chosen is the one that best performed on the CIFAR-10 [59] competition, BiT-L from Google [60]. CIFAR-10 and CIFAR-100 are dataset containing small images with a very low resolution, just like the ones coming from the CCTV camera frames. BiT-L stands for Big Transfer - Large, that makes part of a wider project from Google called *Big Transfer (BiT): General Visual Representation Learning*. They proposed a

novel way to do transfer learning, starting from different architecture of ResNet. The architecture that we utilize is ResNet-101x1 architecture, originally trained to perform multi-label classification on ImageNet-21k [12], a dataset with 14 million images. ResNet-101x1 consists in a good tradeoff between speed and reliable predictions.
The hyperparameters are chosen following the BiT-HyperRule, a heuristic rule from [60]. We used the 20% of the dataset as validation set and we used data augmentation pretty hard, using random jittering, rotation, translation and crop, in order to help the model generalize better. Once the classifier does its prediction, we update each person

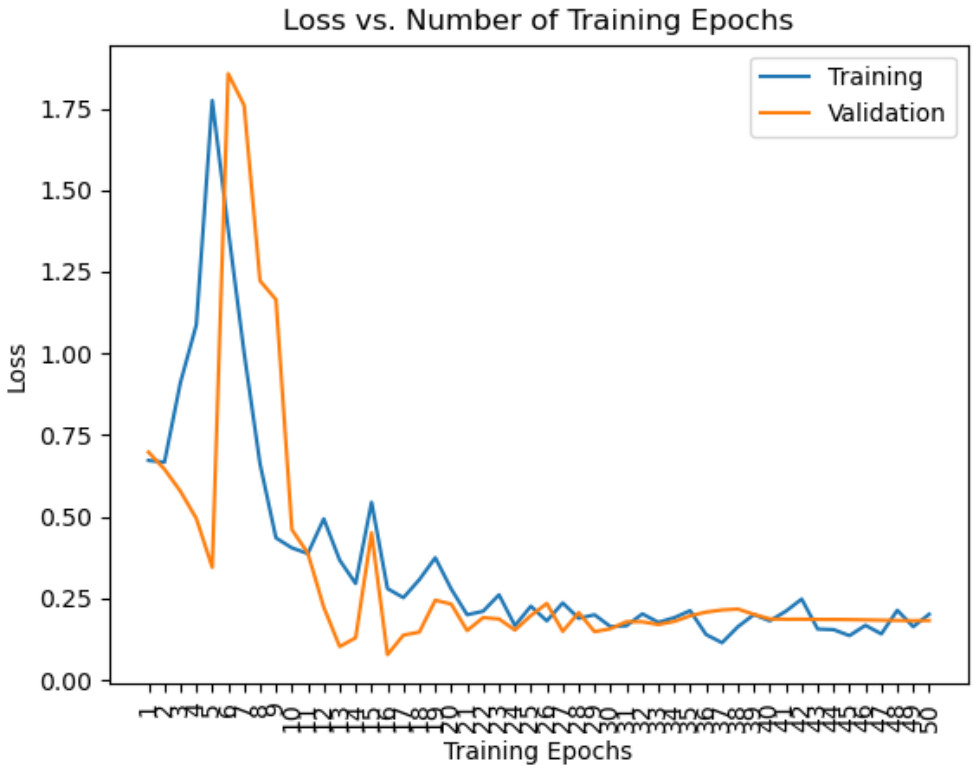


Figure 3.6: Training and validation loss during epochs

information in the following way:

1. for each person-ID we create a circular buffer of dimension N

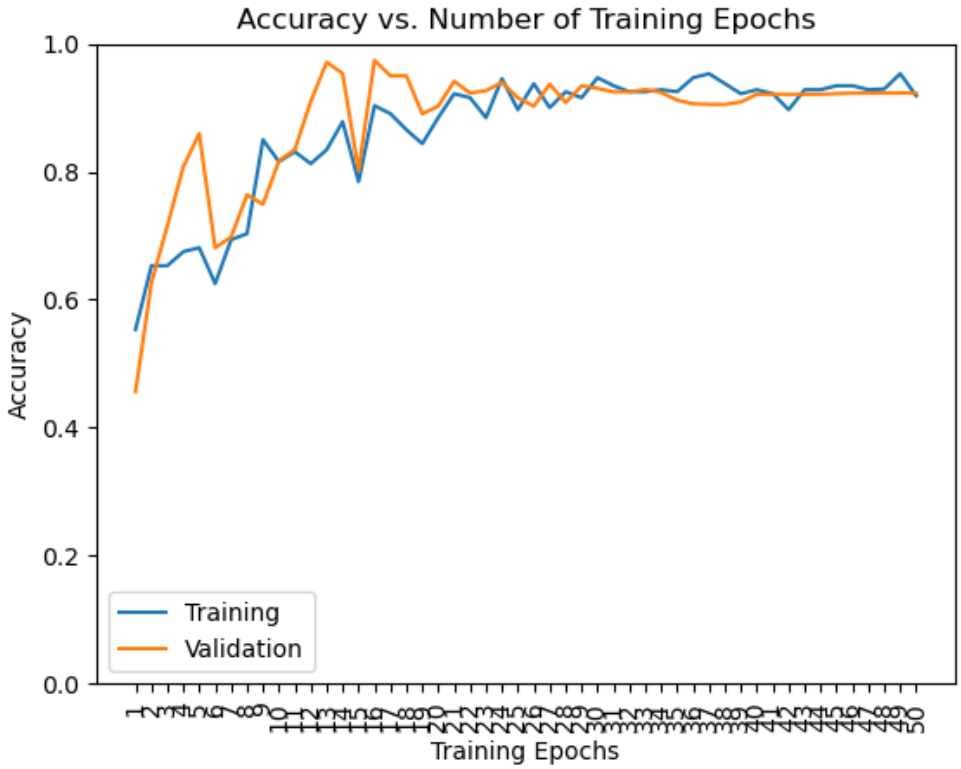


Figure 3.7: Training and validation accuracy during epochs

2. if the prediction has a score higher than 80% (independently by its class), we insert the predicted label in the buffer

3. only when the circular buffer has more than k votes, the person is estimated with the most frequent label in the buffer (majority voting mechanism)

In this way the system is robust against mispredictions, also because it makes use of the temporal information about each person. It isn't common that an individual switches from mask to no mask in a few frames period. In this way, the correct label is preserved even in presence of strong confidence mispredictions. On the other way, if a person appears to take off its mask and then changing its real label, the system reacts rapidly as the buffer fills with correct estimations. In

Fig. 3.8 we can see the final output of our approach.

Figure 3.8: Face mask detection with $N = 31$ and $k = 3$.

3.4 Tools

The whole project has been tested on Windows 10 and written in *Python* [61], the most used programming language for Data Science and Machine Learning tasks. Alongside with it, *Tensorflow* [62] and *PyTorch* [63] are the frameworks used to build the deep learning models and making them run on GPU. For the image processing, *OpenCV* [64] has been intensively adopted.
Making the ground-truth annotations in order to validate the object detectors has been possible thanks to *OpenLabeling* [65].

Chapter 4

Experimental Results

In this chapter will be presented the numerical results regarding the performance of each classifier. The benchmark for the YOLO classifier and the RootNet part are done using a popular open-source dataset called *Oxford Town Centre* dataset [66], consisting in a 5 minutes video of a crowded street, with a number of people varying from 10 to 32. The whole system has been tested on a 4GB NVIDIA GeForce GTX 1650 Max-Q, a mobile graphics card thought for laptops and power saving environments. Despite being a GPU not expressly projected for ML computing tasks, we obtained pretty good results. Unfortunately,

Figure 4.1: Oxford Town Centre frame

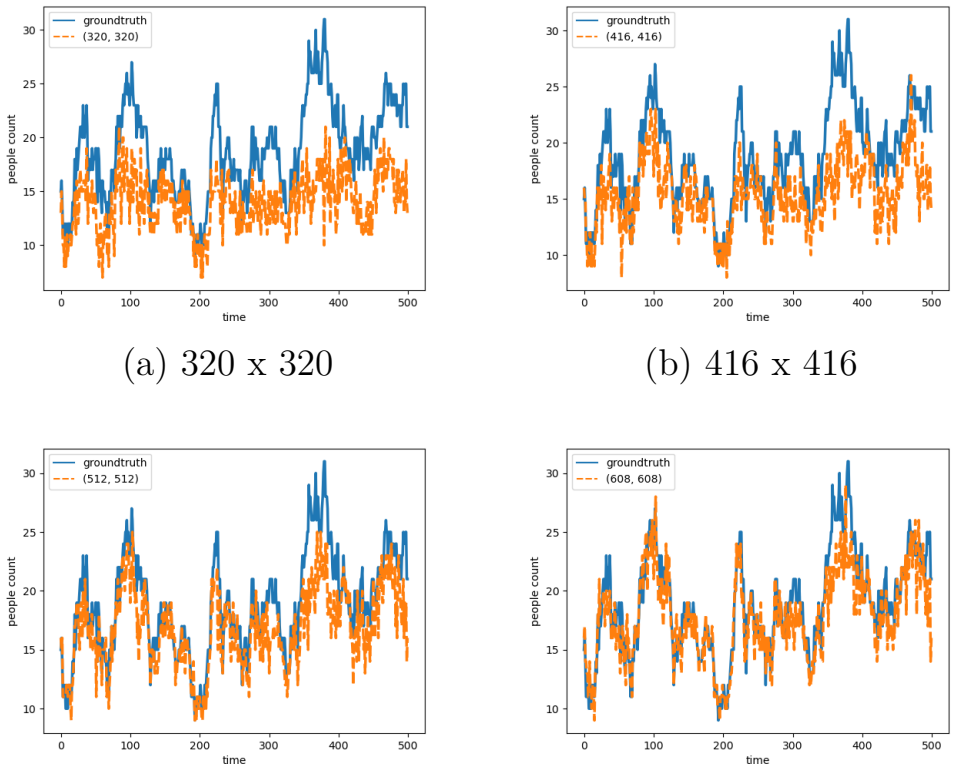
this dataset isn't suited for validating our mask classifier, since there aren't any people wearing masks. To do so we used a PUC camera surveillance footage, where both classes are present.

4.1 People counting

The counting people problem is addressed by the YOLO classifier, that can receive as input images of different size. YOLO accepts only a fixed size range of dimension, such as:

$$d = 320 + 96 \cdot k \ \ where \ k \ in \ \{0, 1, 2, ...\}$$

Varying the dimension of the images (*i.e.* varying the dimension of the net) consists in a variation of the performance. The trade off is, as usual, between having a greater accuracy versus having faster speed (measured in FPS). Fig. 4.2 shows the accuracy, during the whole five



(a) 320 x 320

(b) 416 x 416

(c) 512 x 512

(d) 608 x 608

Figure 4.2: Ground-truth *vs.* predicted number of people

minutes of video, of the classifier taking in input images of different size.

When the resolution is low Fig. 4.2 (a), the predicted output (yellow line) shows a consisting gap respect to the ground-truth (blue line). This gap becomes close to 0 with higher values of resolution Fig. 4.2 (d).
We then take a look of the medium FPS for each detector during the whole process, Fig. 4.3. The faster model, as easily predictable, is the one with the lowest resolution, achieving a mean of **15.7 FPS**. This could be easily explained by the fact that a lower number of weights in a neural network leads to a lower number of computations that the system is asked to process. Our slowest model runs on nearly 6 FPS. The dimension we choose for our following experiment is the one with size `(416, 416)`. It is important to show the other side of the medal by

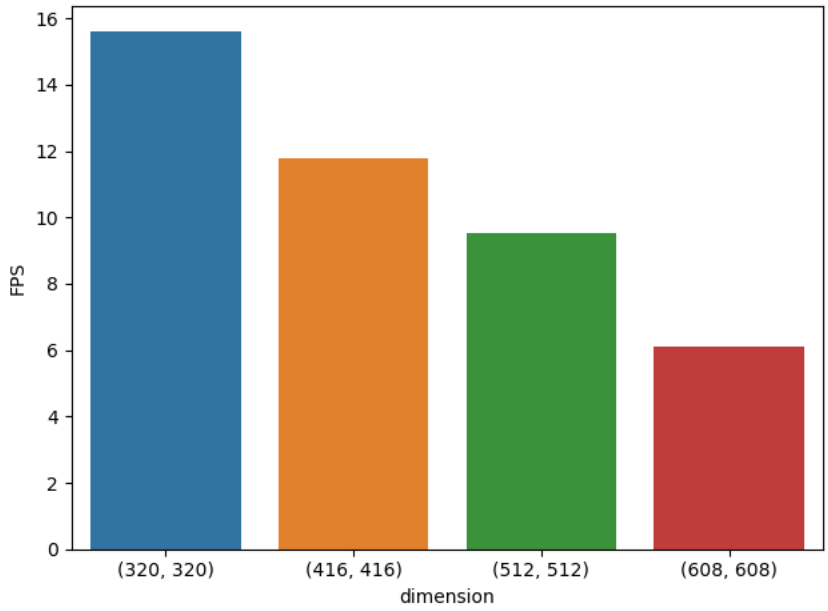


Figure 4.3: Medium FPS for several YOLO detectors

computing the Mean Absolute Error (MAE) to get a numerical value showing the error made by each detector. It is clear from Fig. 4.4 that using a better resolution leads to better detections; the best performing model (the one with net size `(608, 608)`) has a Mean Absolute Error equal to **1.74**, while the worst one scores a MAE of 4.89. It could be

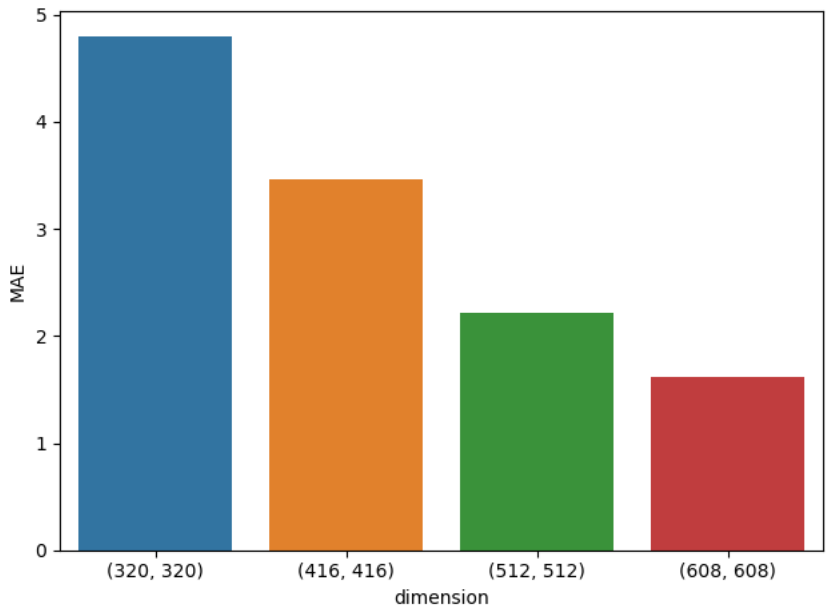


Figure 4.4: MAE for different YOLO detectors

worth analysing the error made by the model for each bucket of range of people inside the frame. In other words, we want to see how good is the detector when we have a `low`, `medium` or `high` density of people inside the image. To do so we divided the data distribution into three buckets:

- *less than or equal than 15*, corresponding to `low density`.
- *a value between 16 and 20*, corresponding to `normal density`.
- *higher than 20*, corresponding to `high density`, *i.e.* a crowded area.

The result is showed up in Fig. 4.5. It is interesting noticing that the detector fails to give an accurate estimate in high density crowded areas. This can be explained by the fact that, when the number of individuals increase, there is an higher chance that people can overlap on each other, creating occlusions, making the prediction unreliable.

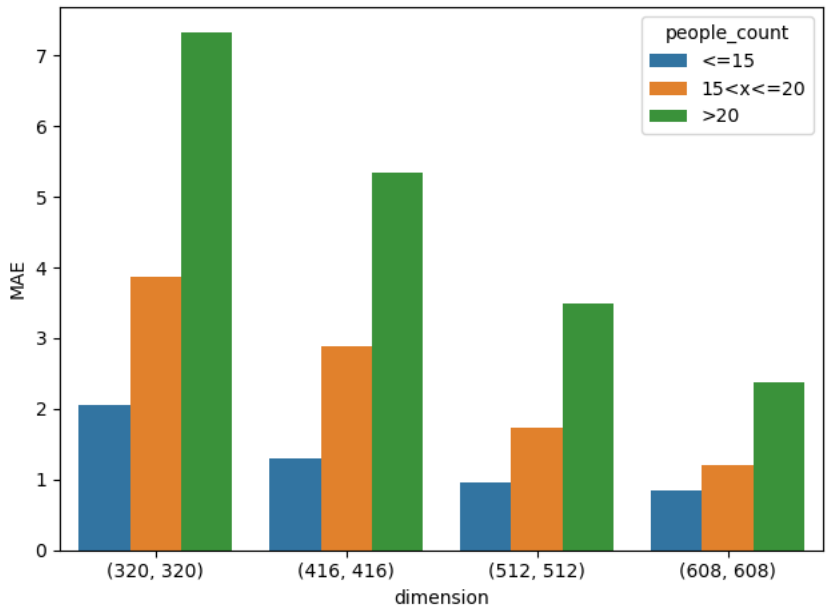


Figure 4.5: MAE for low, normal and high density regions

4.2 Monitoring social distancing

In order to validate the RootNet performance we used the Oxford Town Centre video as test set. The configuration of the model consisted in using YOLOv4 as person detector with input size of (416, 416), a distance threshold of 2 meters (*i.e.* an individual is classified at risk when is closer to another than the indicated distance) and a buffer size of 3. We compared the architecture with and without the tracking algorithm running.
First of all, we need to define some performance measures for binary classification problems. Our specific case, infact, consists of two classes: *people that are close to each other*, *i.e.* at risk, and *people distant from each other*. Fig. 4.6 contains a popular layout that allows visualization of the performance of a classifier, called *confusion matrix*:

- **True Positive (TP)**: are the number of cases when the predicted positive class matches with the actual class (positive)
- **False Positive (FP)**: are the number of cases when the predicted

class is positive but the actual class is negative

- **False Negative (FN)**: are the number of cases when the the predicted class is negative but the actual class is positive
- **True Negative (TN)**: are the number of cases when the predicted negative class matches with the actual class (negative)

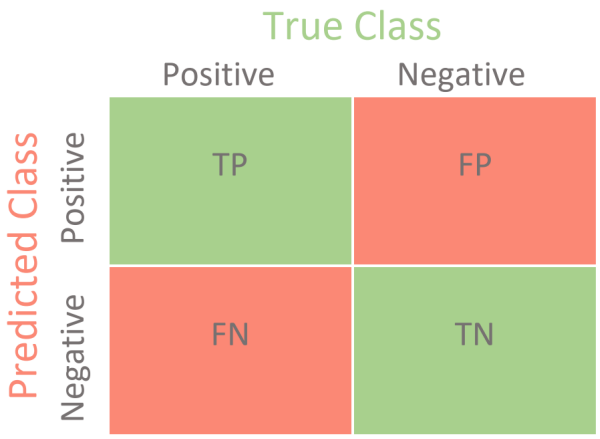


Figure 4.6: Confusion matrix

In our problem the positive class are the people **distant** from each other, while the negative class are the people **close** to each other.
Now we can define some useful performance measures:

- **Recall or Sensitivity or True Positive Rate**: is the proportion of positives that are correctly identified

$$TPR = \frac{TP}{TP + FN}$$

- **Precision**: is the fraction of positives among all items identified as positives

$$PRE = \frac{TP}{TP + FP}$$

- **Specificity or Selectivity or True Negative Rate**: is the proportion of negatives that are correctly identified

$$TNR = \frac{TN}{TN + FP}$$

- **Accuracy**: is the proportion of correct predictions among the total number of instances

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

- **F1 score**: is the harmonic mean of recall and precision

$$F1 = 2 \cdot \frac{PRE \cdot TPR}{PRE + TPR}$$

It is worth taking into account all of this measures since each of them contains some weak sides. For example, accuracy can lead to misleading results when the classes are unbalanced while F1-score ignores the True Negatives.
Now we can take a look at the results of our model. In particular, Fig. 4.7 shows the confusion matrix for our classifier without the tracking running (*i.e.* the prediction has been made by considering the single frame).

As we can see from table 4.1, the model has a pretty good hit rate for both classes but associated with a poor precision. The overall accuracy of the model is **0.70**.

Measure	**Close**	**Distant**
Precision	0.65	0.61
Recall	0.80	0.77
F1 score	0.71	0.68

Table 4.1: RootNet validation measures

We should now compare the performances of the architecture when the tracking algorithm is on. We can clearly see in Fig. 4.8 that the

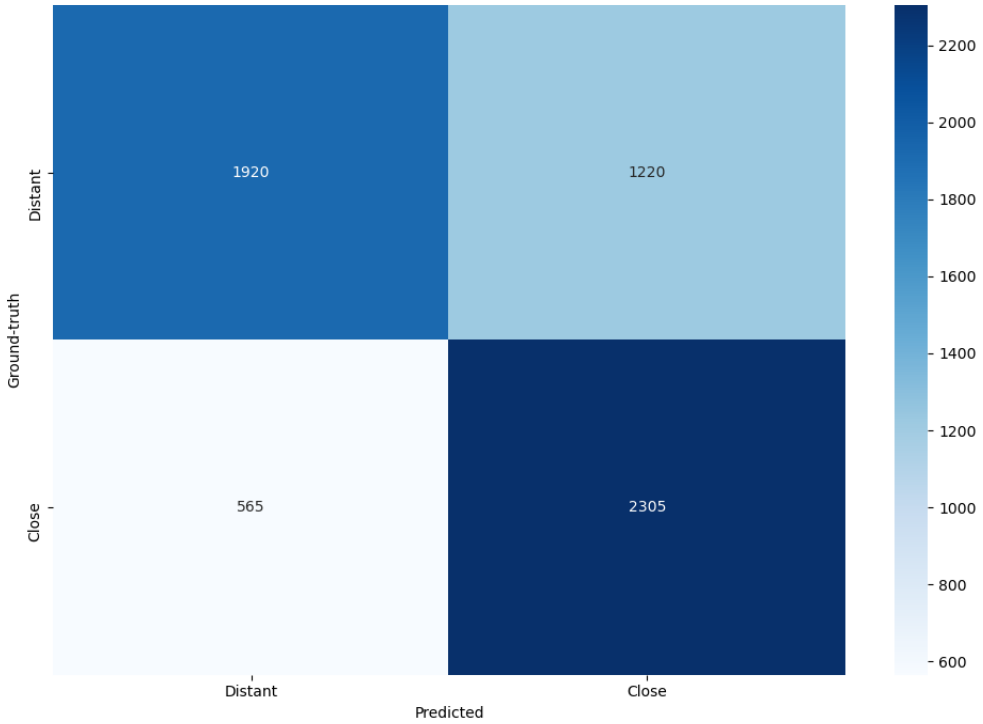


Figure 4.7: RootNet confusion matrix

number of correct predictions for both classes increased. Investigating deeply in the numbers we obtain the results in table 4.2. Both recall and precision increased, scoring an overall accuracy of **0.75**. Apparently, the adoption of the tracking system improves the classifier performance by gaining 5% in terms of accuracy.

Measure	**Close**	**Distant**
Precision	0.72	0.68
Recall	0.82	0.79
F1 score	0.76	0.73

Table 4.2: RootNet with tracking validation measures

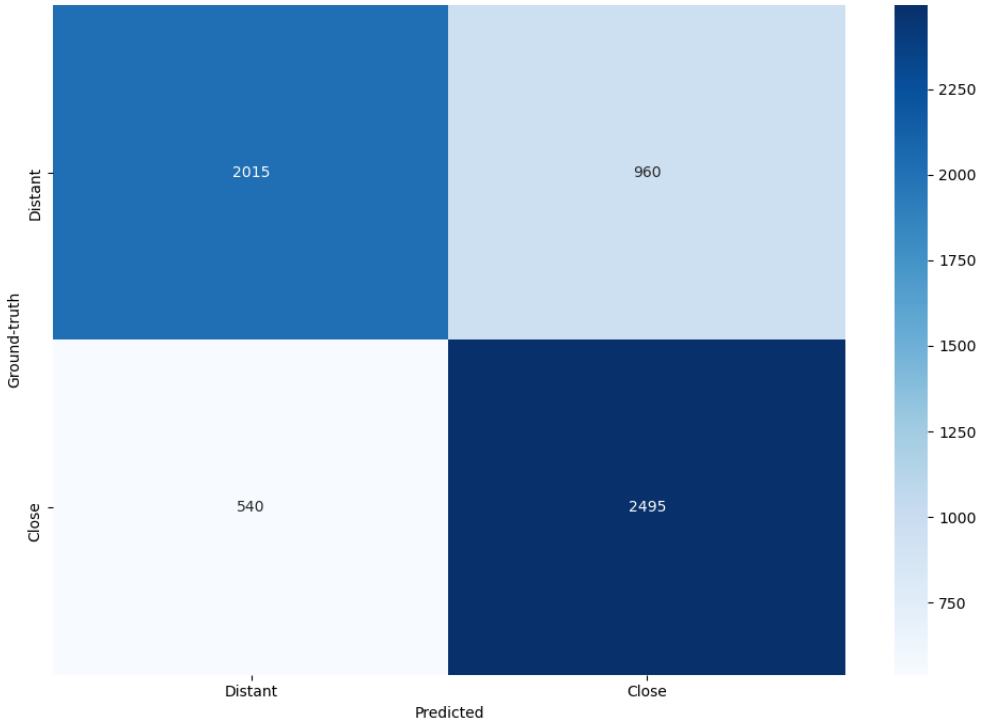


Figure 4.8: RootNet with tracking confusion matrix

4.3 Detecting people wearing masks

The Oxford Town Centre dataset isn't a good test set for this kind of task, since it does not contain any person wearing mask. In order to have a valid test set we used a video coming from PUC surveillance camera where a mixture of people with and without mask are present. Since our approach is multi-stage, we decided to divide this paragraph in a section where we test the object detection system (*i.e.* the Human pose estimator detecting faces) and a section to evaluate only the mask classifier.
We propose also a comparison between the YOLOv4 model fine-tuned on a mask training set and our model.

4.3.1 Object detection

In order to give a reliable comparison with another popular approach, we trained a YOLOv4 model on 1370 images containing people wearing or not wearing face masks. The dataset comes from an open-source Kaggle dataset called *Face mask dataset YOLO format* [67], that stores both images and annotations in YOLO format. In Fig. 4.9 we can see samples of pictures contained in the dataset. As we can clearly notice, the quality and resolution of these images are quite good, much different from the one coming from CCTV sources. In order to validate this

Figure 4.9: Examples of images coming from Kaggle dataset

approach we choose another public face mask annotated dataset with nearly 900 images, available on Kaggle, called *"Face mask detection"* [68]. The type of images collected coming from this source reflects the ones present in the YOLOv4 training set, that let us think that the model will perform pretty well on this task.
To measure the performances of an object detection model we use the so called *Intersection over Union (IoU)*, that represents the intersection over the union of the real bounding boxes and the predicted ones. An IoU of 1 represents perfectly overlapped bounding boxes. We can set different threshold values for the IoU to decide if the prediction is valid or not. The most common value for accepting a valid detection is 0.5, in this case we have a True Positive. If the IoU is lower than 0.5, we have a False Positive and, finally, if an object is present in the frame but the model doesn't detect it we have a False Negative. The next step is calculating Precision and Recall for all the objects detected, that

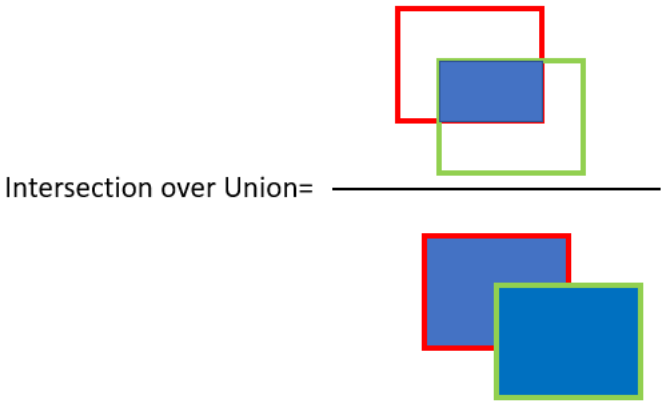


Figure 4.10: Intersection over Union

are associated with a confidence score. The final measure we want to obtain is called *mean Average Precision (mAP)*, and it is calculated using 11 equally spaced recall levels and taking, for each point, the interpolated precision p_{interp}. The final formula to calculate the AP is the following:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r)$$

Fig. shows the relationships between the values of precision and recall for different thresholds for the YOLOv4 model. An high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. This model reached an optimum AP value for mask and no mask class of **90.29%** and **81.55%** respectively. The overall mAP is of **85.92%**, a pretty good results.

Our model, specifically trained on low resolution and small size images, do not perform well on this kind of dataset. Infact, the mAP is of 40.3%. It is also worth to pinpoint the fact that, despite the images contained well shown faces, a lot of False Negatives, *i.e.* cases when the face was present but the model failed to detect it, were triggered

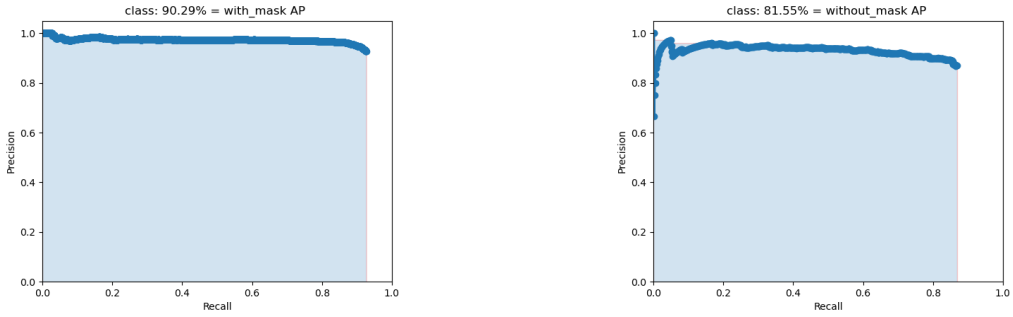


Figure 4.11: Precision vs Recall YOLOv4 on Face mask dataset

because the Human Pose estimator struggled to estimate the entire skeleton when only half body or only the face was showed. Another important factor to be considered is that we're now evaluating the model on raw images, without the possibility of using the tracking utilities only possible on video frame, that, as we will see, gives to the system a huge boost in terms of accuracy. Now is the turn to evaluate

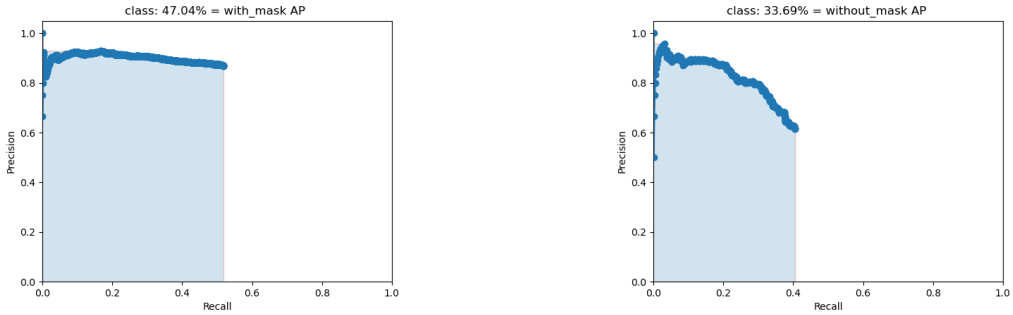


Figure 4.12: Precision vs Recall our model on Face mask dataset

both approaches on the original data sources that the projects was thought for: surveillance camera footage.
The YOLOv4 Average Precision collapse when it takes to detect small objects on low resolution frames. The mask AP, that before reached a value of 90%, now dropped to 15%, while the AP for the no mask class scored an AP of 58%. The mean Average Precision has a value of just 36.34%. The image on the right represents the log average miss

rate, *i.e.* the fraction of mispredicted objects per class, that reaches its highest value with the mask class with a rate of 0.88. Our model on the

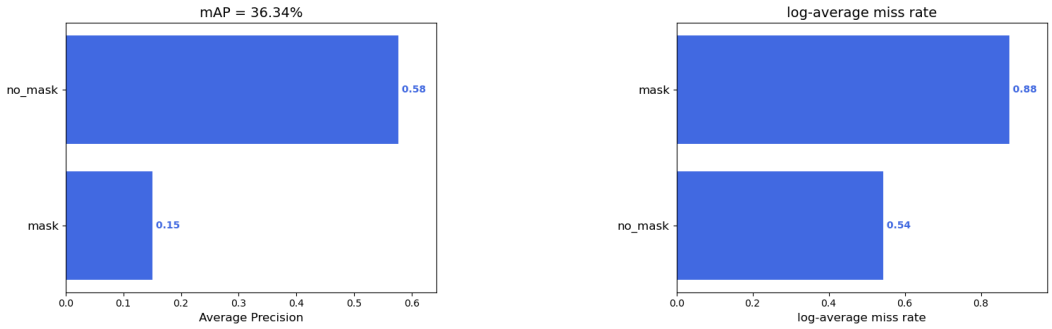


Figure 4.13: mAP and lamr for YOLOv4 on surveillance footage

video with the tracking enabled outperforms the YOLOv4 architecture, as we can clearly see in Fig. 4.14. Both classes scores an AP higher than 70%, with **77%** and **73%** for mask and no mask respectively. The mean Average Precision for the whole model is of **75.07%**. The log average miss rate drops to 0.39 and 0.23 respectively for mask and no mask. Our approach, by considering the temporal related informations, achieves to correctly classify also tiny challenging objects, that other methods struggled to detect.

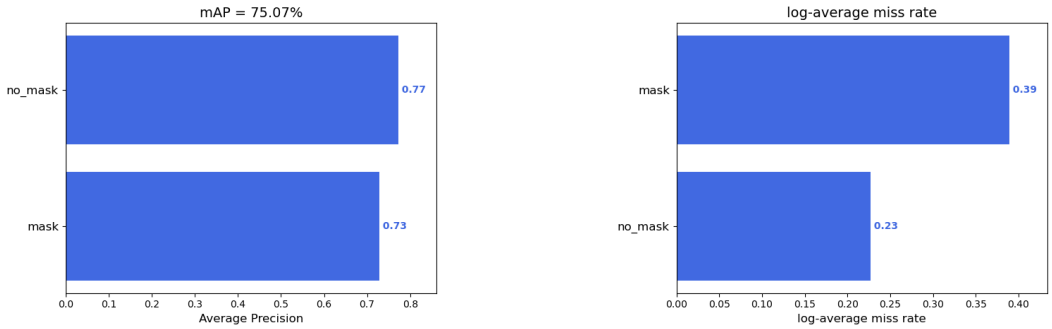


Figure 4.14: mAP and lamr for our model on surveillance footage

4.3.2 Image classifier

Since our approach consists in a multi stage classifier, it is important evaluate each single component independently. We decided to test our fine-tuned mask classifier, the BiT-M R101x1, on a huge dataset to see if it generalizes well. The chosen dataset is another open-source Kaggle dataset, called *"Face mask 12k images dataset"* [69], with 10000 images in the training set (divided equally as mask and no mask) and 2000 for the validation set. We used the training set as our test set. The important difference between this dataset and the previous ones is that each image represents a face (so we don't need any face detector), different from the others that in each picture could have been present several people.
By running our classifier on this test set we achieved a very good result, as clearly showed in Fig. 4.15. Being the two classes balanced (5000

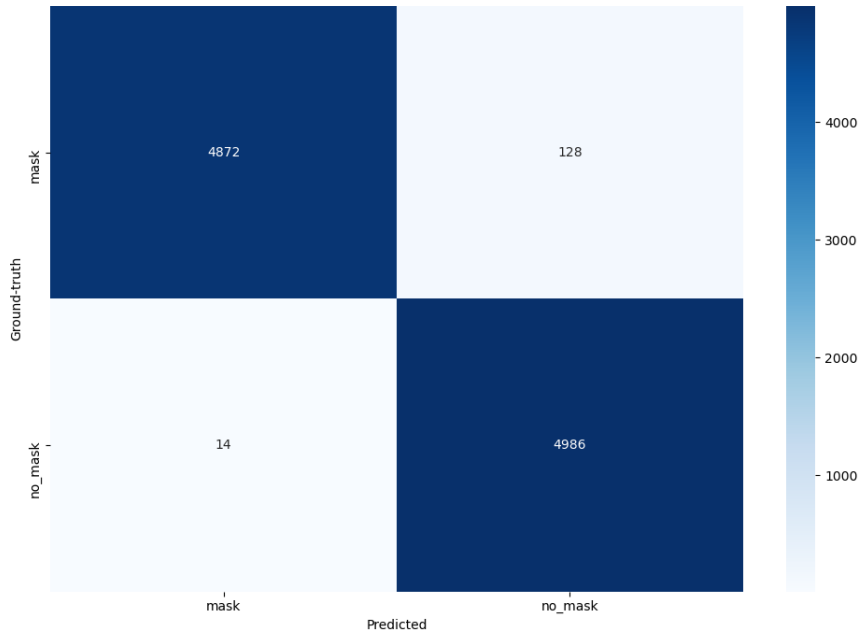


Figure 4.15: Mask classifier confusion matrix

images each), we can reliably taking into consideration the accuracy as performance measure, that reaches a value of **98,58%**. It is also worth calculating the other test measures, separately for each class. We can

be satisfied by the classifier performance since the final F1-score for both classes reaches a value of nearly **99%**. The following figure shows

Measure	**mask**	**no mask**
Precision	0.97	0.9972
Recall	0.997	0.975
F1 score	0.985	0.988

Table 4.3: BiT-M R101x1 validation measures

samples of wrong prediction made by the classifier for each of the two classes. The mispredictions made on the images on the first row can be explained by the fact that those are very uncommon types of mask that the classifier was never trained on, while the error made on the second row are due to occlusion, traslated images and person with beard, that makes the task more challenging.

Figure 4.16: Wrong predictions

Chapter 5

Conclusions and Future Works

The approach proposed in this work tries to solve three problems that are going to become crucial in every day life. The solutions proposed in the previous pages provide a reliable way to address each task, supplying a system that can be used in an everyday scenario. Hospitals, airports, schools, shops are the first candidates where our model can be deployed. This type of technology can be mixed and powered with alarms and monitor systems, providing, not only a visualization tool, but also a concrete way to protect lives in this particular historical moment. In this case, the need of having a real-time system can lead a modification in the basic structure of our approach by choosing less computationally expensive parts, such as a lightweight Human Pose Estimation network. The alternative to speed-up the process is using a head-detector instead of having a whole Human Pose Estimator.
Different type of tracking algorithm can be tried, such as KCF Tracker or MedianFlow tracker. Also the human detector can be switched to faster model, for example choosing to sacrifice accuracy by picking the Tiny version of YOLO, or switching to a complete different object detector such as Detectron2. An interesting choice could be the one of having a dedicated network for each task, using only single stage block to speed up the predictions. In this way the re-utilization and dependency of parts from each other can be eliminated during the

whole pipeline. This type of modifications would facilitate running the model on a embededd platform, such as Jetson Nano.
On the other side, in order to get a boost in accuracy, it could be worth collecting a greater number of training images, from heterogeneous surveillance cameras installed at different angles and perspective to the ground. If having available enough computing power, the mask classifier can be switched to the most performing BiT network, the BiT-L R152x4, a ResNet model 4 times wider than the one used in our approach.
With accurate classifiers, we can perform two extra steps:

- resolve the problem of beards: people with beard will be always classified as masked. This is due also to a low number of training images containing people with beard
- the problem can become a 3-class classification problem, where the extra class can be "mask wore incorrectly", that adds an extra level of difficulty to the problem, especially on low resolution images

Bibliography

[1] *Coronavirus disease (COVID-19) advice for the public*. 2020. URL: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public (cit. on p. 3).

[2] Nicholas R Jones, Zeshan U Qureshi, Robert J Temple, Jessica P J Larwood, Trisha Greenhalgh, and Lydia Bourouiba. «Two metres or one: what is the evidence for physical distancing in covid-19?» In: *BMJ* 370 (2020). DOI: 10.1136/bmj.m3223. eprint: https://www.bmj.com/content/370/bmj.m3223.full.pdf. URL: https://www.bmj.com/content/370/bmj.m3223 (cit. on pp. 3, 4).

[3] Joshua Payne. «Report on the Feasibility of Implementing PIC Codes on a GPU». In: (May 2012) (cit. on p. 10).

[4] P. Dollar, C. Wojek, B. Schiele, and P. Perona. «Pedestrian Detection: An Evaluation of the State of the Art». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2012), pp. 743–761 (cit. on p. 18).

[5] M. Li, Z. Zhang, K. Huang, and T. Tan. «Estimating the number of people in crowded scenes by MID based foreground segmentation and head-shoulder detection». In: *2008 19th International Conference on Pattern Recognition*. 2008, pp. 1–4 (cit. on p. 18).

[6] N. Dalal and B. Triggs. «Histograms of oriented gradients for human detection». In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1 (cit. on p. 18).

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. «Object Detection with Discriminatively Trained Part-Based Models». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645 (cit. on p. 18).

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. arXiv: 1506.02640 [cs.CV] (cit. on pp. 18, 20).

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV] (cit. on pp. 19–21, 34).

[10] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: 1405.0312 [cs.CV] (cit. on p. 19).

[11] Mingxing Tan, Ruoming Pang, and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. 2019. arXiv: 1911.09070 [cs.CV] (cit. on p. 19).

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. «ImageNet: A Large-Scale Hierarchical Image Database». In: *CVPR09*. 2009 (cit. on pp. 20, 41).

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2013. arXiv: 1311.2524 [cs.CV] (cit. on p. 20).

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. «SSD: Single Shot MultiBox Detector». In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2 (cit. on p. 20).

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. *Focal Loss for Dense Object Detection*. 2017. arXiv: 1708.02002 [cs.CV] (cit. on p. 20).

[16] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: 1911.11929 [cs.CV] (cit. on p. 20).

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition». In: *Lecture Notes in Computer Science* (2014), pp. 346–361. ISSN: 1611-3349. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23 (cit. on p. 20).

[18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. *Path Aggregation Network for Instance Segmentation*. 2018. arXiv: 1803.01534 [cs.CV] (cit. on p. 20).

[19] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV] (cit. on p. 20).

[20] Antoni Chan and Nuno Vasconcelos. «Bayesian Poisson Regression for Crowd Counting». In: Nov. 2009, pp. 545–551. DOI: 10.1109/ICCV.2009.5459191 (cit. on p. 21).

[21] Nikos Paragios. «A MRF-based Approach for Real-Time Subway Monitoring». In: (Jan. 2001) (cit. on p. 21).

[22] Valério Nogueira, Hugo Oliveira, José Silva, Thales Vieira, and Krerley Oliveira. «RetailNet: A Deep Learning Approach for People Counting and Hot Spots Detection in Retail Stores». In: July 2019. DOI: 10.1109/SIBGRAPI.2019.00029 (cit. on p. 21).

[23] Lucas Massa, Adriano Barbosa, Krerley Oliveira, and Thales Vieira. «LRCN-RetailNet: A recurrent neural network architecture for accurate people counting». In: Apr. 2020 (cit. on pp. 21, 23).

[24] Ke Chen, Chen Change Loy, Shaogang Gong, and Tao Xiang. «Feature Mining for Localised Crowd Counting». In: Jan. 2012. DOI: 10.5244/C.26.21 (cit. on p. 21).

[25] Victor Lempitsky and Andrew Zisserman. «Learning To Count Objects in Images.» In: Jan. 2010, pp. 1324–1332 (cit. on p. 23).

[26] Y. Wang and Y. Zou. «Fast visual object counting via example-based density estimation». In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 3653–3657 (cit. on p. 23).

[27] Bolei Xu and Guoping Qiu. «Crowd density estimation based on rich features and random projection forest». In: Mar. 2016, pp. 1–8. DOI: 10.1109/WACV.2016.7477682 (cit. on p. 23).

[28] J. Hall. *Social Distance Monitoring*. 2020. URL: https://levelfivesupplies.com/social-distance-monitoring/ (cit. on p. 24).

[29] StereoLabs. *Using 3D Cameras to Monitor Social Dis-tancing*. 2020. URL: https://www.stereolabs.com/blog/using-3d-cameras-to-monitor-social-distancing/ (cit. on p. 24).

[30] Sizhen Bian, Bo Zhou, Hymalai Bello, and Paul Lukowicz. «A Wearable Magnetic Field Based Proximity Sensing System for Monitoring COVID-19 Social Distancing». In: *Proceedings of the 2020 International Symposium on Wearable Computers*. ISWC '20. Virtual Event, Mexico: Association for Computing Machinery, 2020, pp. 22–26. ISBN: 9781450380775. DOI: 10.1145/3410531.3414313. URL: https://doi.org/10.1145/3410531.3414313 (cit. on p. 24).

[31] Adarsh Jagan Sathyamoorthy, Utsav Patel, Yash Ajay Savle, Moumita Paul, and Dinesh Manocha. *COVID-Robot: Monitoring Social Distancing Constraints in Crowded Scenarios*. 2020. arXiv: 2008.06585 [cs.RO] (cit. on p. 24).

[32] LandingAI. *Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace*. 2020. URL: https://landing.ai/landing-ai-creates-an-ai-tool-to-help-customers-monitor-social-distancing-in-the-workplace/ (cit. on p. 24).

[33] Dongfang Yang, Ekim Yurtsever, Vishnu Renganathan, Keith A. Redmill, and Ümit Özgüner. *A Vision-based Social Distancing and Critical Density Detection System for COVID-19*. 2020. arXiv: 2007.03578 [eess.IV] (cit. on pp. 24, 25).

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. arXiv: 1506.01497 [cs.CV] (cit. on p. 24).

[35] Prateek Khandelwal, Anuj Khandelwal, Snigdha Agarwal, Deep Thomas, Naveen Xavier, and Arun Raghuraman. *Using Computer Vision to enhance Safety of Workforce in Manufacturing in a Post COVID World*. 2020. arXiv: 2005.05287 [cs.CV] (cit. on pp. 25, 32).

[36] Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. *Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques*. 2020. arXiv: 2005.01385 [cs.CV] (cit. on p. 25).

[37] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. *Detectron*. https://github.com/facebookresearch/detectron. 2018 (cit. on p. 28).

[38] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. 2017. arXiv: 1703.06870 [cs.CV] (cit. on p. 28).

[39] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. *RMPE: Regional Multi-person Pose Estimation*. 2016. arXiv: 1612.00137 [cs.CV] (cit. on p. 28).

[40] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. *Spatial Transformer Networks*. 2015. arXiv: 1506.02025 [cs.CV] (cit. on p. 28).

[41] Bin Xiao, Haiping Wu, and Yichen Wei. «Simple Baselines for Human Pose Estimation and Tracking». In: Apr. 2018. ISBN: 978-3-030-01230-4. DOI: 10.1007/978-3-030-01231-1_29 (cit. on p. 28).

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV] (cit. on p. 28).

[43] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. *Cascaded Pyramid Network for Multi-Person Pose Estimation*. 2017. arXiv: 1711.07319 [cs.CV] (cit. on p. 28).

[44] Ke Sun et al. *High-Resolution Representations for Labeling Pixels and Regions*. 2019. arXiv: 1904.04514 [cs.CV] (cit. on pp. 28, 34).

[45] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. *DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation*. 2015. arXiv: 1511.06645 [cs.CV] (cit. on p. 28).

[46] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. *DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model*. 2016. arXiv: 1605.03170 [cs.CV] (cit. on p. 28).

[47] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2018. arXiv: 1812.08008 [cs.CV] (cit. on p. 28).

[48] G. Rogez, P. Weinzaepfel, and C. Schmid. «LCR-Net: Localization-Classification-Regression for Human Pose». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1216–1224 (cit. on p. 29).

[49] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. «Single-Shot Multi-Person 3D Pose Estimation From Monocular RGB». In: *3D Vision (3DV), 2018 Sixth International Conference on*. IEEE. Sept. 2018. URL: http://gvv.mpi-inf.mpg.de/projects/SingleShotMultiPerson (cit. on p. 29).

[50] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. *Camera Distance-aware Top-down Approach for 3D Multi-person Pose Estimation from a Single RGB Image*. 2019. arXiv: 1907.11346 [cs.CV] (cit. on pp. 29, 30, 34).

[51] Che-Yen Wen, Shih-Hsuan Chiu, Yi-Ren Tseng, and Chuan-Pin Lu. «The Mask Detection Technology for Occluded Face Analysis in the Surveillance System». In: *Journal of forensic sciences* 50 (June 2005), pp. 593–601. DOI: 10.1520/JFS2004409 (cit. on p. 30).

[52] A. Nieto-Rodríguez, Manuel Mucientes, and Victor Brea. «System for Medical Mask Detection in the Operating Room Through Facial Attributes». In: June 2015, pp. 138–145. ISBN: 978-3-319-19389-2. DOI: 10.1007/978-3-319-19390-8_16 (cit. on p. 30).

[53] M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker. «Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition». In: *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. 2019, pp. 1–5 (cit. on p. 30).

[54] Mohamed Loey, Gunasekaran Manogaran, Mohamed Taha, and Nour Eldeen Khalifa. «A Hybrid Deep Transfer Learning Model with Machine Learning Methods for Face Mask Detection in the Era of the COVID-19 Pandemic». In: *Measurement* 167 (July 2020), p. 108288. DOI: 10.1016/j.measurement.2020.108288 (cit. on p. 30).

[55] Mingjie Jiang, Xinqi Fan, and Hong Yan. *RetinaMask: A Face Mask detector*. 2020. arXiv: 2005.03950 [cs.CV] (cit. on p. 31).

[56] Dongxiao Li Bosheng Qin. *Identifying Facemask-wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19*. 2020 (cit. on pp. 32, 33).

[57] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. *Simple Online and Realtime Tracking with a Deep Association Metric*. 2017. arXiv: 1703.07402 [cs.CV] (cit. on p. 34).

[58] *Face mask detection in street camera video streams using AI behind the curtain*. 2020. URL: https://tryolabs.com/blog/2020/07/09/face-mask-detection-in-street-camera-video-streams-using-ai-behind-the-curtain/ (cit. on p. 39).

[59] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009 (cit. on p. 40).

[60] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. *Big Transfer (BiT): General Visual Representation Learning*. 2019. arXiv: 1912.11370 [cs.CV] (cit. on pp. 40, 41).

[61] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697 (cit. on p. 43).

[62] Martın Abadi et al. «Tensorflow: A system for large-scale machine learning». In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283 (cit. on p. 43).

[63] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 43).

[64] G. Bradski. «The OpenCV Library». In: *Dr. Dobb's Journal of Software Tools* (2000) (cit. on p. 43).

[65] J. Cartucho, R. Ventura, and M. Veloso. «Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots». In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2336–2341 (cit. on p. 43).

[66] Jules. Harvey Adam. LaPlace. *MegaPixels.cc: Origins, Ethics, and Privacy Implications of Publicly Available Face Recognition Image Datasets*. 2019. URL: https://megapixels.cc/ (visited on 04/18/2019) (cit. on p. 44).

[67] *Face mask dataset yolo format*. 2020. URL: https://www.kaggle.com/aditya276/face-mask-dataset-yolo-format (cit. on p. 53).

[68] *Face mask detection*. 2020. URL: https://www.kaggle.com/andrewmvd/face-mask-detection (cit. on p. 53).

[69] *Face mask 12k images dataset*. 2020. URL: https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset (cit. on p. 57).