POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Improvement of a Dataset for the Creation of an Automatic Mask and Respirator Detection System

Supervisors

Prof. Bartolomeo Montrucchio

Dr. Antonio Costantino Marceddu

Candidate

Luigi Di Sergio

December 2023

Summary

The COVID-19 pandemic made the whole world change its lifestyle in several respects: working at home has become much more common and people have started to pay more attention to safety. Social distancing and the use of protective masks helped the population to contain the spread of the epidemic, proving to be an effective means of dealing with this type of problem. Enforcement has been entrusted to staff dedicated to this task, but it is a job easily replaced through the use of automation.

The *FMR-DB (Facial Masks and Respirators Database)* was developed in order to provide a basis for the training of machine learning systems capable of performing this task not only in the context of disease containment but in all contexts in which the use of masks plays a key role in safety protection. It consists of 4200 images collected using various search engines and the images are divided according to the presence or absence of masks or respirators and occlusions within the image. The images were selected to offer diversity in the selection of the subjects depicted, trying to represent different sexes, ages, and ethnicities to offer as universal a model as possible.

The labeling process was carried out by means of *LabelImg*, a graphical image annotation tool, and the label files are saved in *PascalVOC* and *YOLOv7* format: two of the most common labeling systems were chosen to give the possibility of adapting the database to different possible uses according to the desired architecture.

Although both formats were chosen, the FMR-DB was built with the aim of training a model using *YOLOv7*, a deep learning-based recognition system based on the YOLO family of models that have become widely used in the industry in the last few years. YOLOv7 has been chosen because both its speed and accuracy are state-of-the-art and because it offers a range of models designed to be used on different types of hardware.

A recognition system trained with FMR-DB could be used to ensure safety in the working environment, in hospitals, and in all contexts where the use of respirators prevents damage to health. The database could be improved by expanding the image collection to cover different lighting conditions and perspectives to cover extreme cases.

Acknowledgements

I would like to thank prof. Bartolomeo Montrucchio and dr. Antonio Costantino Marceddu for the support provided during the realization of this thesis.

A special thanks to my mother Rossana, to my father Franco and to my sister Chiara, who have raised, supported and helped me throughout my entire life.

Finally, I would like to thank my other family members and friends who have supported me along the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In 2020, the world was shaken by the COVID-19 pandemic, a disease that spread among some 700,000,000 people and caused around 7 million deaths [29]. The population had to adapt to a different lifestyle to cope with the spread of the virus: general lockdowns were instituted to prevent infection, forcing part of the population to work from home, social gatherings were limited or banned, and protective masks were used when necessary.
Various studies have shown the effectiveness of the use of protective masks in preventing infection and this solution has been adopted in many countries as a measure to contain contagion.

Regulations on the use of masks in public varied according to the environment, but regardless of the context, most of the monitoring of compliance was left to the staff of the various institutions. This process is an unconstructive task when carried out by on-duty workers as it takes time away from the tasks they should be performing on a regular basis. The concept does not only apply to the near case of disease prevention but is applicable to any context in which the use of masks or respirators can protect people's health and safety such as hospital facilities or industrial environments where particulates or fumes are produced.

This control task should be delegated to an automated system capable of recognising in real time whether people are actually wearing a face mask or respirator and which type it is, regardless of the context. The use of protective masks or respirators causes problems in existing facial recognition algorithms, which is why it was decided to create the *FMR-DB (Facial Masks and Respirators Database)* [30].

1.2 Thesis Structure

This thesis continued the work of Marceddu, Antonio Costantino and Montrucchio, Bartolomeo on the development of the FMR-DB collecting additional images and carrying out the labelling process. Furthermore, a YOLOv7 model was trained using using fine tuning and achieving excellent results.

The thesis is divided into five chapters described in detail below.

- Chapter 2, called *Machine Learning and Convolutional Neural Networks*, serves as an introduction to the world of machine learning and convolutional neural networks. Basic concepts fundamental to understanding the work done are described.
- Chapter 3, called *YOLO*, describes the evolution of YOLOv7, the object detector chosen to utilise the data from the FMR-DB.
- Chapter 4, called *FMR-DB (Facial Masks and Respirators Database)*, describes the structure of the database in detail with regard to images and labels.
- Chapter 5, called *Results* contains the settings used to train the model and the representation of the results obtained.
- Chapter 6, called *Conclusions*, provides comments on the previous chapters and discusses possible improvements and future development.

Chapter 2

Machine Learning and Convolutional Neural Networks

The objective of the FMR-DB is to provide a basis on which to develop applications capable of recognising the type of mask worn by subjects in a control environment. For this to happen, machine learning algorithms must be used to train a convolutional neural network.

2.1 Machine Learning

Machine learning (ML) is a branch of *artificial intelligence* (AI) that uses data and algorithms to understand arguments by gradually improving its effectiveness.
The origin of the name 'machine learning' is attributed to Arthur L. Samuel who in 1959 used the term to describe how a computer can learn to play checkers [31]. In little more than 60 years, this field of study has become extremely relevant and its applications be can witnessed in the most diverse fields. This development was made possible by the improvement of computational power and data storing possibilities.
Machine learning algorithms are divided into three categories that differ according to the way the machine learns:

- Supervised learning: The system learns by taking into account the data provided by the developer and the outputs that need to be produced by finding a way to link the former to the latter. In mathematical terms, a vector of characteristics is provided and it corresponds to the input data. Through a step-by-step process of optimising an objective function, the goal is to generate a function capable of predicting the output associated with new input data.

- Unsupervised learning: In this case, the system does not obtain a desired output, but only input data, and the machine's task is to find in those data what features represent or link them together by searching for hidden patterns.

- Reinforcement learning: The system is placed within an environment in which a series of actions can be performed and its objective is to maximise the value of an objective function that depends on the possible actions performed.

2.2 Neural Network and Backpropagation

Neural networks (or *artificial neural networks*) is a branch of machine learning in which the model is built based on the neural structure of biological brains as shown in Figure 2.1. In this model, a series of nodes, called neurons, have connections, called edges, which, like synapses, allow signals to be transferred between the nodes.

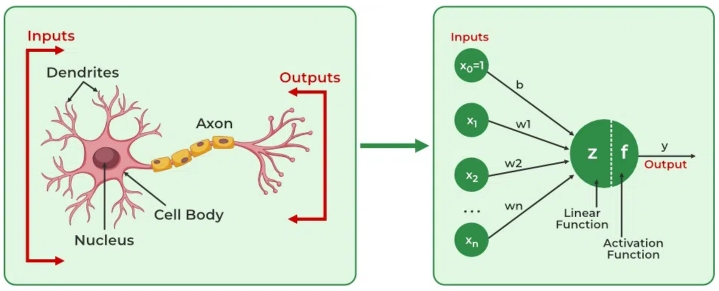


Figure 2.1: Representation of a neuron and an artificial neuron [1].

The nodes perform computations and transfer numerical signals to the other neurons to which they are connected by considering the weights associated with the traversed edges. Neurons are organised in layers and transfer information from an input layer to an output layer via one or more hidden layers. The structure formed this way is depicted in Figure 2.2.

The structure that is formed can be represented by means of a weighted directed graph where the neurons are the vertices and the synapses are the edges to which a weight is associated. Neurons can be modelled as a mathematical structure that transforms input values x_i into an output y through a non-linear function. Each connection between neurons is associated with a weight w_j and in addition the parameter w_0 called bias has to be considered : unprocessed data is obtained by averaging these values and then has to be passed to an activation function $f(y)$ to normalise the result and get z.

$$y = \sum_{n=i}^{N} w_i x_i + w_0$$

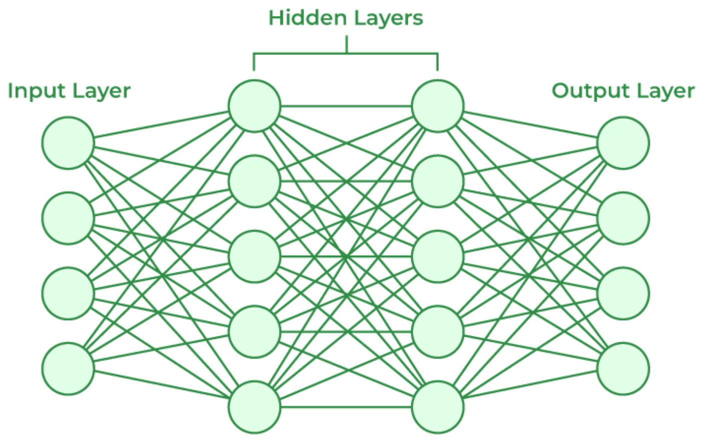


Figure 2.2: Representation of a neural network [2].

$$z = f(y)$$

2.2.1 Activation Function

Depending on the case, different activation functions can be used. The most popular in the field of machine learning are the following:

- Sigmoid function.
$$f(x) = \frac{1}{1 + e^x}$$

- Tanh function.
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- ReLU function.
$$f(x) = \max(0, x)$$

- Leaky ReLU function.
$$f(x) = \max(0.1x, x)$$

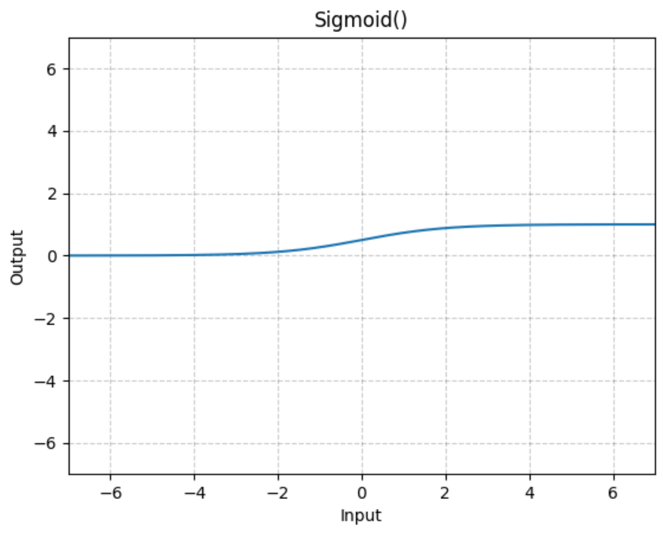


Figure 2.3: Sigmoid function [3].

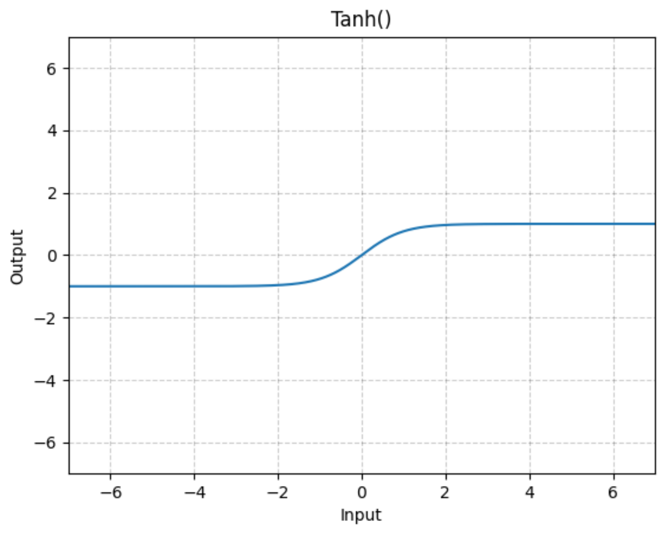


Figure 2.4: Tanh function [4].

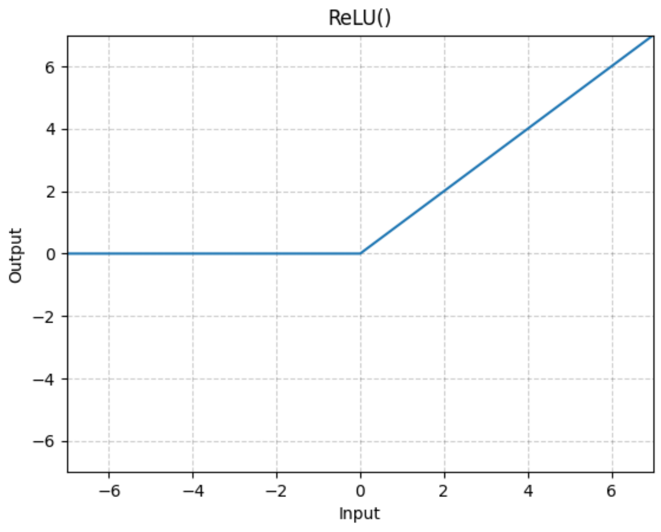


Figure 2.5: ReLU function [5].

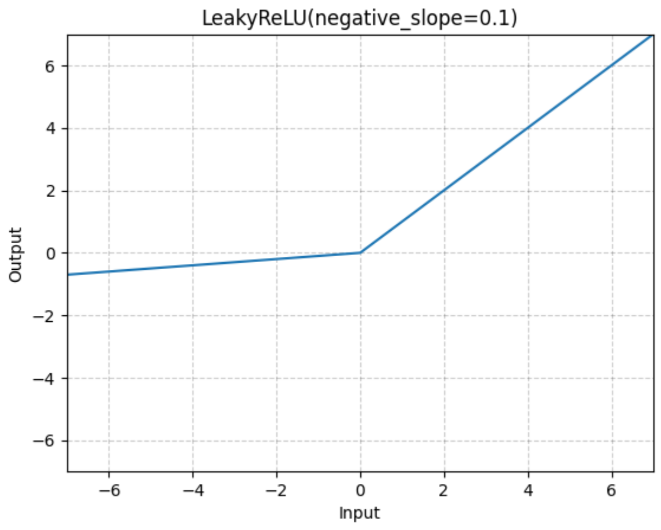


Figure 2.6: Leaky ReLU function [6].

- SiLU function.

$$f_k(x) = x_k \sigma(x_k)$$

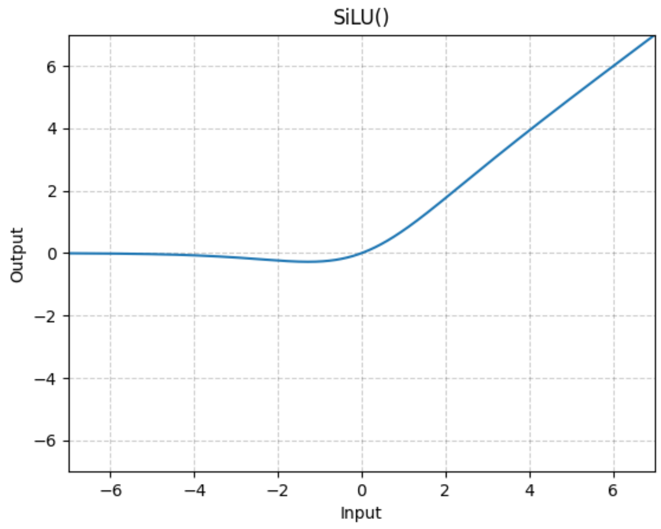


Figure 2.7: SiLU function [7].

- Softmax function.

$$f(x_i) = \frac{exp(x_i)}{\sum_j exp(z_j)}$$

The choice of which activation function to use depends above all on the type of variable to be predicted and the structure of the network: some functions are better suited for regression, others for classification. Also the position of the layer considered must be taken into account.

2.2.2 Feedforward Neural Networks

Feedforward neural networks (FNN) are neural networks within which the flow of information goes from the input layer to the output layer via the hidden layers without creating cycles within the graph. This type of neural network is in contrast to recurrent neural networks (RNN) where data can flow in both directions.

Feedforward neural networks are trained using the method of backpropagation, in which internal parameters are adjusted to improve the result of the outcome of a pass through the network.
Backpropagation is applied in a multi-level network in which the following steps take place:

- Input data traverse the network from the first layer to the last passing through hidden layers.
- Edge weights are adjusted to reduce the error between the predicted result and the correct label.
- Iteration continues until a minimum in the loss function is reached.

Researching the minimum in the loss function is done through the use of gradient descent: an algorithm that searches for local minimum within differentiable functions. Often a variation of this, stochastic gradient descent, is used as a matter of efficiency.

2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of feedforward neural network in which learning occurs through the optimisation of filters (or kernels) that are applied to the input data through convolution [32].
Convolution is a mathematical operation on two functions f and g that indicates how the shape of the former is modified by the latter and is expressed by the symbol $*$. Specifically, it is defined as the integral of the product of the two functions after one of them has been flipped around the y-axis and shifted. A graphical representation of the convolution of functions is shown in Figure 2.8.

$$(f * g)(t) := \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

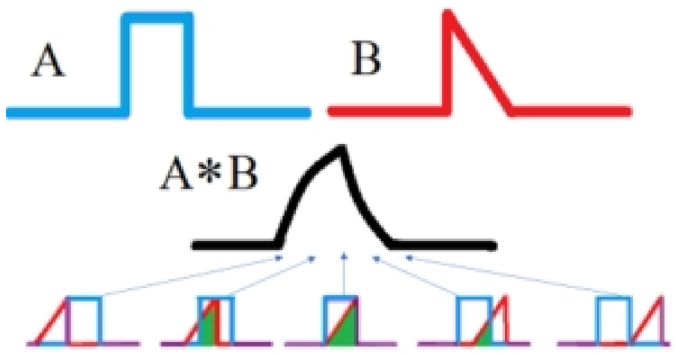


Figure 2.8: Example of convolution between two functions A and B [8].

Convolutional neural networks is well suited for use when the input data are images: computer vision, image recognition, image classification are just some of the uses of these algorithms but to work well they need large labelled datasets of images.

Digital images can be represented as arrays of pixels with an associated value that defines the hue, saturation and brightness of that pixel. Each pixel in the image is therefore associated with a vector of information with three layers of depth as it is necessary to consider the three colour components.

In the convolution process, a matrix called a filter is superimposed on the image and the filter values are multiplied with the underlying image values to obtain a numerical result for the pixel in question. This operation is continued by running the filter over the image until all pixels have been analysed as shown in Figure 2.9.

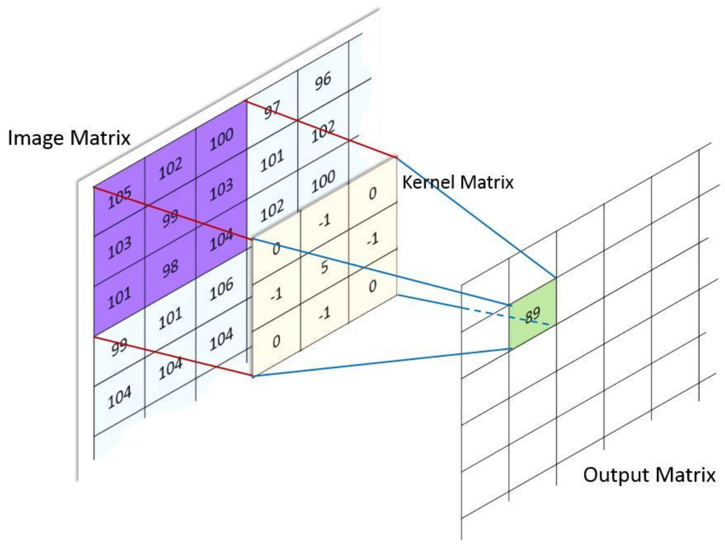


Figure 2.9: Convolution of a filter on an image [9].

2.3.1 Pooling

The convolution layer is followed by the pooling layer in which the output size of the layer to which it is applied is reduced, allowing faster computation and decreasing the overfitting problem.

Pooling occurs by grouping the output of a cluster of neurons into a single result to be passed on to the next layer. This operation can be done globally over the entire image or locally in a specific area. There are two types of pooling: average pooling where the average value of the group considered is taken and max pooling where the maximum is used instead [33].

2.3.2 Fully Connected Layers

The last step of a layer in a convolutional neural network is the activation layer. The output of the pooling is passed to an activation function, which has the task of normalising the result by associating the data with a value between 0 and 1 or between -1 and 1 depending on the activation function used.

In the final part of the network there are the fully connected layers within which the final classification takes place. The neurons in these latter layers are connected to all the activation functions of the previous layer and process the data after flattening the input received. In this section, a label is assigned to the input data in order to associate it with a class.

2.3.3 Feature Extraction

A convolutional neural network is composed of several layers and different features are analysed depending on the depth at which they are positioned within the system. In the first convolution layers, low-level features such as edges and colour are captured (these types of features are displayed in the Figure 2.10), while going further into the network, more and more complex structures are identified until the required feature can be recognised.

After the image has passed through the network, a class is associated to it. The result of this operation is used to evaluate the performance of the system as errors are penalised within the loss function. The network learns by trying to minimise the deviation between the predicted results and the actual data classes. Different loss functions can be used depending on the type of task performed by the network.

2.3.4 Fine Tuning

An alternative to fully training a convolutional neural network is to do fine tuning. This technique consists of using a pre-trained model as a starting point and then adding more information by using what has already been learnt previously. This is possible because simple features common to all images (such as curves and edges) are recognised in the first layers of a network, while the recognition of more complex structures is left to the deeper layers. Using this property, it is possible to refine an already trained model by adding information (a new input image database) related to the specific case study. This makes it possible to train models from small databases and to greatly reduce the time required to produce an efficient model.

2.3.5 Data Augmentation

Data augmentation is the process by which the dataset to be passed to the neural network for training is expanded through modifications made to copies of the source

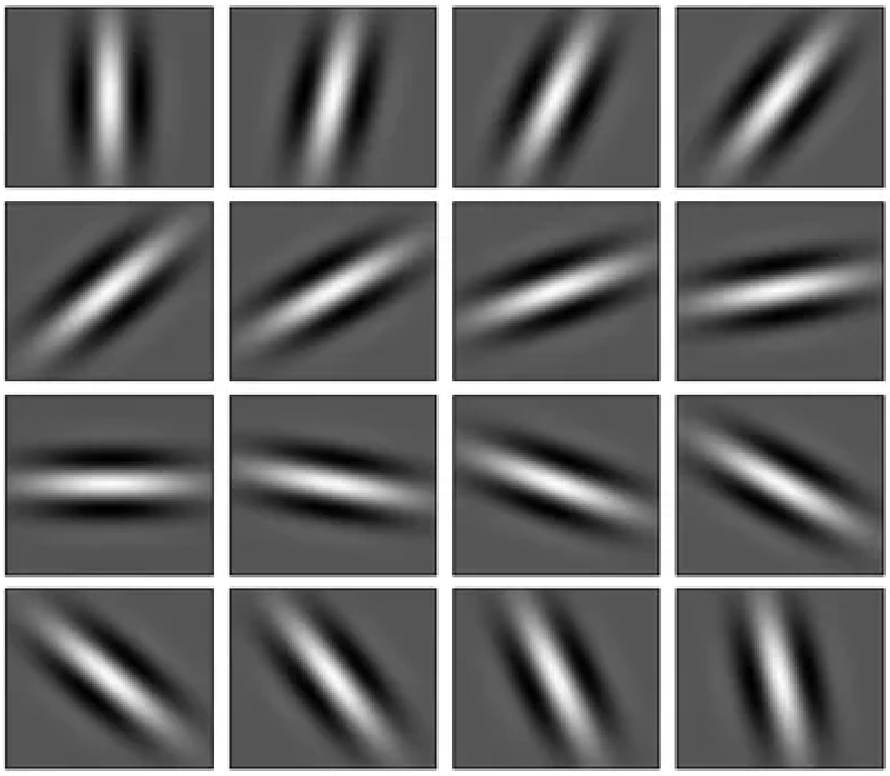


Figure 2.10: Gabor's Functions [10].

data [34]. The aim is to expand the initial dataset and counteract the problem of overfitting. In the case of images, data augmentation techniques modify the initial data by altering its geometry, colours or by combining them. There are many techniques for modifying images and some examples are given below. As for colours, hue, saturation and value are changed according to weights that affect the different elements. The geometry of the images can be modified by rotating, scaling, translating, shearing or flipping them on different axes. The combination of images takes place in the mosaic, creating a new image that combines elements of the original dataset to create a new one. In mixup, instead, the images are superimposed by blending them together. Some examples of data augmentation techniques are shown in Figure 2.11.

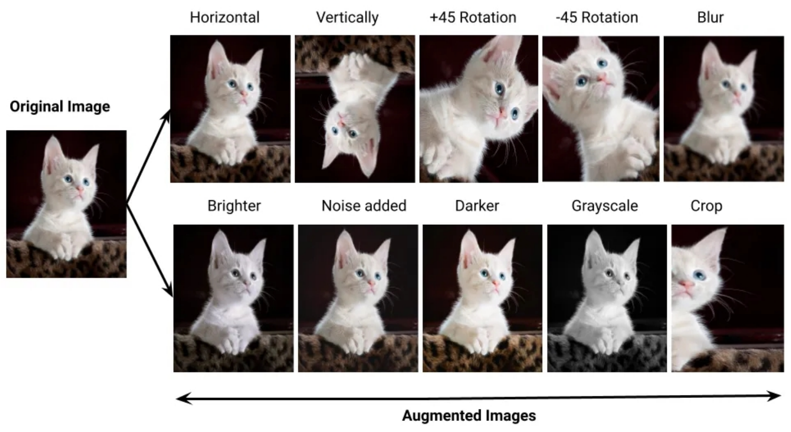


Figure 2.11: Example of data augmentation techniques [11].

2.4 Metrics

In the field of image recognition, several metrics are used to measure the effectiveness of the trained system. In order to understand the metrics considered, a trivial case with only one label is examined: input data have to be processed by the model and may or may not be associated with the label. Since the data may or may not belong to the label class, there are four possible scenarios:

- True positive (TP): The data is correctly associated with the class to which it belongs.
- True negative (TN): The data is correctly not associated with the class to which it does not belong.
- False positive (FP): The data is erroneously associated with the class to which it does not belong.

- False negative (FN): The data is erroneously not associated with the class to which it belongs.

		Predicted Output	
		Predicted Positive	Predicted Negative
Truth	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Truth Table

Depending on the use to be made of the trained network, some metrics are more significant than others, e.g. in the case of the recognition of tumour structures, it is important not to leave out any positives and the presence of a small number of false positive results but no false negative results can therefore be tolerated, thus seeking a high level of recall. The metrics that are going to be considered, in the output of the trainings done on YOLO, are represented according to the confidence used by the system to produce that result.

2.4.1 Accuracy

Accuracy measures the number of correct predictions in relation to the total number of predictions. Accuracy does not describe the system taking into account the different distribution of elements for the different classes under consideration: in very unbalanced cases, a high accuracy might indicate correct behaviour with regard to the class with many elements while the least represented class performs incorrectly.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

2.4.2 Confidence

Confidence is a metric that indicates the belief that a statement is true. When a label with a high confidence value is assigned, there are few false positives as the system wants to make sure that it does not make a wrong assignment. However, this leads to an increase in the number of false negatives since, when in doubt, it prefers not to assign the label. Conversely, in the case of low confidence, the system assign the element to the class even when it is not convinced. The number of false negatives decreases but this association leads to mistakes and thus to an increase in false positives.

2.4.3 Intersection over Union (IoU)

The Intersection over Union (IoU) describes the overlapping surface between the predicted and actual bounding box as depicted in Figure 2.12. The higher the value of the IoU, the closer the prediction resembles the truth.

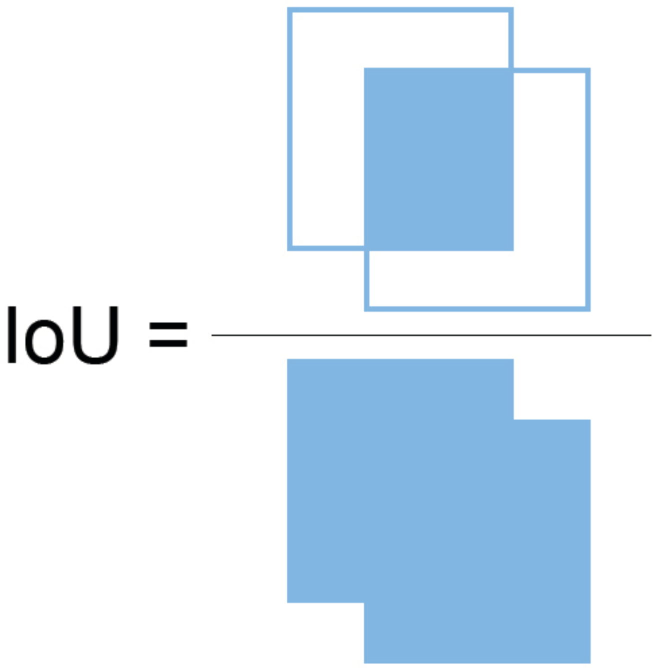


Figure 2.12: Intersection over Union

2.4.4 Precision

Precision measures the number of true positives in relation to the sum of true positives and false positives. A high precision value indicates the fact that the system recognises few false positives. The level of precision increases as confidence increases because the number of false positives is small.

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

2.4.5 Recall

Recall indicates the ratio between the number of true positives and the sum of the number of true positives and false negatives. It describes how well the system detects all positives in the given set to be analysed. The level of recall decreases as confidence increases because several positives are omitted from the prediction increasing the number of false negatives.

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

2.4.6 PR-Curve

The precision-recall curve is obtained by considering precision and recall values as a function of confidence. Since precision increases and recall decreases as the confidence value increases, this metric gives a better overall view of model's quality as it focuses on the confidence value that has a balance between precision and recall has to be found.

2.4.7 F1-Score

The F1-score is a metric developed to find the best confidence value at which the optimal F1-score is obtained.

$$Recall = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

2.4.8 Mean Average Precision (mAP)

Mean average precision (mAP) is a metric used to assess the quality of an object recognition model. Average precision (AP) is calculated as a function of precision and recall based on the confidence threshold. It is indicated as the weighted average of precision at each threshold where the weight is the variation in recall over the previous interval. Mean average precision is obtained by calculating the average of Average Precision over the total number of classes. Mean average precision is the metric used to benchmark the quality of object detection models as it considers the trade-off between precision and accuracy [35].

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

2.4.9 Confusion Matrix

The confusion matrix describes the behaviour of the system with respect to the different classes under analysis by correlating the predicted class with the actual class to which the images belong. It can be useful in understanding the strengths and weaknesses of the model produced as the performance of each class is specified. An example confusion matrix is presented in Figure 2.13.

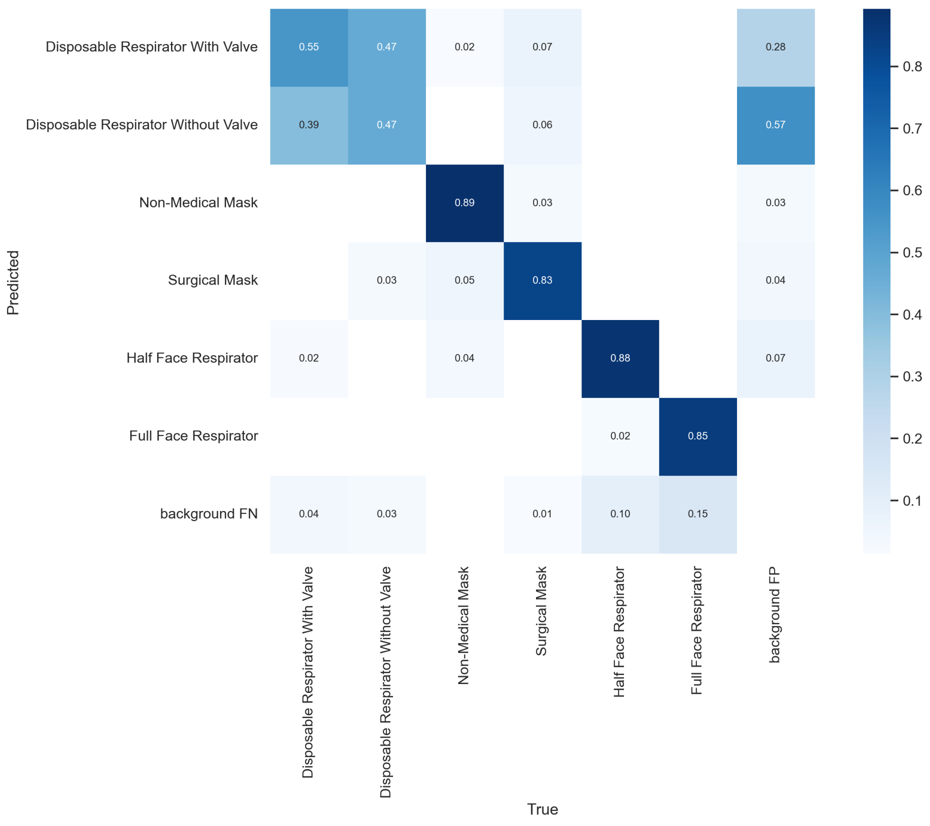


Figure 2.13: Example of a Confusion Matrix

Chapter 3

YOLO

3.1 YOLO

YOLO (You Only Look Once) is a real-time object detection model presented in 2015 in which recognition is treated as a single regression problem. Images are 'only looked once' to check the presence and position of objects as it goes directly from image pixels to the assignment of bounding boxes and class probabilities [12].

YOLO differed from previous recognition models that repurposed classifiers to perform detection presenting itself as a controversial alternative within these systems. The proposed approach uses a single neural network that considers the features of the entire image simultaneously to predict the bounding box of all classes.

The system divides the input image into an S x S grid and if the centre of an object lies within a grid cell, it is given the task of recognising the object. Each grid cell is given the task of predicting the bounding boxes and their confidence values, and the conditional probabilities of the classes. During testing, the conditional probabilities of the classes are multiplied by the confidence associated with the individual boxes to obtain the class-specific confidence for each box. A representation of the process is shown in Figure 3.1.

The original architecture of YOLO is based on 24 convolution layers and 2 fully connected layers as shown in 3.2. A faster version of the architecture called Fast YOLO, that used a neural network with fewer layers, was also presented achieving lower mAP but faster times.

During training, a single bounding box predictor is responsible for each object based on which prediction has the highest IoU on the ground truth.

At the end of the inference, it may be necessary to resort to non-maximal suppression (NMS) to resolve the issue of multiple bounding boxes generated for a single object. Non-maximal suppression is used to choose the best bounding box to describe the object and remove all others leaving only the best option.

The loss function that the model seeks to minimise directly corresponds to

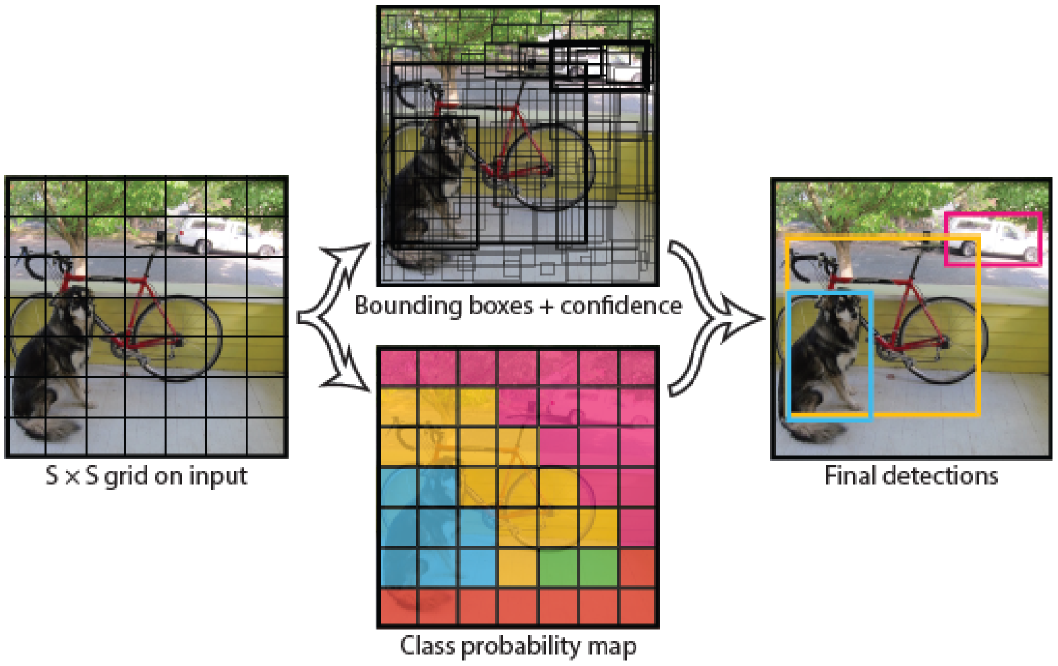


Figure 3.1: YOLO's detection system [12].

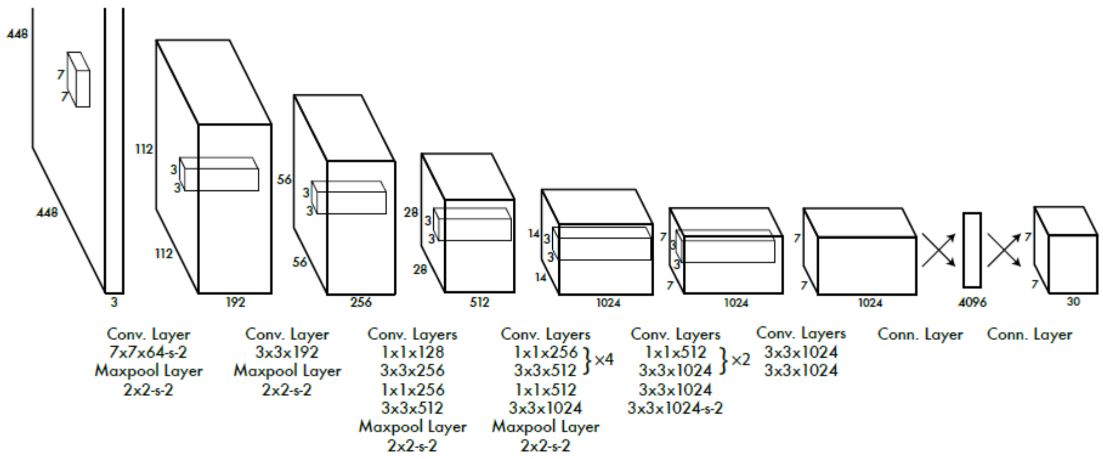


Figure 3.2: YOLO's architecture [12].

detection performance:

$$\lambda_{\mathbf{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] +$$

$$\lambda_{\mathbf{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] +$$

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 +$$

$$\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 +$$

$$\sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

The first two terms indicate localization loss in terms of position and size; the third and fourth terms penalise confidence when an object is present in the cell and when it is not; the last term relates to classification.

The YOLO model was generated by pre-training the convolutional layers on ImageNet 1000-class competition dataset and then training, validating and testing the network on PASCAL VOC 2007 and 2012 achieving results of 63.4 mAP and 45 fps.

YOLO's strengths, speed in recognition and good mAP results, have made it popular among object detection systems as it can be used in various fields (such as medical [36], surveillance [37], autonomous vehicles [38]) without requiring high-end hardware.

However, its structure has imposed limitations in particular areas of recognition. Since each grid cell is only able to predict two bounding boxes and one class, it has difficulties with small objects appearing in groups. It is also hard to generalize objects that are in configurations new to those with which it was trained.

3.2 YOLO's Evolution

The evolution of YOLO is now considered, taking into account the changes introduced and the improvements in performance from the first version to the most recent models. The metric used for performance evaluation is mean average precision and it is necessary to consider that for the first two versions of YOLO, PASCAL VOC 2007 and 2012 was used as the dataset for training and benchmarking but it was replaced for the later ones by Microsoft COCO (Common Objects in Context).

Furthermore, from YOLOv4 onwards there is change in vocabulary as it has become common to describe the architecture of object detectors in three parts: backbone, neck and head. The backbone is the part of the architecture where features are extracted from the input image and passed to the neck which combines and refines them. The head is the final component of the structure and is entrusted with the task of making predictions and performing post-processing operations [39].

3.2.1 YOLOv2

The first subsequent version is YOLOv2 (also known as YOLO9000), published in CVPR 2017. Compared to the first version, it came with several changes in structure and improvements in benchmarks [13].
It has a new architecture called Darknet-19 with 19 convolutional layers and 5 maxpooling layers and introduces several new features in the operation of the model. Some of the improvements are:

- Anchor boxes, a set of bounding boxes with predefined shapes used to generate the bounding boxes of objects by combining them with each other considering offset values. Their structure is shown in Figure 3.3.

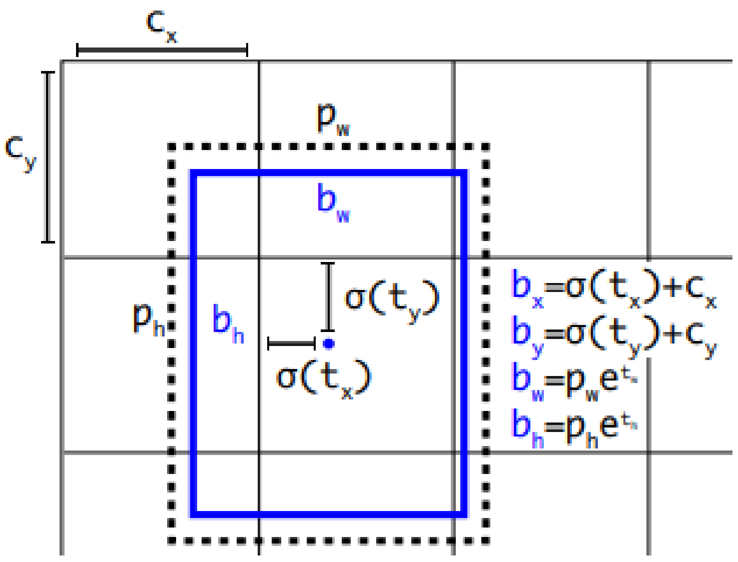


Figure 3.3: YOLOv2's anchor boxes [13].

- Batch normalization is introduced to improve accuracy and remove dropout from the model without overfitting.
- Fine-tuning at higher resolution (448x448) was carried out to improve the performance of the model.

- Multi-scale training is used due to the fact that YOLOv2 does not use fully connected layers. During the training phase, the input size was changed several times to make it robust.

YOLOv2 achieves a mAP of 78.6 on PASCAL VOC 2007 and PASCAL VOC 2012 with 40 fps, improving performance over YOLO at the expense of a few fps as presented in Figure 3.4.

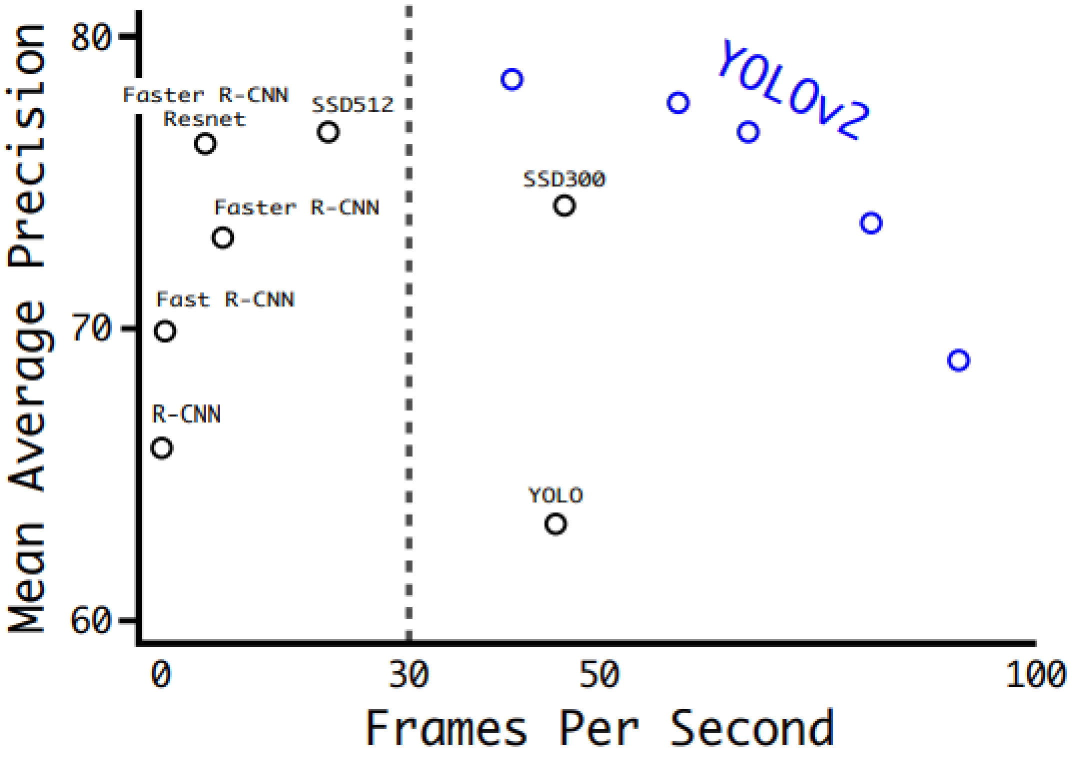
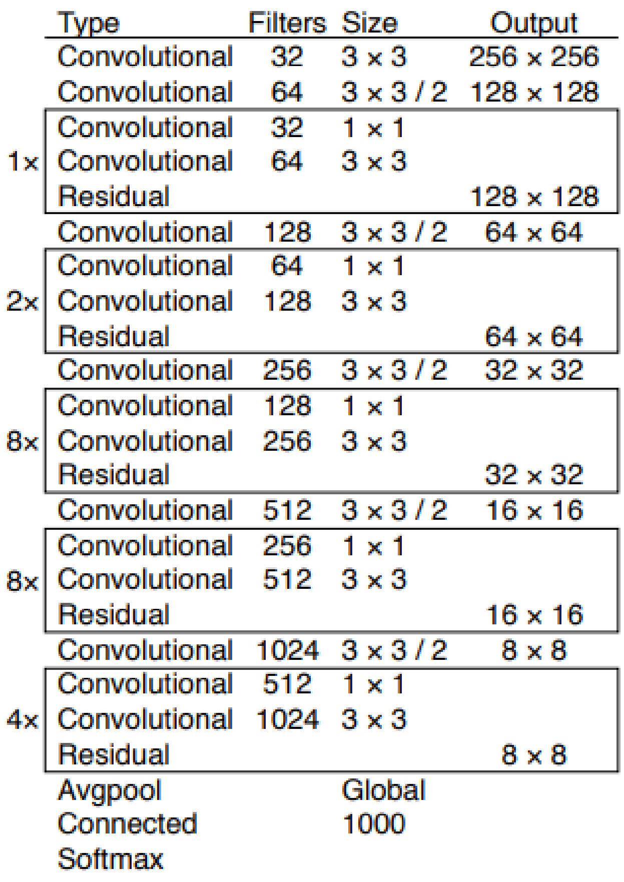


Figure 3.4: YOLOv2's benchmark [13].

3.2.2 YOLOv3

YOLOv3 was published in 2018 on ArXiv with the aim of improving accuracy and speed compared to YOLOv2 [14]. The architecture used in YOLOv3 was called Darknet-53, shown in Figure 3.5 because the number of layers was increased to 53, achieving excellent performance compared to other backbones in use at the time. The benchmarks are displayed in Figure 3.6.

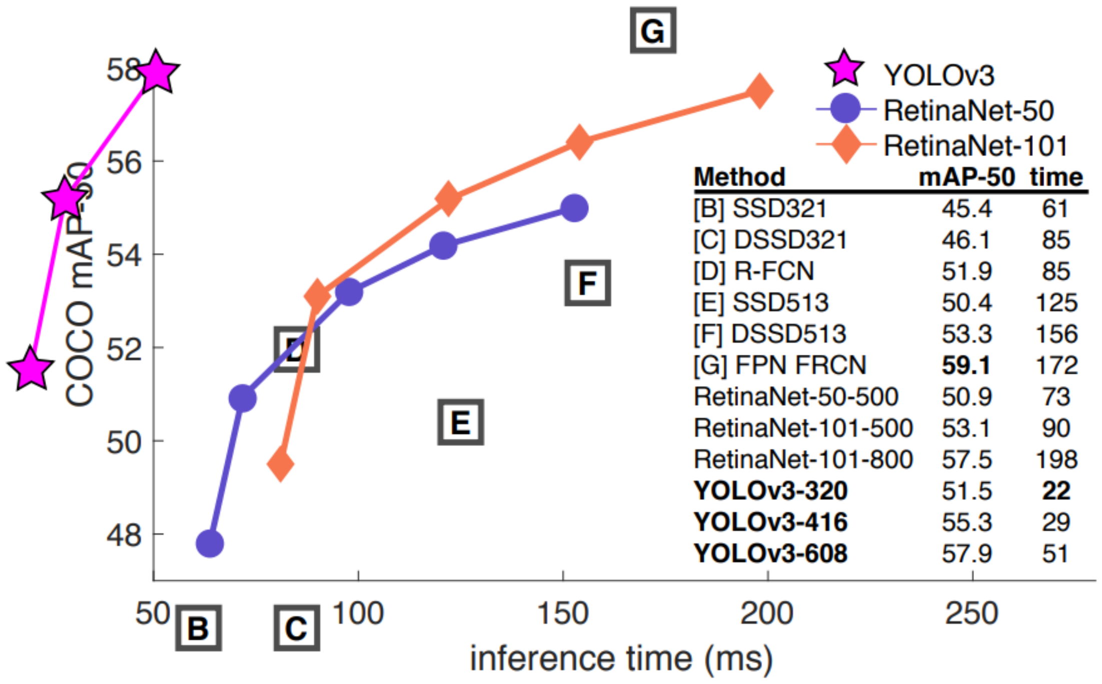
An important introduction made by YOLOv3 is the use of multi-scale predictions, which consists of predicting boxes at different scales. The technique is based on feature pyramid networks [40] and improves performance by solving one



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.5: YOLOv3's backbone [14].

of YOLO’s main shortcomings, namely the recognition of small objects. For object prediction, the concept of ’objectness’ was introduced, i.e. the probability that an object exists in the region of interest.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 3.6: YOLOv3’s benchmark from [14] as adapted from the focal loss paper [15].

3.2.3 YOLOv4

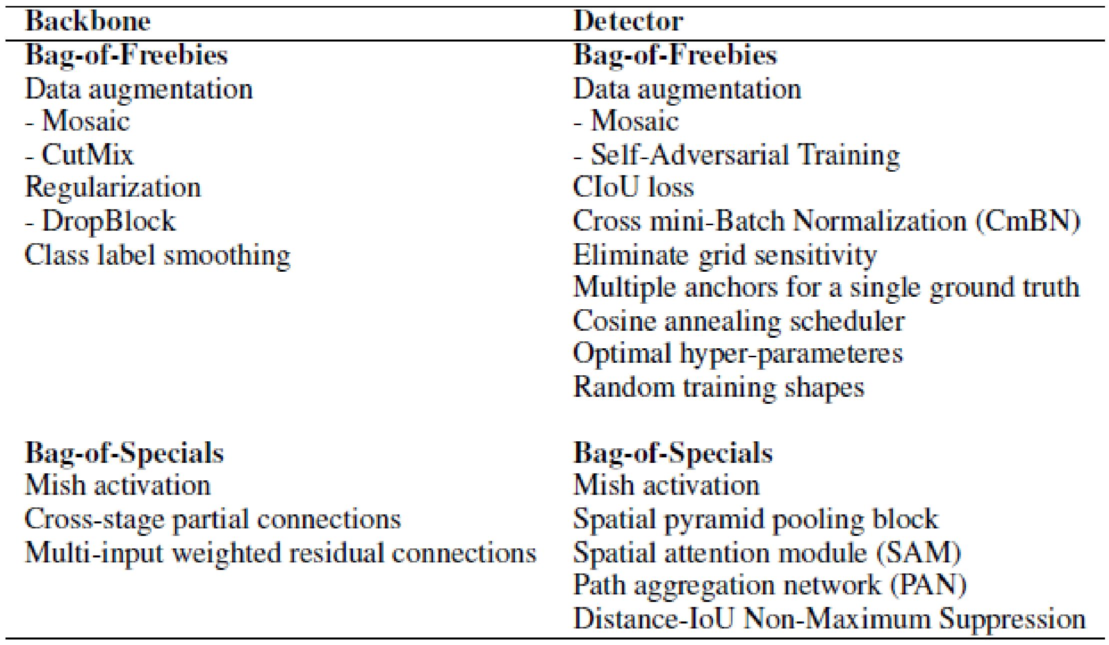
YOLOv4 was published in 2020 in [16] and is the first version not made by the original creator, J. Redmon. The creators of YOLOv4 continued the original project with the same philosophy of having a high-speed, open source object detector bringing several improvements to the previous version. In the paper, improvements are divided into two categories:

- Bag of freebies: a series of methods that are implemented in the training phase and that improve the model by increasing the training time but do not affect the inference time such as the use of data augmentation.
- Bag of specials: a set of plugin modules and post-processing methods that improve inference accuracy at the expense of a small time cost.

The authors experimented with the different elements described in these two sections in order to find an efficient balance. For the backbone architecture, a modified version of Darknet-53 was chosen, implementing cross-stage-partial-connections

(CSP). Spatial pyramid pooling (SPP) [41] and path aggregation network (PAN) [42] were introduced for the neck. Spatial pyramid pooling allows image features to be extracted independently of the scale at which they are represented while path aggregation network uses path integral graph to achieve fast convergence rate and great performance as displayed in Figure 3.8. YOLOv3 anchors were used for the head. The model resulting from the combination of these techniques was called CSPDarknet53-PANet-SPP.

The various techniques used by the bag of freebies include the use of mosaic data augmentation and the use of genetic algorithms to improve hyper-parameters in the first epochs of training. A list of the features introduced is shown in Figure 3.7.



Backbone	Detector
Bag-of-Freebies	**Bag-of-Freebies**
Data augmentation	Data augmentation
- Mosaic	- Mosaic
- CutMix	- Self-Adversarial Training
Regularization	CIoU loss
- DropBlock	Cross mini-Batch Normalization (CmBN)
Class label smoothing	Eliminate grid sensitivity
	Multiple anchors for a single ground truth
	Cosine annealing scheduler
	Optimal hyper-parameteres
	Random training shapes
Bag-of-Specials	**Bag-of-Specials**
Mish activation	Mish activation
Cross-stage partial connections	Spatial pyramid pooling block
Multi-input weighted residual connections	Spatial attention module (SAM)
	Path aggregation network (PAN)
	Distance-IoU Non-Maximum Suppression

Figure 3.7: YOLOv4's bag of freebies and bag of specials [16].

3.2.4 YOLOv5

YOLOv5 was released in 2020, a few months after YOLOv4, and takes the previous version as its inspiration to make changes and improvements, shown in Figure 3.9, with the aim of offering an easy-to-use system for real-world results [17].

The backbone of the architecture starts with CSPDarknet53 with the addition of an initial Stem phase, introduced to reduce the computational cost and then analysing the features at different scales. The neck uses and improves techniques similar to those introduced in YOLOv4 such as SPP and PAN while the head is

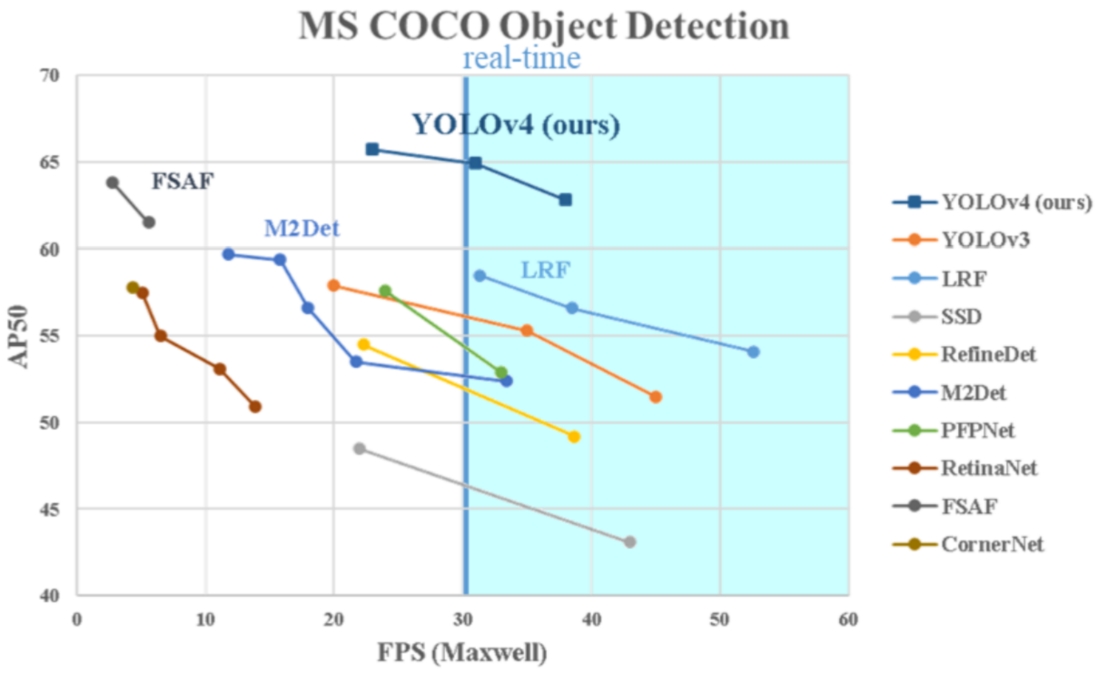


Figure 3.8: YOLOv4's benchmark [16].

similar to the one used in YOLOv3 [39]. YOLOv5 offers different models with different levels of performance depending on the type of machine that runs the system, ranging from YOLOv5n (Nano) to YOLOv5x (Extra Large) where accuracy increases at the expense of speed.

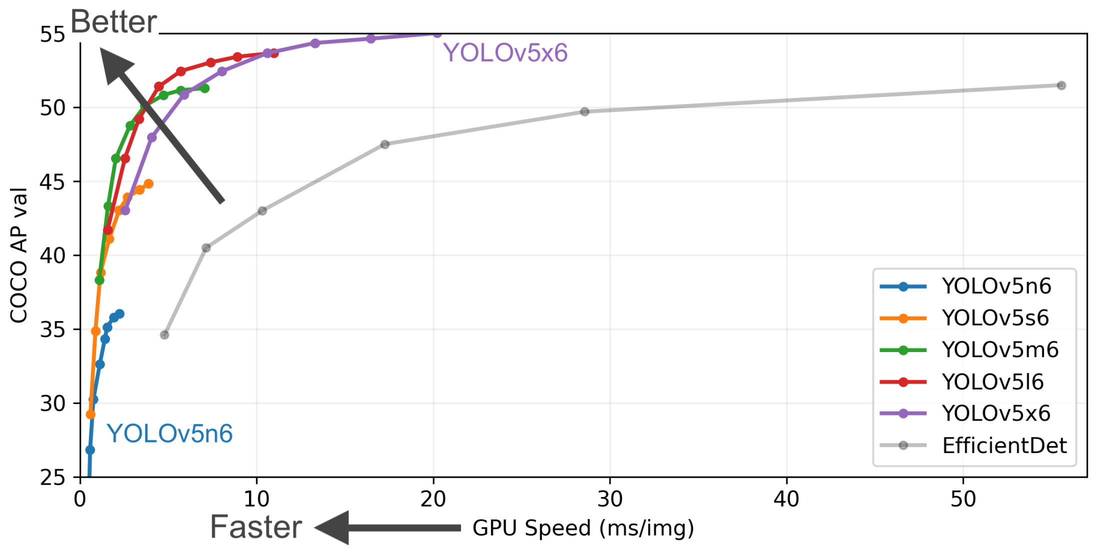


Figure 3.9: YOLOv5's benchmark [17].

3.2.5 YOLOv6

YOLOv6 was published in 2022 by Li. et al. building on previous versions of YOLO [18]. The results achieved by the model are presented in Figure 3.10.

The backbone architecture, named EfficientRep, takes RepVGG [43] as a starting element for the construction of small networks and uses CSPStackRep block for larger models. The neck follows the example of YOLOv4 and YOLOv5 using a PAN topology and enhancing it with the use of RepBlocks and CSPStackRep Blocks to achieve RepPAN while the use of decoupled head in the head is simplified to achieve Efficient Decoupled Head. Several changes have been introduced in this version such as:

- The loss function was changed by choosing VariFocal Loss and SIoU/GIoU as classification and regression loss functions.
- Several strategies were evaluated with regard to label assignment, and TAL was chosen for reasons of performance and simplicity.
- The use of the common practice of self-distillation for regression was introduced.

- The model was trained with the RepOptimiser to mitigate performance degradation in quantizing reparameterization based models.

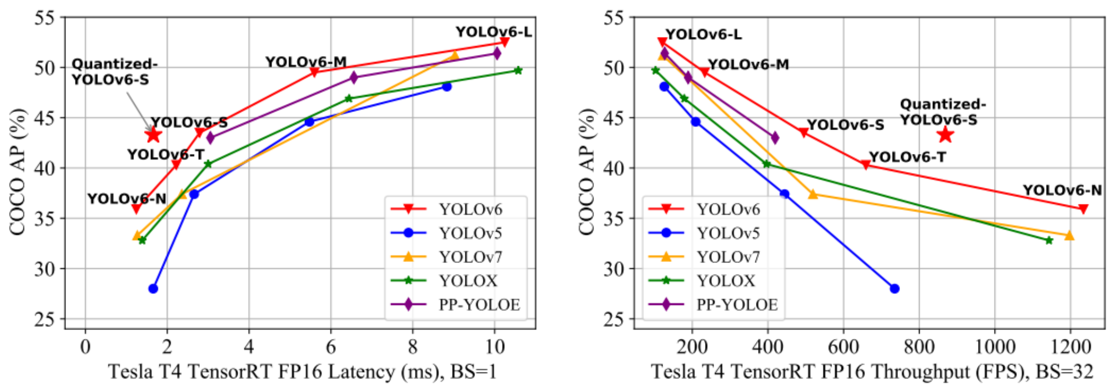


Figure 3.10: YOLOv6's benchmark [18].

3.3 YOLOv7

YOLOv7 was published on 6 July 2022 by Wang et al. and was chosen as the object detector to be used in this work because of its ease of use, its ability to benefit from the different models presented and its excellent performance in terms of speed and accuracy as shown in Figure 3.11 [19].

The problem of achieving an efficient network was analysed with regard to the architecture of YOLOv7. The choice made to solve this problem was to use the E-ELAN (Extended ELAN), which is a modified version of the ELAN (Efficient Long-Range Attention Network). E-ELAN is responsible for controlling the shortest longest gradient path to converge the network efficiently using different computational blocks. The overall architecture of YOLOv7 is depicted in Figure 3.12.

Another improvement added in this version is the introduction of model scaling for concatenation-based models. The objective of model scaling is to adjust certain model attributes in order to obtain models that work at different inference rates, but the application of this technique is problematic in the context of a concatenation-based architecture as it is necessary to analyse the different scaling factors together. The proposed solution is the compound scaling method, which makes it possible to maintain the properties of the model and its optimal structure at different scales.

In the YOLOv7 paper [19] the concept introduced in YOLOv4 [16] of bag-of-freebies is used again to describe techniques that improve model accuracy without affecting inference times but only training times.

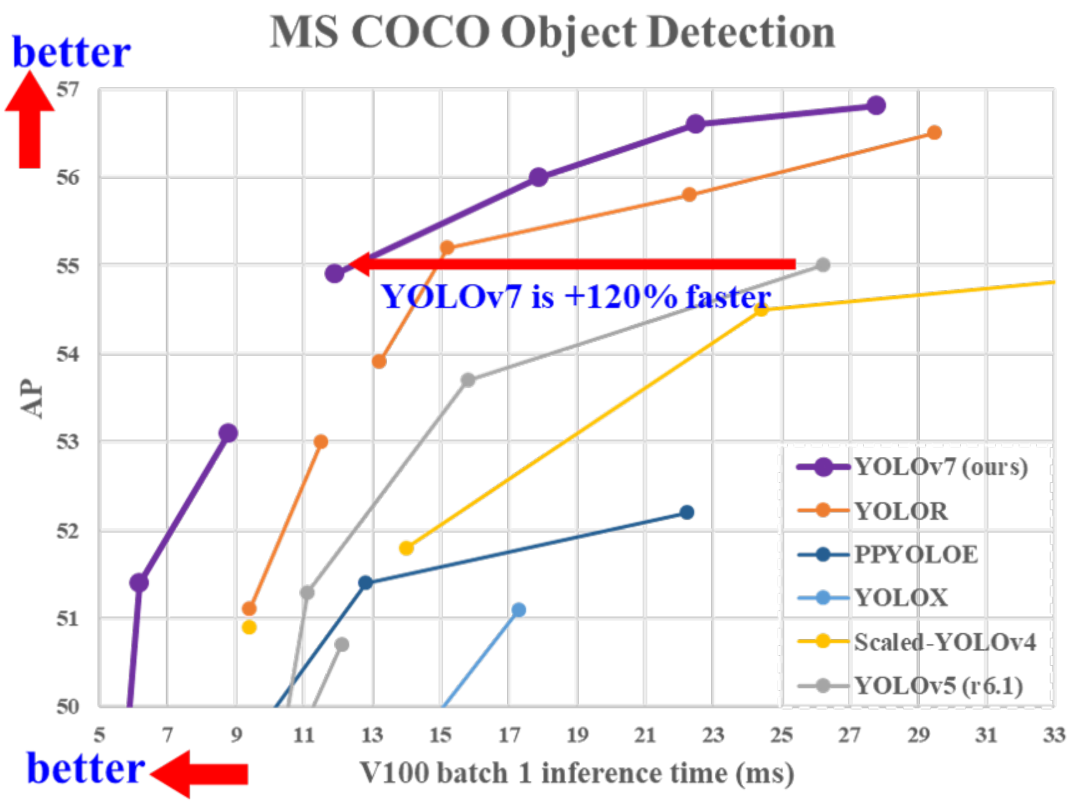


Figure 3.11: YOLOv7's benchmark [19].

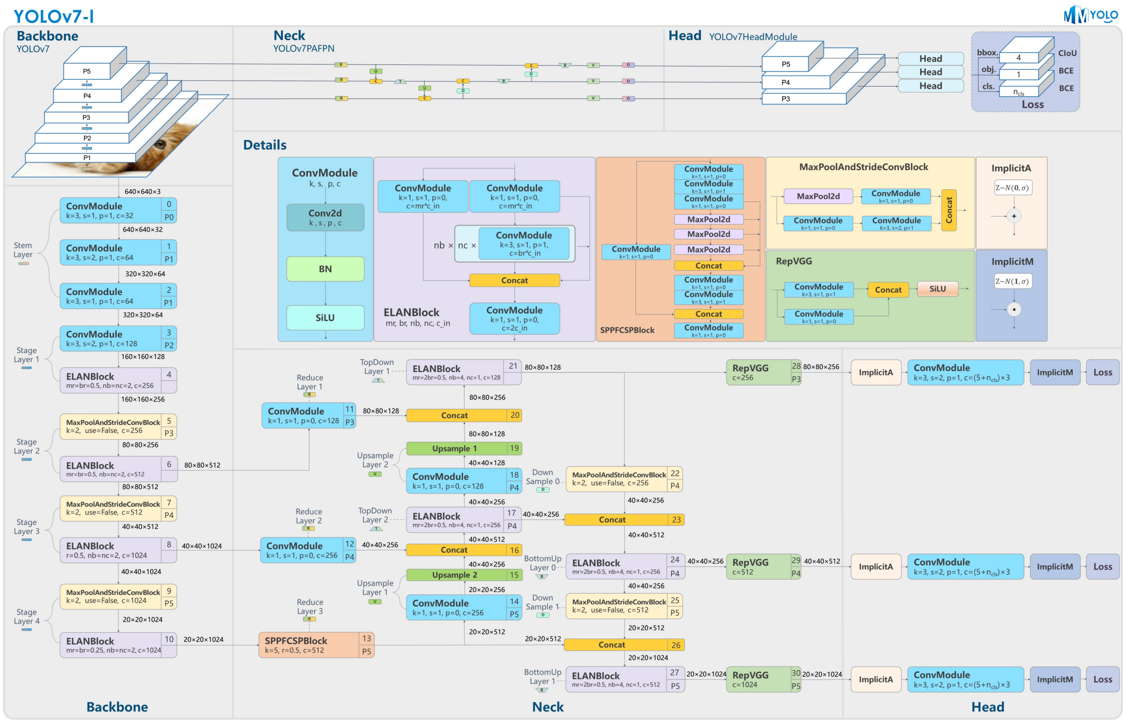


Figure 3.12: YOLOv7's architecture [20].

Among the various techniques, the use of gradient flow propagation paths was chosen to analyse how re-parameterized convolution can be combined with different networks. For re-parameterized convolution, RepConv was used without the identity connection (called RepConvN) as it does not fit a convolutional layer without residuals or concatenation.

Another concept addressed is that of deep supervision [44], which consists of the use of additional auxiliary heads in the central layers of the network to assist the actual head of the network (called the lead head). To manage a label assignment system that works for both types of heads, a new method was introduced.
The system consists of using lead head prediction to guide both auxiliary heads and lead heads since the lead head label assigner has great learning capabilities and the label assigned by it should be more representative of the connection between data and target. In this process two sets of labels are generated, the first is the one normally generated by the lead head label assigner while the second is obtained by relaxing the constraints to obtain coarser assignment. The relevance of the fine label and coarse label changes dynamically during the learning process so that the former is always more important.

Chapter 4

FMR-DB (Facial Masks and Respirators Database)

FMR-DB (Facial Masks and Respirators Database) [30] was created by Marceddu, Antonio Costantino and Montrucchio, Bartolomeo of the Politecnico di Torino with the aim of helping researchers around the world in tackling the advancing coronavirus pandemic by studying a system capable of recognising the type of mask or respirator worn by a person.

The contribution made to the development of the FMR-DB consists of the addition of images up to the current total number. The work also includes the labelling of the entire database according to the criteria set out in Section 4.4.

4.1 Images

The database consists of 4200 images collected on the Internet through the use of various search engines.

The database is first divided between images in which the subjects are wearing masks or respirators and images in which they are not in order to have a set against which to compare the labelled images. Within the images without masks or respirators, there are some representing people with their faces completely exposed and others where faces are partially covered by various common occlusions.
With regard to the section in which masks or respirators are present, the images were divided into six categories according to the type of object worn: surgical masks, non-medical masks, disposable respirators with a valve, disposable respirators without a valve, full-face respirators and half-face respirators are differentiated.

The overall database structure is depicted in the following list:

- With Facial Mask or Respirator.
  - Disposable Respirators With Valve.
  - Disposable Respirators Without Valve.
  - Full-Face Respirators.
  - Half-Face Respirators.
  - Non-Medical Masks.
  - Surgical Masks.
- Without Facial Mask or Respirator.
  - With Occlusions
    - Hands On Mouth.
    - Hats.
    - Neck Warmers And Bandanas.
    - Sunglasses.
  - Without Occlusions.

Further, more specific subdivisions are described in detail below.

4.2 Without Facial Mask or Respirator

There are 1,300 images where no masks or respirators are present. These include images representing people with their faces completely uncovered and others with their faces covered by the most common types of occlusions: sunglasses, hats, neck warmers and bandanas and hands over their mouths.
Each occlusion category counts 200 images, while the one without occlusion count 500 pictures.

4.3 With Facial Mask or Respirator

Facial masks and respirators are the focus of this thesis and the different subdivisions proposed in the database are therefore analysed in detail.
In total there are 2900 units, 500 each for the categories of surgical masks, non-medical masks, disposable respirators without a valve and disposable respirators with valve, and 450 for half and full face respirators.

4.3.1 Surgical Masks

Surgical masks (shown in Figure 4.1) are protective equipment often used by workers in the healthcare sector and act as a mechanical barrier for the nose and mouth. Their use dates back to the late 19th century and it has become practice to use them within healthcare facilities to prevent infections [45]. In some eastern countries, they are widely used by the population to combat seasonal allergies and air pollution [46].

Figure 4.1: Example image of a person wearing a surgical mask [21].

4.3.2 Non-Medical Masks

Non-medical masks (shown in Figure 4.2) are masks that do not meet safety standards and therefore cannot be used in healthcare. They are used at people's discretion in contexts where dust and other particulates have to be dealt with but have no technical specifications in production. They are often made of fabric or cotton since there is no specific material indicated for them.

4.3.3 Half-Face and Full-Face Respirators

Reusable respirators (shown in Figure 4.3 and 4.4) are personal protective equipment used in various fields and also called elastomeric respirators because they seal the face with elastomeric material. Half-face respirators are used in all contexts where eye protection is not required where full-face respirators are used instead.
They are characterised by the presence of either mechanical filters, where protection is achieved by separating unwanted matter from the inside of the respirator by

Figure 4.2: Example image of a person wearing a non-medical mask [22].

means of suitable materials through physical action, or chemical cartridges, where a substance (usually activated charcoal) is used to absorb harmful substances and prevent them from entering the respiratory tract.

Reusable respirators have been subdivided according to the protective equipment used if present, differentiating between eye and head protection for the half-face model and head only for the full-face model. The resulting structure for half-face respirators is shown below:

- Simple.
- With Eye And Head Protection.
- With Eye Protection.

- With Head Protection.

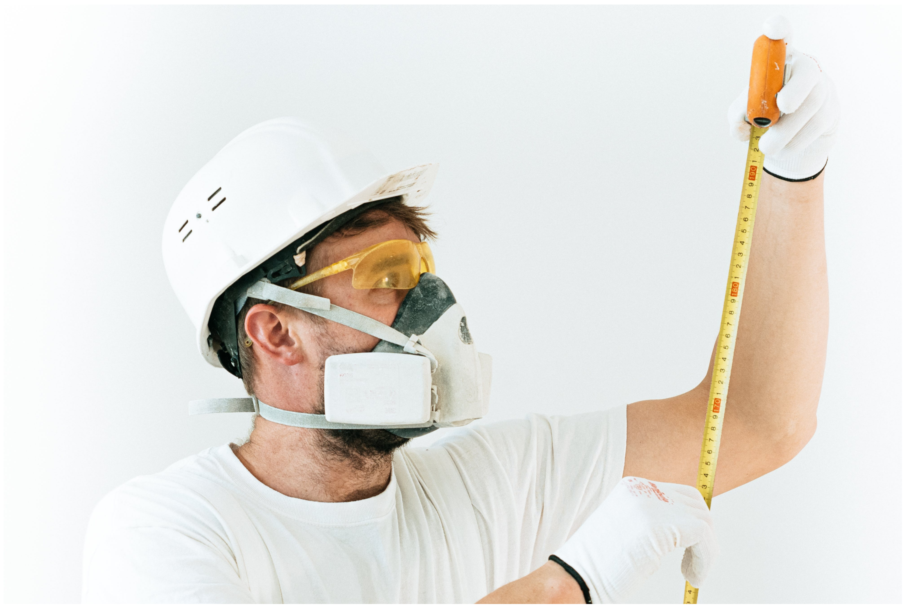


Figure 4.3: Example image of a person wearing a half-face respirator [23].

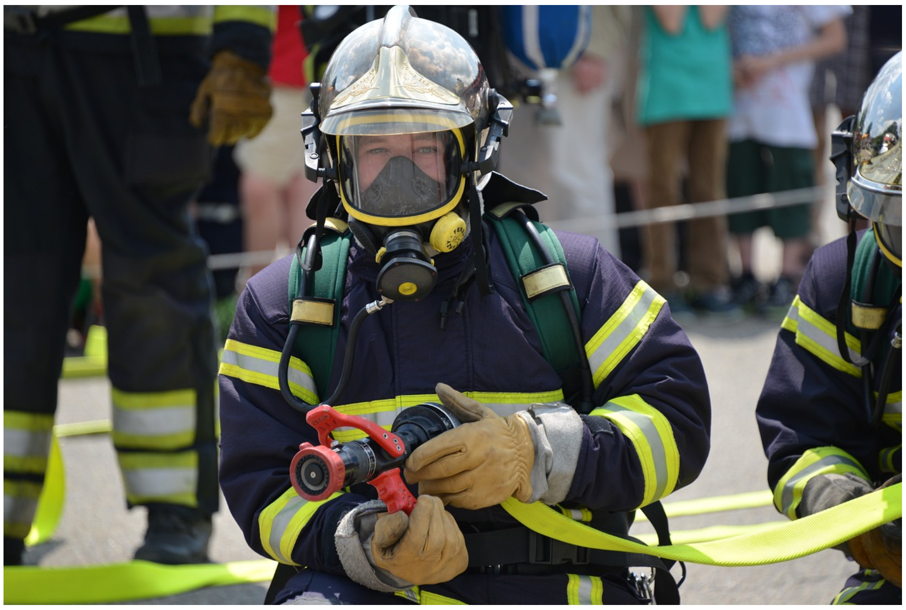


Figure 4.4: Example image of a person wearing a full-face respirator [24].

4.3.4 Disposable Respirators With and Without Valve

Reusable respirators are used in specific contexts and can be expensive, so disposable respirators (shown in Figure 4.5 and 4.6) are available on the market to solve other needs. Some respirators use filtered valves to facilitate breathing during use. Disposable Respirators are used in many contexts, ranging from health care to manufacturing, and are therefore made of different materials depending on the type of external agent to be insulated. In the database, images of disposable respirators were sorted according to two criteria: the presence or absence of eye and/or head protection equipment and, where possible, according to the type of protection provided (FFP1, FFP2, FFP3 or other unknown). The resulting internal division of the database is the same for both classes and has the following format

- Simple.
- With Eye And Head Protection.
  - FFP1.
  - FFP2.
  - FFP3.
  - Other - Unknown.
- With Eye Protection.
  - FFP1.
  - FFP2.
  - FFP3.
  - Other - Unknown.
- With Head Protection.
  - FFP1.
  - FFP2.
  - FFP3.
  - Other - Unknown.

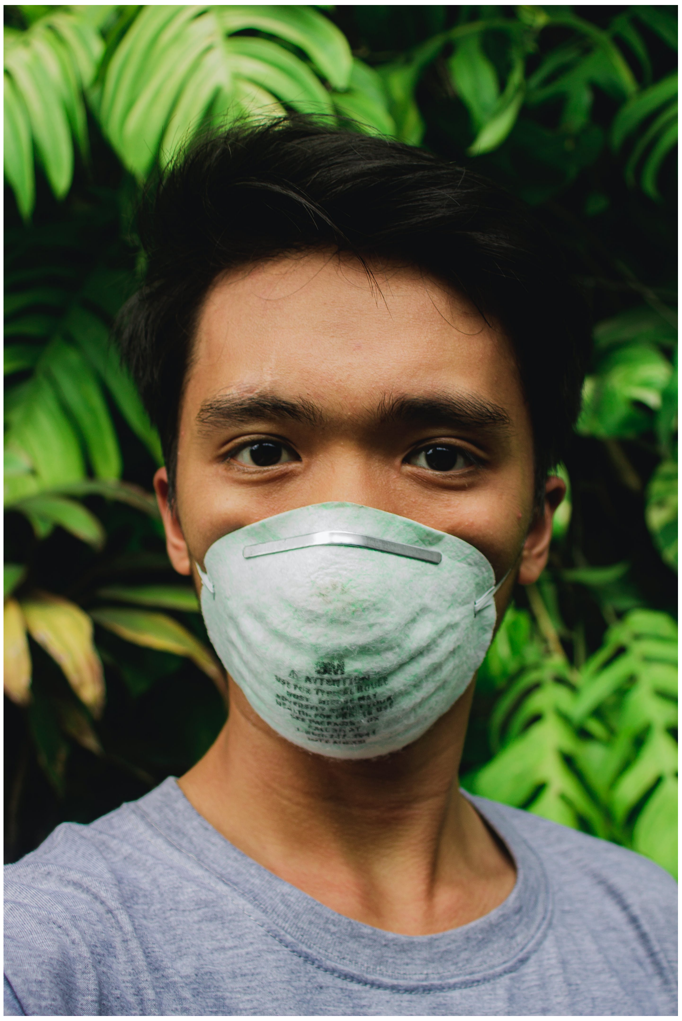


Figure 4.5: Example image of a person wearing a Disposable Respirator Without Valve [25].

4.4 Labels

Labelling of the database was done with LabelImg, a graphical image annotation software written in Python and supporting PASCAL VOC, YOLO and CreateML formats [27]. The program interface is shown in Figure 4.7.

The different classes labelled in the database are:

- Person.
- Disposable Respirator With Valve.
- Disposable Respirator Without Valve.

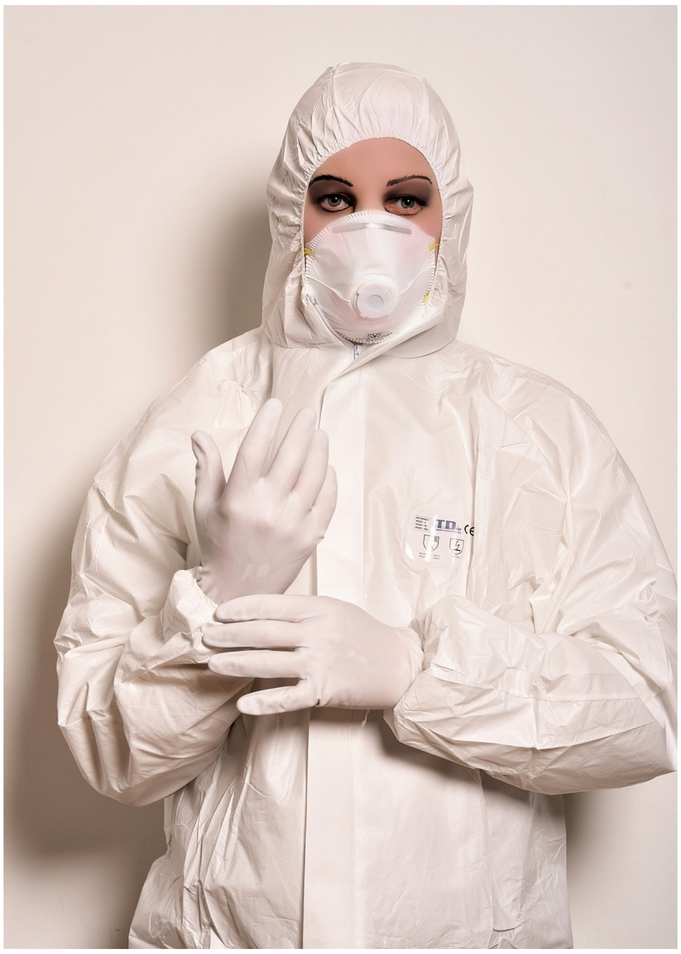
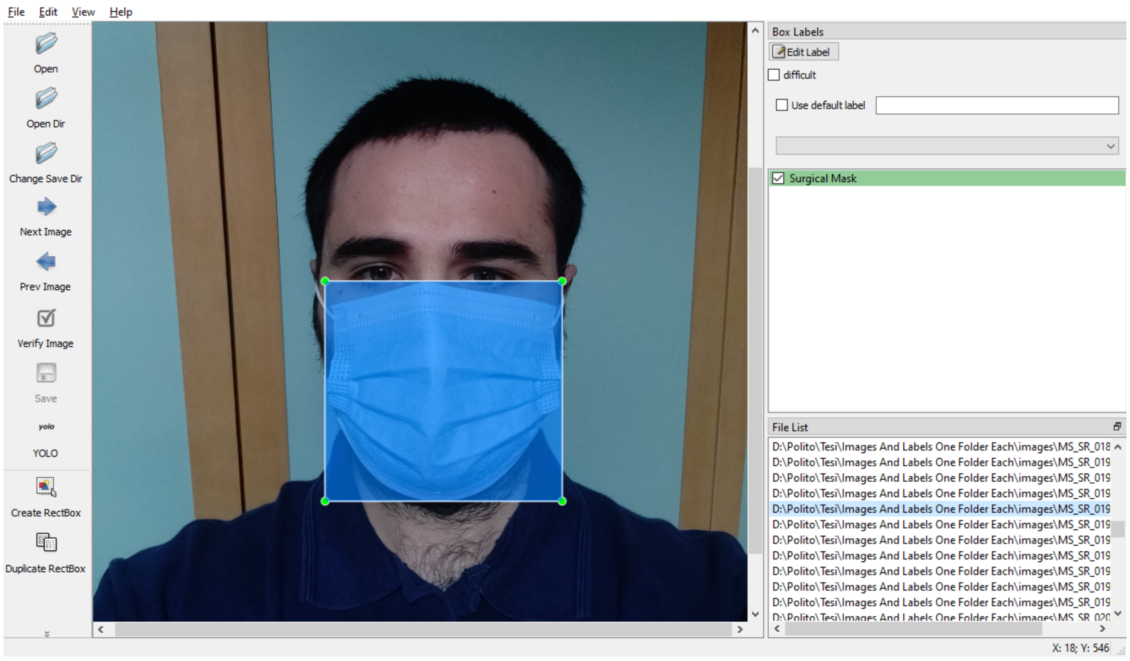


Figure 4.6: Example image of a person wearing a Disposable Respirator With Valve [26].

- Non-Medical Mask.
- Surgical Mask.
- Eye Protection.
- Head Protection.
- Hands On Mouth.
- Hats.
- Neck Warmers And Bandanas.
- Sunglasses.
- Person with Smaller Box.



Surgical Mask

```
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_018
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_019
D:\Polito\Tesi\Images And Labels One Folder Each\images\MS_SR_020
```

Figure 4.7: Screenshot example of LabelImg [27].

- Half Face Respirator.
- Full Face Respirator.

The database consists of 14 labels, 6 of which explicitly indicate the type of mask or respirator worn, 6 represent the occlusions in the images and 2 labels are dedicated to recognising the people within the images.

The labels for the recognition of people are different in that the Person class considers the top of the head, the edges of the ears and the chin as the boundaries of the bounding box; the Person with Smaller Box class is more representative of current facial recognition models and considers vertically the face from the temples to the hollow between the lower lip and chin and fits horizontally by forming a square with a side equal to the first measurement.

The presence of occlusions was taken into account for the different possible application scenarios of the database: depending on the context, the person under analysis might or might not use the proposed elements, and considering the additional labels, a more efficient model could be built for the scope of use.

In the specific case of YOLO, the labels are 'txt' files where each line corresponds to a label. For each label, 5 data items are specified:

- The class to which the object belongs.
- The coordinates of the centre of the label relative to the dimensions of the image normalised between 0 and 1 (in order x and y).

- The dimensions of the label relative to the dimensions of the image normalised between 0 and 1 (in order of width and height).

The visualisation of a YOLO label is shown in Figure 4.8.

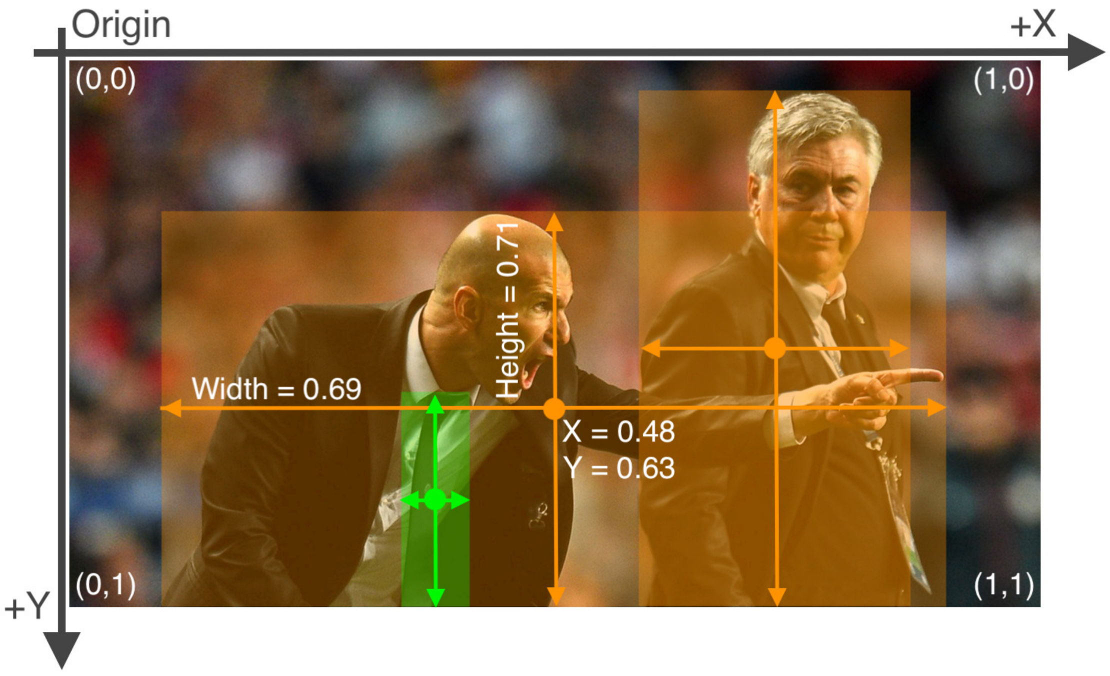


Figure 4.8: YOLO label [28].

Chapter 5

Results

This chapter analyzes the results of several trainings done on the FMR-DB using YOLOv7 in order to obtain a robust and efficient model for the recognition of face masks or respirators within images.

5.1 Before the Training

The database was divided into three parts for training purposes by assigning 80% of the images to the training set, 10% to the validation set and 10% to the test set as shown in Figure 5.1.
YOLOv7 needs a well-defined division of data between images and labels into different folders and subfolders grouping data intended for training and data intended for validation.

To divide the database into its parts, a program was written in Python which, taking as input the folder containing the images and the folder containing the labels, would randomly split the images and their respective labels into three folders called 'train', 'val' and 'test' according to the proportion 80-10-10. The program code 5.1 represents the source code of this program.

```python
import os
import random
import shutil


def split():

    count = 0
    img_arr = os.listdir("images")
    random.shuffle(img_arr)
    length = len(img_arr)
```

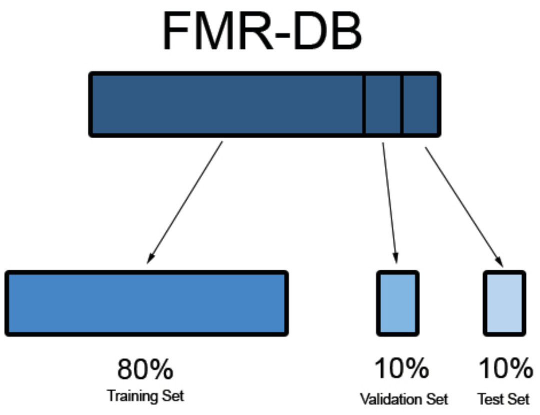


Figure 5.1: Database was splitted in 80% for the training set, 10% for the validation set and 10% for the test set.

```
train = "train"
val = "val"
test = "test"
images_path = os.path.join(os.getcwd(), "images")
label_path = os.path.join(os.getcwd(), "labels")
train_path = os.path.join(images_path, train)
val_path = os.path.join(images_path, val)
test_path = os.path.join(images_path, test)

train_label_path = os.path.join(label_path, train)
val_label_path = os.path.join(label_path, val)
test_label_path = os.path.join(label_path, test)

os.mkdir(train_path)
os.mkdir(val_path)
os.mkdir(test_path)

os.mkdir(train_label_path)
os.mkdir(val_label_path)
os.mkdir(test_label_path)

for i in img_arr:
    base = i.split(".")[0]
    label_address = base+".txt"
    img_path_old = os.path.join(images_path, i)
    img_label_old = os.path.join(label_path, label_address)
    if count < int(length*0.8):
        shutil.move(img_path_old, train_path)
        shutil.move(img_label_old, train_label_path)
        img_path = os.path.join(train_path, i)
        with open("train.txt", "a") as x:
            x.write(img_path+"\n")
    elif count < int(length*0.9):
        shutil.move(img_path_old, val_path)
        shutil.move(img_label_old, val_label_path)
        img_path = os.path.join(val_path, i)
        with open("valid.txt", "a") as x:
            x.write(img_path + "\n")
    else:
        shutil.move(img_path_old, test_path)
        shutil.move(img_label_old, test_label_path)
```

```
            img_path = os.path.join(test_path, i)
            with open("test.txt", "a") as x:
                x.write(img_path + "\n")
        count = count+1
```

Listing 5.1: The program written to divide the database into training, validation, and test sets.

The training and testing phase was carried out on a desktop computer having the following components:

- CPU: AMD Ryzen 7 2700X.
- GPU: NVIDIA GeForce RTX 2060 GAMING Z 6 GB GDDR6.
- RAM: Corsair Vengeance LPX 16 GB DDR4, 3000 MHz.
- Motherboard: ASUS ROG STRIX B450-F GAMING.
- 500 GB NVMe M.2 SSD + 1 TB HDD.

The network training was performed on a desktop PC and due to hardware issues it was necessary to reduce the number of images being analysed and classes to be trained.
The purpose of the training is to create a model capable of recognising the type of face mask or respirator worn by a person in a photo or video and therefore the selection of data was made with this objective in mind. Images without face masks or respirators were ignored to reduce the amount of data to be processed in the model and only classes containing the objects under analysis were selected:

- Disposable Respirators With Valve.
- Disposable Respirators Without Valve.
- Full-Face Respirators.
- Half-Face Respirators.
- Non-Medical Masks.
- Surgical Masks.

This selection reduced the number of images in the dataset to 2900 and the number of classes under consideration to 6. The split dataset thus obtained this distribution:

- 2320 photos for training.
- 290 photos for validation.
- 290 photos for testing.

5.2 Training Settings

Network training was done by fine tuning using YOLOv7. The YOLOv7 guide on the official repository on GitHub [47] was followed to set up the environment required for fine tuning the network. The basic training parameters were obtained by adapting the standard system settings to the possibilities of the hardware.

- –workers: 4
  Number of subprocesses to be parallelised .
- –device: 0
  Set the use of the GPU for training.
- –batch-size: 16
  The number of samples crossing the network at a time.
- –img: 640 640
  Size to which images are scaled.
- –cfg: yolov7-tiny.yaml
  The architecture of the model used for training.
- –weights: yolov7_training.pt
  Starting weights used as a basis for fine tuning.
- –hyp: hyp.scratch.tiny.yaml
  The hyperparameters used.
- –epochs
  The number of epochs for which the network is trained.

5.2.1 Training Parameters

The number of workers and the batch-size were selected as the highest power of 2 allowed by the computer in use: higher numbers were tried but led to instability or fatal errors in training execution.
The device selected is the NVIDIA GeForce RTX 2060 GPU as it has better performance than the CPU for the calculations required.
The number of epochs was an important number on which several experimental tests were carried out to find the best quantity that took into account a compromise between accuracy and overfitting.
The weights of the standard version of YOLOv7 were used as a starting point for fine-tuning as they are indicated as an excellent compromise between training time and accuracy.
An attempt was made to use the basic architecture of YOLOv7 but this resulted in excessively long training times and instability during execution. Instead, it was decided to use YOLOv7-tiny, a lighter architecture designed for edge-computers that still delivers excellent mAP results.

5.2.2 Training Hyperparameters

The starting hyperparameters are those indicated for YOLOv7-tiny and experimental tests were carried out on them to find the optimal configuration. In Section 5.3 the term basic hyperparameters is used to refer to the following parameters and it is made explicit in which trainings there are changes with respect to them:

1. lr0: 0.01
2. lrf: 0.01
3. momentum: 0.937
4. weight_decay: 0.0005
5. warmup_epochs: 3.0
6. warmup_momentum: 0.8
7. warmup_bias_lr: 0.1
8. box: 0.05
9. cls: 0.5
10. cls_pw: 1.0

11. obj: 1.0
12. obj_pw: 1.0
13. iou_t: 0.20
14. anchor_t: 4.0
15. fl_gamma: 0.0
16. hsv_h: 0.015
17. hsv_s: 0.7
18. hsv_v: 0.4
19. degrees: 0.0
20. translate: 0.1
21. scale: 0.5
22. shear: 0.0
23. perspective: 0.0
24. flipud: 0.0
25. fliplr: 0.5
26. mosaic: 1.0
27. mixup: 0.05
28. copy_paste: 0.0
29. paste_in: 0
30. loss_ota: 0

The most interesting hyper-parameters were selected for experimentation and are described below.
Hyperparameters 1 and 2 represent the learning rate to be assigned to the first epoch and the learning rate to be assigned to the last epoch, respectively. The learning rates of the intermediate epochs are derived by linearly dividing the interval and increasing the value each time.

Hyperparameters 16 to 27 define the weights of certain data augmentation techniques implemented by YOLO such as rotation, translation or more complex ones such as mosaic and mixup. The response of the model to changes in the learning rate and in the weights of certain data augmentation techniques,such as rotation (degrees), scaling (scale), and mosaic, are examined.

5.2.3 Other Notes

For reasons of conciseness acronyms are used to indicate the model classes within the tables according to the following coding:

- Surgical mask : SR.
- Non medical mask: NM.
- Full face respirator: FF.
- Half face respirator: HF.
- Disposable respirator without valve: NV.
- Disposable respirator with valve : WV.

5.3 Results

5.3.1 First Training

The first training was done by maintaining the basic parameters and hyperparameters with a low number of 20 epochs to test the functioning of YOLOv7. The results are not great but this is normal considering the small number of epochs for which the training lasted. Despite the small number of epochs, the model manages to obtain minimal results for the most distinct classes while it struggles to differentiate Disposable Respirators with and without valve as they are very similar.

The model achieves a mAP@.5 of 0.8313 and a mAP@.5:.95 of 0.5948 and its accuracy is not equally distributed among the different classes as shown in Figure 5.2.

5.3.2 Experimental Training

The first parameter to be improved is the number of epochs, so it was decided to do a series of tests with a progressive increase in the number of epochs to find the best value that does not lead to overfitting or deterioration of results.

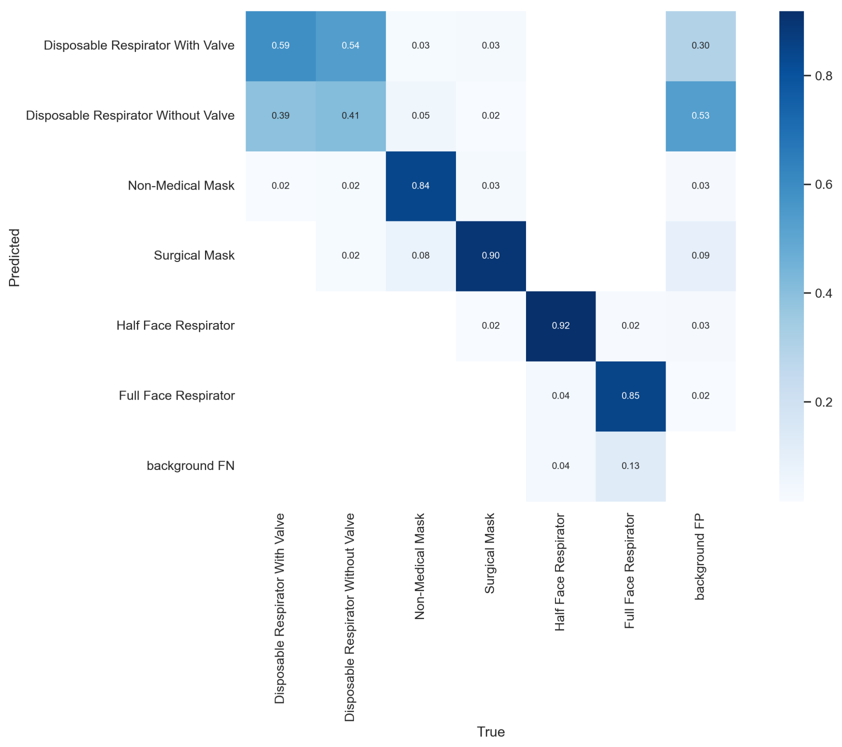


Figure 5.2: Confusion Matrix obtained with 20 epochs.

It has been tried to increase the number of epochs to 50, 75, 100, 125 and 150 with increasing results in terms of mAP@.5 and mAP@.5:.95 as shown in Table 5.1. These metrics reflect the general performance of the model but not the specific performance of the different classes under analysis.

epochs	**mAP@.5**	**mAP@.5:.95**
20	0,8313	0,5948
50	0,9424	0,7861
75	0,9836	0,847
100	0.9867	0.8701
125	**0.9908**	0.8838
150	0.9868	**0.8883**

Table 5.1: Training results with different epoch numbers.

Analysing the results specific to the classes, it was noted that some classes accuracy deteriorated as the number of epochs increased as seen in Table 5.2, and it was decided to select the number of epochs that allowed for a higher average value of accuracy.

epochs	**SR**	**NM**	**FF**	**HF**	**NV**	**WV**	**Mean**
20	0.9	0.84	0.85	0.92	0.41	0.59	0.7517
50	0.98	0.89	0.98	0.94	0.62	0.8	0.8683
75	0.98	0.94	0.98	0.98	0.85	0.98	0.9517
100	0.98	0.94	0.98	0.96	0.93	0.96	0.9583
125	1	0.97	0.98	0.98	0.89	0.96	**0.9633**
150	0.98	0.92	0.98	0.98	0.89	0.98	0.955

Table 5.2: Class specific accuracy for different epoch numbers.

Although the training with 150 epochs has the highest mAP@.5:.95 value, it should be noted that the training with 125 epochs has a slightly lower value, obtains a higher mAP@.5, and has the highest average accuracy value.

It was therefore decided to experiment with an intermediate value between 100 and 125 epochs in order to seek an improvement in the results using 112 epochs.

Comparing the accuracy and mAP data, it can be observed that 112 and 125 have similar values in terms of performance as shown in Table 5.3 and Table 5.4 but it was decided to continue the experiment using 112 epochs as a reference as it seems to provide a more balanced distribution between the different classes.

epochs	**mAP@.5**	**mAP@.5:.95**
100	0.9867	0.8701
112	0.9885	0.8745
125	**0.9908**	**0.8838**

Table 5.3: Training results to search for the number of epochs more precisely.

The next point of analysis was to experiment with the different hyperparameters in order to search for a configuration capable of providing better results. The hyperparameters were changed independently of each other in order to check the weight they have on the result.
The different cases under analysis are addressed in the Tables 5.5 and 5.6.

epochs	**SR**	**NM**	**FF**	**HF**	**NV**	**WV**	**Mean**
100	0.98	0.94	0.98	0.96	0.93	0.96	0.9583
112	0.97	1	0.98	0.98	0.95	0.95	**0.9717**
125	1	0.97	0.98	0.98	0.89	0.96	0.9633

Table 5.4: Class specific accuracy to search for the number of epochs more precisely.

lr0	**lrf**	**mos**	**deg**	**scale**	**mAP@.5**	**mAP@.5:.95**
0.01	0.01	0	0	0.5	0.6627	0.4249
0.01	0.01	1	30	0.5	0.9869	0.669
0.005	0.01	1		0.50	0.9873	0.8616
0.02	0.01	1	0	0.5	0.9876	**0.8803**
0.01	0.01	1	0	0.5	**0.9885**	0.8745
0.01	0.01	1	10	0.5	0.9821	0.715
0.01	0.01	1	0	0.3	0.9832	0.8555
0.01	0.02	1	0	0.5	0.9847	0.8681

Table 5.5: Training with 112 Epochs.

With regard to mosaic, there is a marked deterioration in results when absent, probably due to the need to greatly increase the number of epochs required for effective training. Rotation is another parameter whose change has led to a worsening of the model, with regard to both test values. The change in the scaling parameter was taken into account as a test value and again there is no improvement in the results compared to the base case.
The initial and final learning rates are changed in these tests: the highest value of mAP@.5:.95 is achieved at an initial learning rate of 0.02, while as far as the other metrics are concerned, the basic hyperparameters are confirmed to have the best performance (initial learning rate 0.01).

5.3.3 Final Model

Ultimately, the model obtained using 112 epochs with the following hyperparameters was chosen as the best one:

lr0	**lrf**	**mos**	**deg**	**scale**	**SR**	**NM**	**FF**	**HF**	**NV**	**WV**	**Mean**
0.01	0.01	0	0	0.5	0.56	0.44	0.17	0.66	0.22	0.59	0.44
0.01	0.01	1	30	0.5	0.98	0.92	0.98	0.98	0.81	0.96	0.9383
0.005	0.01	1	0	0.5	0.98	0.89	0.98	0.98	0.93	0.95	0.9517
0.02	0.01	1	0	0.5	1	0.95	0.96	0.98	0.93	0.95	0.9617
0.01	0.01	1	0	0.5	0.97	1	0.98	0.98	0.95	0.95	**0.9717**
0.01	0.01	1	10	0.5	0.98	0.89	0.98	0.98	0.82	0.98	0.9383
0.01	0.01	1	0	0.3	1	0.97	0.96	0.98	0.91	0.95	0.9616
0.01	0.02	1	0	0.5	0.98	0.97	0.98	0.98	0.87	0.98	0.96

Table 5.6: Hyperparameters changes while training with 112 Epochs.

- lr0 = 0.01.
- lrf = 0.01.
- mosaic = 1.
- degrees = 0.
- scale = 0.5.

This model is able to achieve a mAP@.5 of 0.9885 and a mAP@.5:.95 of 0.8745 while maintaining an average accuracy value of 0.9717.

In Figure 5.3, it is possible to observe the progress of the network during training while Figure 5.4, 5.5, 5.6 and 5.7 show the precision, recall, PR and F1-score curves.

Finally, the Figure 5.8 shows the confusion matrix of the best trained model. Distinguishing disposable respirators with or without a valve gives the worst accuracy score as the two classes are extremely similar and differentiated only by the presence of the valve. This flaw is present but to a negligible extent as the performance of the model is still excellent for each class represented.
The column containing the false positive backgrounds is of little significance as some of the images in the database contain unlabelled masks or respirators in the background and therefore the model can correctly recognise these items but they are shown in this category although the classification is correct.

The model produced during training can be used to do inference on new data using YOLOv7 features as shown in Figure 5.9 and 5.10.

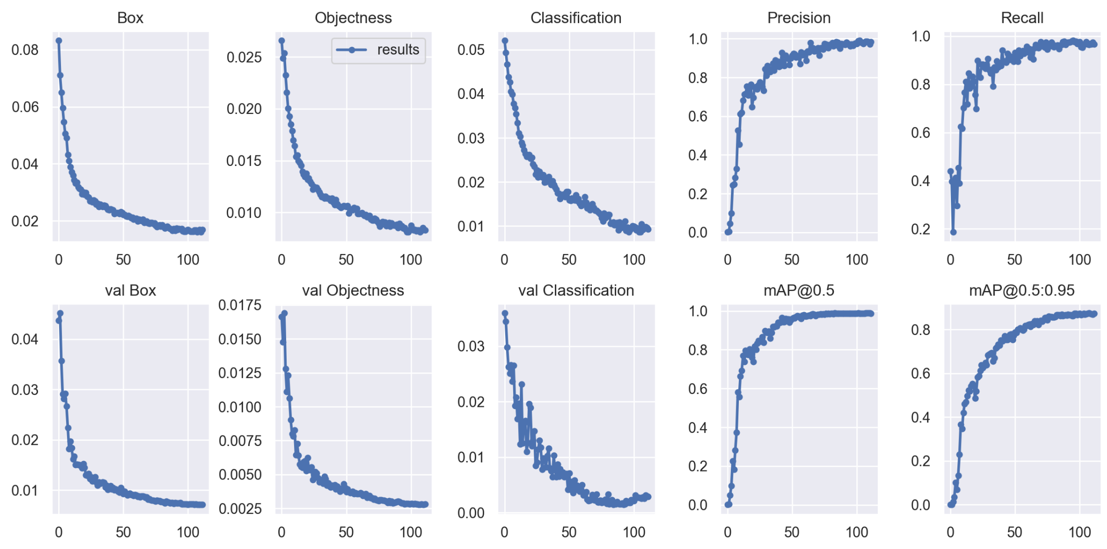


Figure 5.3: Training trend graphs at 112 epochs.

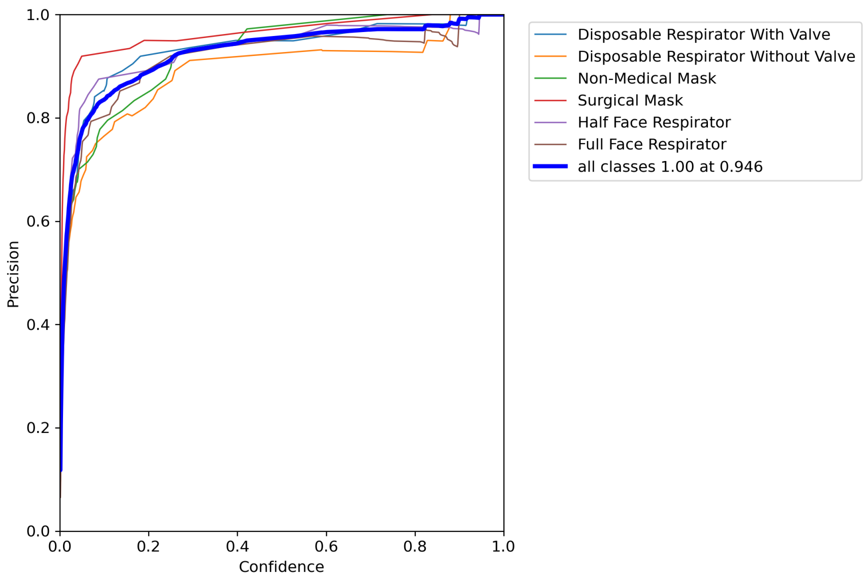


Figure 5.4: Precision graph.

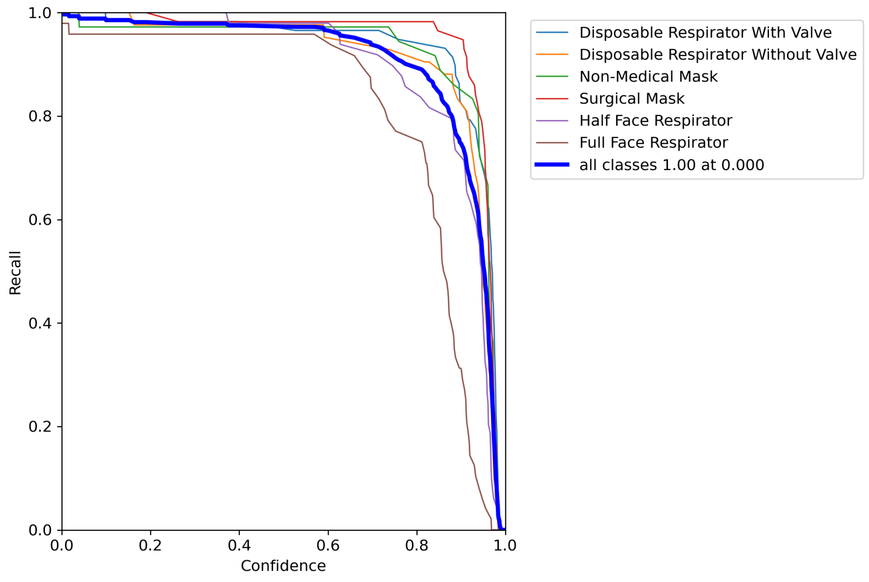


Figure 5.5: Recall graph.

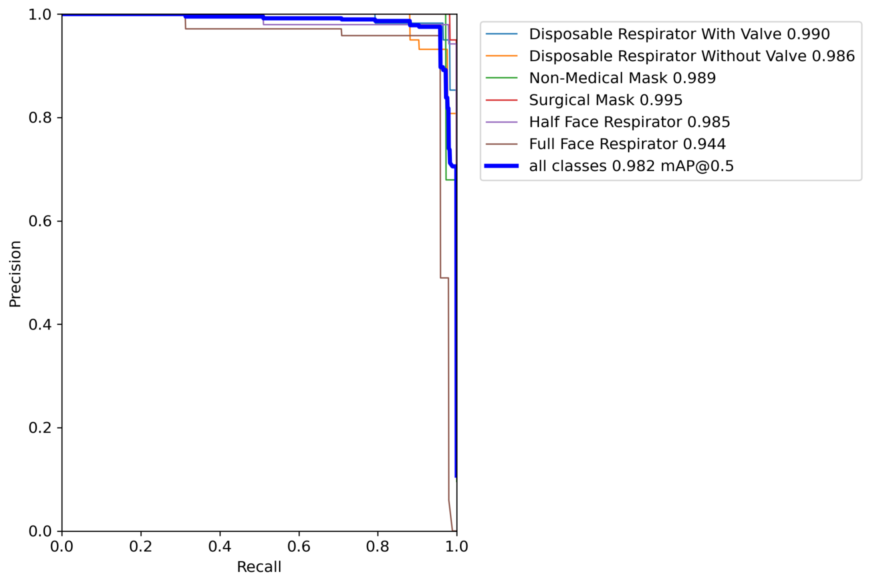


Figure 5.6: PR-curve graph.

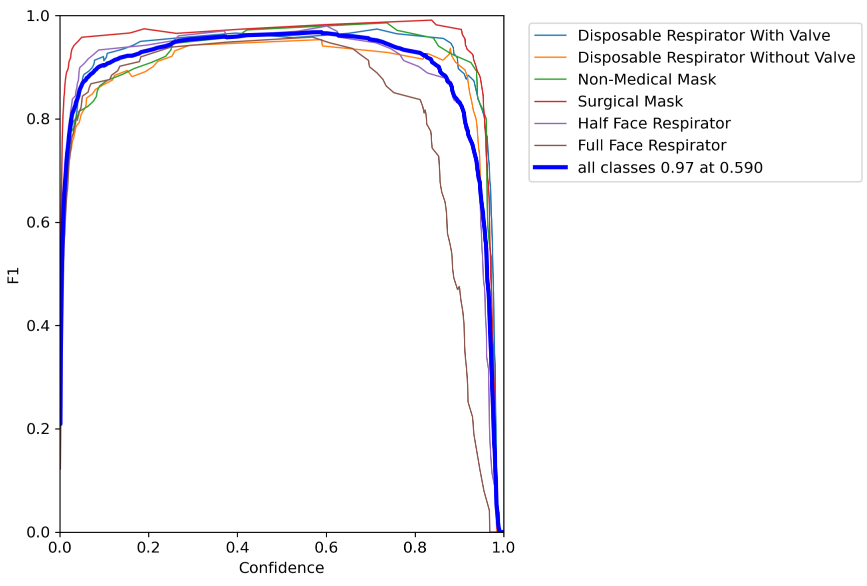


Figure 5.7: F1-score graph.

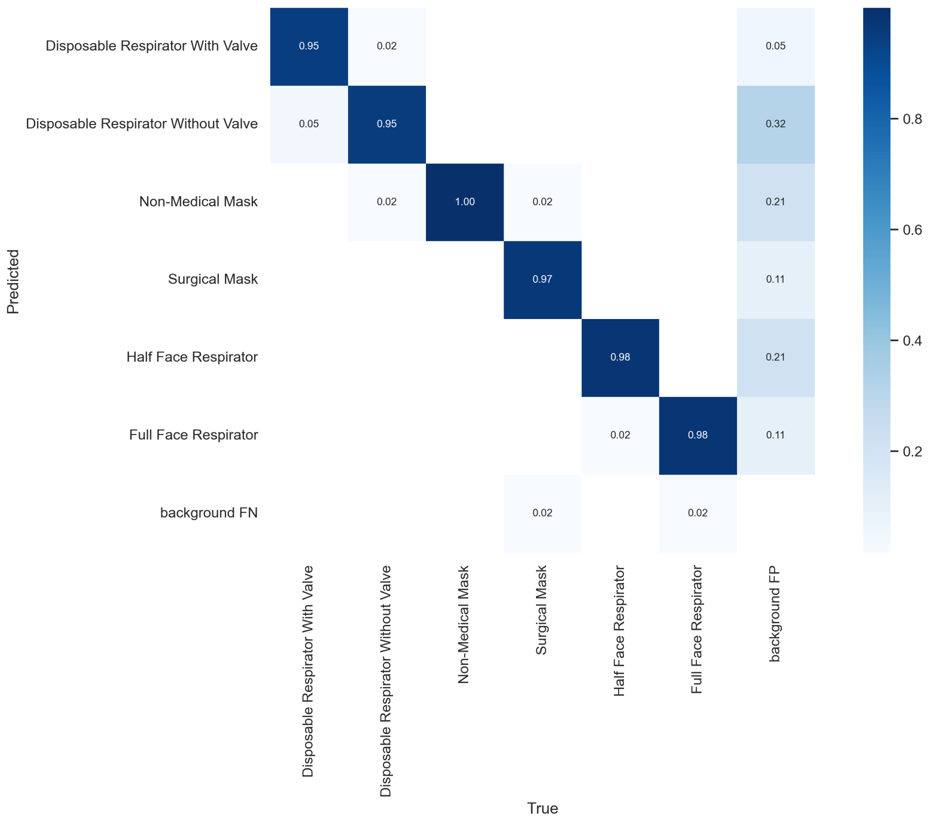


Figure 5.8: Confusion Matrix of the best model.

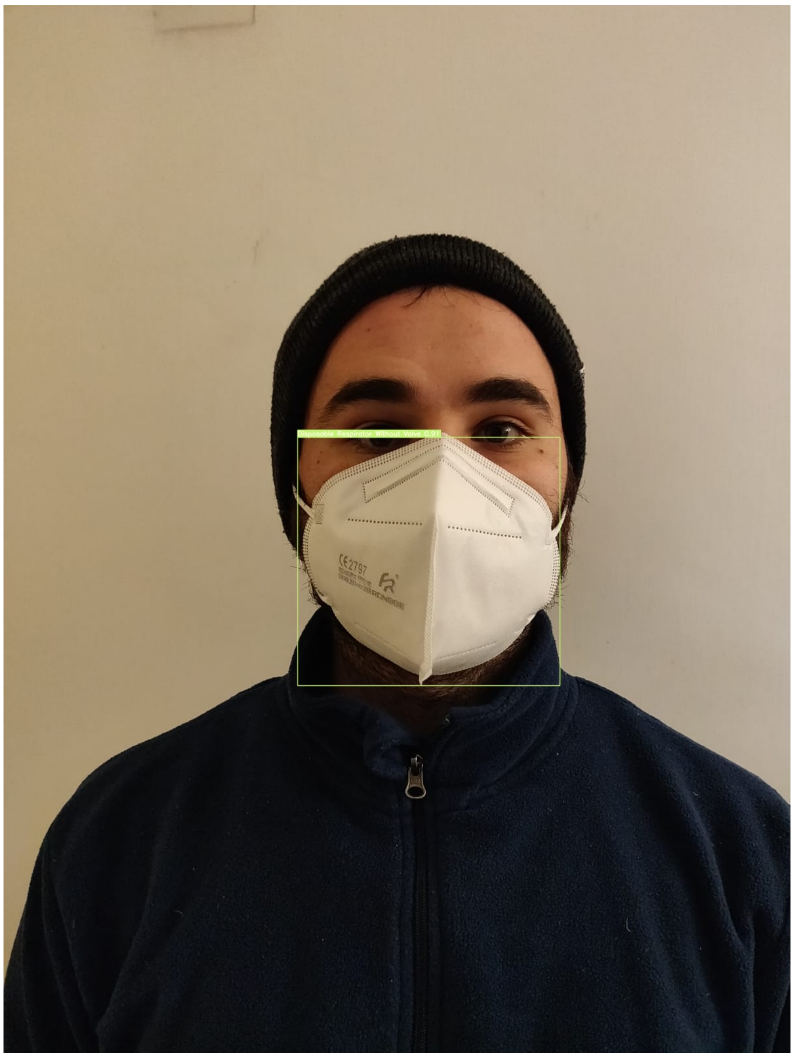


Figure 5.9: Inference made on disposable respirator without valve.

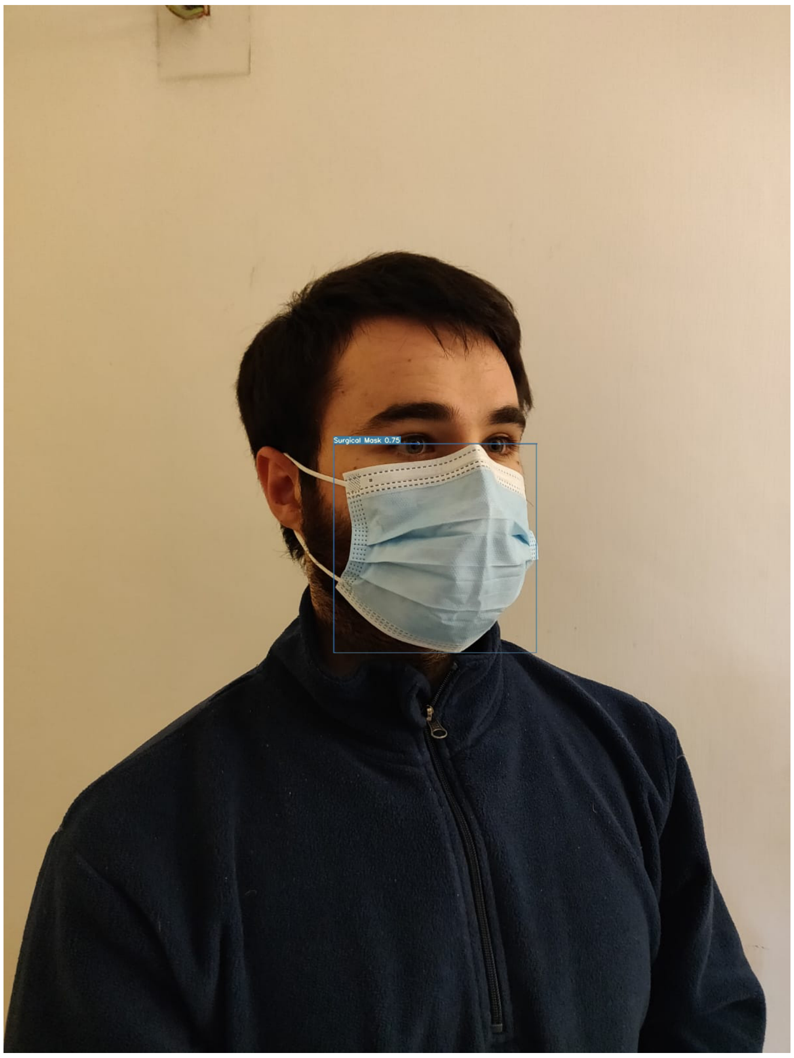


Figure 5.10: Inference made on surgical mask.

Chapter 6

Conclusions

6.1 Results

In this thesis, the FMR-DB (Facial Masks and Respirators Database) was expanded by adding images to the dataset collected by Marceddu, Antonio Costantino and Montrucchio, Bartolomeo.

The database now consists of 4200 images representing people wearing 6 different types of face masks or respirators and people with uncovered faces to provide a term of comparison. The types of face masks or respirators under analysis are:

- Disposable Respirators With Valve.
- Disposable Respirators Without Valve.
- Full-Face Respirators.
- Half-Face Respirators.
- Non-Medical Masks.
- Surgical Masks.

Possible occlusions within the image such as eye or head protection equipment were taken into account in order to provide an enlarged case study that can be used for different purposes.

In addition, a state-of-the-art neural network was trained using YOLOv7 through a process of experimentation to find the hyperparameters and settings that would yield the best results in terms of accuracy and performance.

The model developed manages to achieve excellent performance in all classes represented, allowing it to be used in various fields.

6.2 Future Work

Since the quality of the models produced by neural networks depends mainly on the input data, it would be possible to continue the work done by expanding the database. In particular, it would be interesting to deepen the subdivision of the categories by further differentiating the various types of face masks or respirators.

In addition, it would be possible to collect images in order to cover a series of more specific environmental circumstances such as lighting conditions and extreme camera angles.

Regarding the training of the neural network, it would be useful to create new models using other state-of-the-art systems in order to find the most effective one for the FMR-DB.

In the specific case of YOLOv7, an attempt could be made to train the network by considering other different combinations of parameters and hyperparameters and to make use of more complicated models through the use of higher performance machines.

In conclusion, this paper provides a database and an object detector model based on it: the natural continuation would be to use the model produced to develop an application capable of being used in practical working conditions.

Bibliography

[1] harkiran78. Artificial neural networks and its applications, 2023. URL `https://media.geeksforgeeks.org/wp-content/uploads/20230410104038/Artificial-Neural-Networks.webp`. [Online]. Accessed on Nov. 28, 2023.

[2] harkiran78. Artificial neural networks and its applications, 2023. URL `https://media.geeksforgeeks.org/wp-content/cdn-uploads/20230602113310/Neural-Networks-Architecture.png`. [Online]. Accessed on Nov. 28, 2023.

[3] PyTorch Foundation. Sigmoid, 2023. URL `https://pytorch.org/docs/stable/_images/Sigmoid.png`. [Online]. Accessed on Nov. 28, 2023.

[4] PyTorch Foundation. Tanh, 2023. URL `https://pytorch.org/docs/stable/_images/Tanh.png`. [Online]. Accessed on Nov. 28, 2023.

[5] PyTorch Foundation. ReLU, 2023. URL `https://pytorch.org/docs/stable/_images/ReLU.png`. [Online]. Accessed on Nov. 28, 2023.

[6] PyTorch Foundation. LeakyReLU, 2023. URL `https://pytorch.org/docs/stable/_images/LeakyReLU.png`. [Online]. Accessed on Nov. 28, 2023.

[7] PyTorch Foundation. SiLU, 2023. URL `https://pytorch.org/docs/stable/_images/SiLU.png`. [Online]. Accessed on Nov. 28, 2023.

[8] Jeremy Alix. Convolution integral: Simple definition, 2022. URL `https://www.statisticshowto.com/wp-content/uploads/2020/10/convolution-of-functions-300x180.png`. [Online]. Accessed on Nov. 28, 2023.

[9] Brilliant.org. Convolutional neural network, 2023. URL `https://brilliant.org/wiki/convolutional-neural-network`. [Online]. Accessed on Nov. 28, 2023.

[10] Anuj shah. Through the eyes of gabor filter, 2023. URL `https://miro.medium.com/v2/resize:fit:640/format:webp/1*-KeFDswepXGWtr5MORnjng.png`. [Online]. Accessed on Nov. 28, 2023.

[11] TagX. Data augmentation for computer vision, 2022. URL `https://medium.com/@tagxdata/data-augmentation-for-computer-vision-9c9ed474291e`. [Online]. Accessed on Nov. 28, 2023.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91.

[13] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *CVPR 2017*, pages 6517–6525, 07 2017. doi: 10.1109/CVPR.2017.690.

[14] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL `http://arxiv.org/abs/1804.02767`.

[15] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. URL `http://arxiv.org/abs/1708.02002`.

[16] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. URL `https://arxiv.org/abs/2004.10934`.

[17] Glenn Jocher. YOLOv5 by Ultralytics, May 2020. URL `https://github.com/ultralytics/yolov5`. [Online]. Accessed on Nov. 28, 2023.

[18] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[19] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[20] OpenMMLab. Yolov7, 2022. URL `https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov7`. [Online]. Accessed on Nov. 28, 2023.

[21] Anna Shvets. Woman wearing surgical mask, 2020. URL https://www.pexels.com/it-it/foto/donna-strada-fuori-medico-3902881/. [Online]. Accessed on Nov. 28, 2023.

[22] Maksim Goncharenok. Woman wearing non medical mask, 2020. URL https://www.pexels.com/it-it/foto/donna-ritratto-in-posa-maschera-viso-4750169/. [Online]. Accessed on Nov. 28, 2023.

[23] Antoni Shkraba. Man wearing half-face respirator, 2020. URL https://www.pexels.com/it-it/foto/uomo-persona-mano-tenendo-4981787/. [Online]. Accessed on Nov. 28, 2023.

[24] blickpixel. Fireman wearing full-face respirator, 2014. URL https://pixabay.com/it/photos/vigili-del-fuoco-515776/. [Online]. Accessed on Nov. 28, 2023.

[25] Nothing Ahead. Uomo che indossa la maschera per il viso, 2020. URL https://www.pexels.com/it-it/foto/uomo-che-indossa-la-maschera-per-il-viso-3571628/. [Online]. Accessed on Nov. 28, 2023.

[26] leo2014. Woman wearing disposable respirator with valve, 2020. URL https://pixabay.com/it/photos/surgeon-gloves-latex-gloves-doctor-4785104/. [Online]. Accessed on Nov. 28, 2023.

[27] Tzutalin. Labelimg, 2015. URL https://github.com/tzutalin/labelImg. [Online]. Accessed on Nov. 28, 2023.

[28] glenn jocher. Object detection datasets overview, 2023. URL https://docs.ultralytics.com/datasets/detect/. [Online]. Accessed on Nov. 28, 2023.

[29] Worldometers. Covid-19 coronavirus pandemic, 2023. URL https://www.worldometers.info/coronavirus/. [Online]. Accessed on Nov. 28, 2023.

[30] Antonio Costantino Marceddu and Bartolomeo Montrucchio. Recognizing the type of mask or respirator worn through a cnn trained with a novel database. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1490–1495, 2021. doi: 10.1109/COMPSAC51774.2021.00221.

[31] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

[32] Dulari Bhatt, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. Cnn variants for computer

vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20), 2021. ISSN 2079-9292. doi: 10.3390/electronics10202470. URL `https://www.mdpi.com/2079-9292/10/20/2470`.

[33] Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, page 1237–1242. AAAI Press, 2011. ISBN 9781577355144.

[34] Abid Ali Awan. A complete guide to data augmentation, 2022. URL `https://www.datacamp.com/tutorial/complete-guide-data-augmentation`. [Online]. Accessed on Nov. 28, 2023.

[35] Deval Shah. Mean average precision (map) explained: Everything you need to know, 2022. URL `https://www.v7labs.com/blog/mean-average-precision`. [Online]. Accessed on Nov. 28, 2023.

[36] Yali Nie, P. Sommella, Mattias O'Nils, Consolatina Liguori, and Jan Lundgren. Automatic detection of melanoma with yolo deep convolutional neural networks. In *IEEE International Conference on e-Health and Bioengineering EHB 2019*, 11 2019. doi: 10.1109/EHB47216.2019.8970033.

[37] Hanan Ashraf, Muhammad Imran, Abdulrahman Qahtani, Abdulmajeed Alsufyani, Omar Almutiry, Awais Mahmood, Muhammad Khan, and Mohamed Habib. Weapons detection for security and video surveillance using cnn and yolo-v5s. *Computers, Materials and Continua*, 70:2761–2775, 01 2022. doi: 10.32604/cmc.2022.018785.

[38] Nurin Mirza Afiqah Andrie Dazlee, Syamimi Abdul Khalil, Shuzlina Abdul-Rahman, and Sofianita Mutalib. Object detection for autonomous vehicles with sensor-based technology using yolo. *International Journal of Intelligent Systems and Applications in Engineering*, 10(1):129–134, Mar. 2022. doi: 10.18201/ijisae.2022.276. URL `https://ijisae.org/index.php/IJISAE/article/view/1365`.

[39] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond, 2023.

[40] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. URL `http://arxiv.org/abs/1612.03144`.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014. URL `http://arxiv.org/abs/1406.4729`.

[42] Zheng Ma, Ming Li, and Yu Guang Wang. PAN: path integral based convolution for deep graph neural networks. *CoRR*, abs/1904.10996, 2019. URL http://arxiv.org/abs/1904.10996.

[43] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. *CoRR*, abs/2101.03697, 2021. URL https://arxiv.org/abs/2101.03697.

[44] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets, 2014.

[45] Bruno J Strasser and Thomas Schlich. A history of the medical mask and the rise of throwaway culture. *The Lancet*, 2020. URL https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)31207-1/fulltext.

[46] Jeff Yang. A quick history of why asians wear surgical masks in public, 2014. URL https://qz.com/299003/a-quick-history-of-why-asians-wear-surgical-masks-in-public. [Online]. Accessed on Nov. 28, 2023.

[47] Wong Kin-Yiu. Yolov7, 2022. URL https://github.com/WongKinYiu/yolov7/tree/main#acknowledgements. [Online]. Accessed on Nov. 28, 2023.