UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

POSTGRADUATE PRORAMME

“INFORMATION SYSTEMS AND SERVICES”

MASTER THESIS

“Deep Learning and Object detection”

Theodoros Papageorgiou

Supervisor: Michael Filippakis

Piraeus, 2022

Contents

I dedicate this thesis to Ioanna.

Acknowledgments

I would like to acknowledge and thank the supervisor of my thesis, Associate Professor Michael Filippakis, for his trust and Ph.D. Maria Eleni Poulou for her valuable assistance during this work.

Finally, I must express my warmest thanks to my beloved partner Ioanna and my sister Karolina for her continuous support and love over the years.

Abstract

The rise of COVID-19 disease came up with new restrictions in everyday life, like quarantine, social distancing, wearing face masks and more. In the current diploma thesis object detection algorithms are applied on image and video data in order to detect if a person wears or not a face mask as a preventative measure against the spread of COVID-19 disease. More specifically, in this dissertation a comparative study was conducted between one-stage detectors YOLOv3, YOLOv4 and YOLOv5 on an RGB image dataset, which consists of 1694 images and 7791 labels. The performance of YOLOv5 model is the best one between the three models.

Περίληψη

Η εξάπλωση της νόσου COVID-19 είχε ως αποτέλεσμα την επιβολή μέτρων προστασίας στην καθημερινή ζωή, όπως η καραντίνα, η κοινωνική απομάκρυνση, η χρήση μάσκας προστασίας και άλλα. Στην παρούσα διπλωματική εργασία εφαρμόστηκαν αλγόριθμοι ανίχνευσης αντικειμένου σε δεδομένα εικόνων και βίντεο με σκοπό να εντοπίσουν εάν ένα άτομο φοράει ή όχι μάσκα προστασίας σαν προληπτικό μέτρο κατά της εξάπλωσης της νόσου COVID-19. Συγκεκριμένα, στην παρούσα εργασία διεξάχθηκε συγκριτική μελέτη μεταξύ των ανιχνευτών ενός επιπέδου YOLOv3, YOLOv4 και YOLOv5 σε ένα σύνολο δεδομένων εικόνων RGB, το οποίο αποτελείται από 1694 εικόνες και 7791 ετικέτες. Την καλύτερη επίδοση μεταξύ των τριών μοντέλων είχε το μοντέλο YOLOv5.

Abbreviations

AI	Artificial Intelligence
NN	Neural Network
ANN	Artificial Neural Network
FFNN	Feed Forward Neural Networks
FBNN	Feedback Neural Networks
ReLU	Rectified Linear Unit
DL	Deep Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
mAP	Mean average precision
RoI	Regions of Interest
YOLO	You Only Look Once
DNN	Deep Neural Network
tp	true positive
tn	true negative
fp	false positive
fn	false negative

Figures

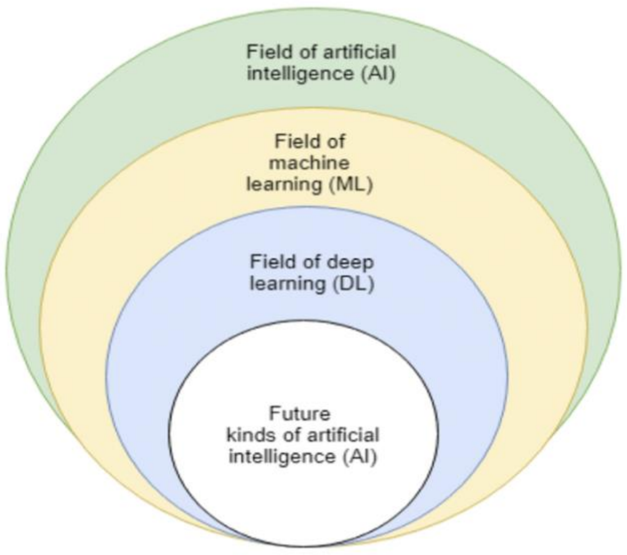
Tables

1. Artificial Intelligence

Artificial Intelligence (AI) consists of a huge number of subfields, and it is part of Computer Science. AI enables the creation of intelligent machines which can behave and think like humans as well as make decisions independently. The term AI consists of the word artificial and denotes something that is a human creation, as well as the word intelligence which refers to the ability of thought.

The human factor is crucial in AI, because the language of the "machine" consists of 0 and 1. Therefore, man must create a suitable communication environment between him and the "machine" to produce correct results. Also, AI is based on the analysis of provided data, the intelligence that the algorithm will acquire is proportional to the amount of data it is asked to analyze (Kumar & Choudhary, n.d.).

Figure 1. Representation of AI and its subfields (Abiodun et al., 2018).


Field of artificial intelligence (AI)
Field of machine learning (ML)
Field of deep learning (DL)
Future kinds of artificial intelligence (AI)

1.1. Machine Learning

Machine Learning (ML) is an AI application that enables a system to learn automatically as well as improve its ability to execute tasks though experience gained by repetition and the amount of information. ML also relies on developing software programs that are able to access and use data for learning purposes, and its ultimate goal is to allow computers automatically learn and make decisions without human intervention or assistance.

There are four main categories into which ML is divided: supervised, unsupervised, semi-supervised and reinforcement learning. They are separated based on the learning algorithm.

Supervised learning. The characteristic of supervised learning is that the instances of the training dataset are labeled. Commonly the labels are class labels in classification problems. The goal of this type of learning is to generate a function that maps inputs to desired outputs and therefore to induce models which can be used to classify other unlabeled data (Cunningham et al., 2008).

Unsupervised learning. Unsupervised learning is mainly used for clustering and feature reduction. In this type of learning the inputs are unlabeled and the goal is to find hidden patterns in the data (Batta, 2020; Usama et al., 2019).

Semi-supervised learning. This type of learning is a combination of supervised and unsupervised learning and it uses a mix of labeled and unlabeled inputs (Batta, 2020).

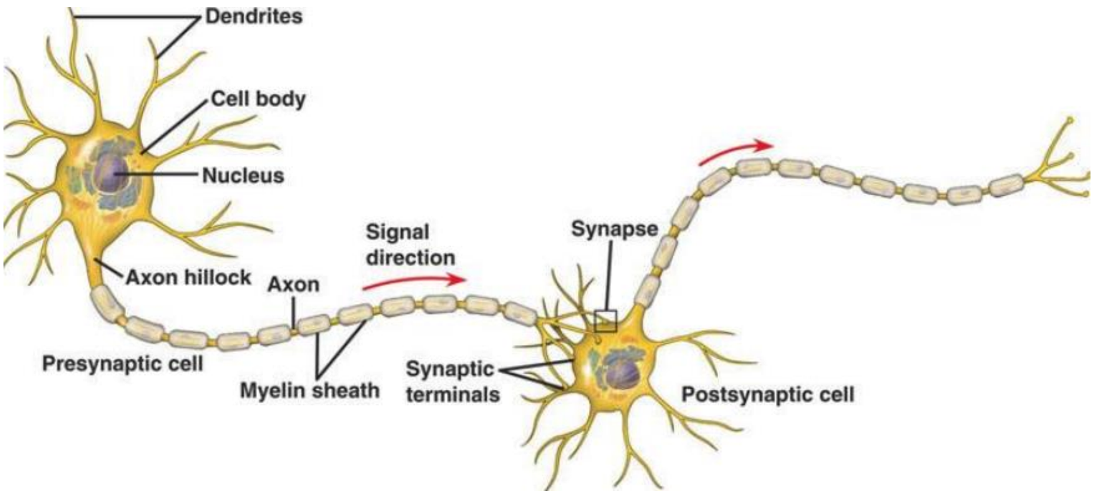
Reinforcement learning. In reinforcement learning, the algorithm receives feedback from the environment only after choosing an output for every data observation. It is typically used in sequential decision making problems (Simeone, 2018).

1.2. Biological and Artificial Neural Networks

1.2.1. Biological Neurons

The human brain consists of a large number of neurons and each neuron is linked to many other neurons via synapses forming very complex neural networks. Specifically, a neuron sends messages via synapses to another neurons. Each neuron consists of the dendrites, which receive electrical chemical energy from last neurons as inputs of a neuron, the axon hillock, which transfers electrical chemical energy into membrane potential, the axon, which transfers the membrane potential into an action potential and the synapse, which transfers the action potential into electrical and chemical energy and conduct them to the next neuron when the action potential is equal or greater than a certain threshold. Humans have the ability to self-learning because neural networks have a large number of parallel computing and distributed information processing. Therefore, scientists hope to apply new computer-based programs to utilize the self-learning ability of humans to emulate the human brain (Lin, 2017).

Figure 2. Schematic representation of a biological neuron (Lin, 2017).



1.2.2. Neural networks

In supervised learning, the scope of a Neural Network (NN) is to create a function that maps the input χ to output y. The relation of x and y can be represented as $y = f(x)$. Usually, a NN is implemented by using layers in which all neurons in a layer are connected to every neuron in the following layer, hence the network is called a Fully Connected Neural Network (FCNN).

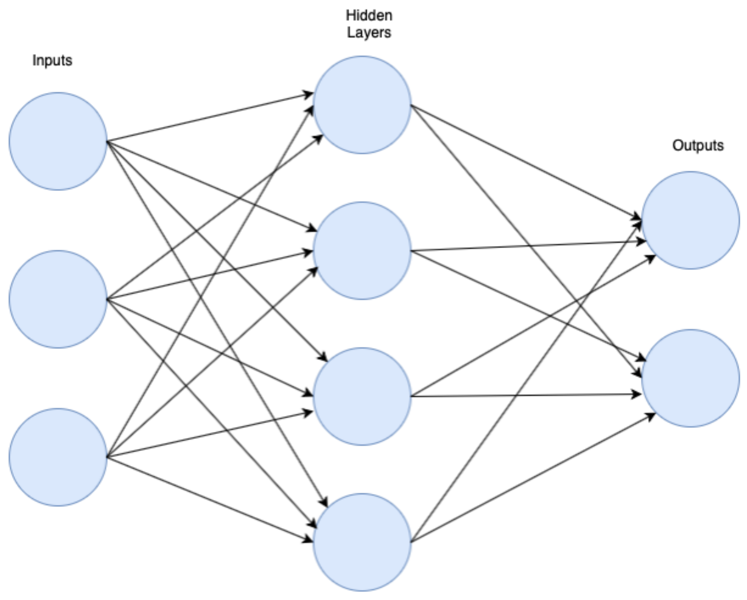
1.2.2.1. Artificial Neural Networks

Artificial Neural Networks (ANNs) are computer systems that are inspired by the neurons of the human brain. An ANN is a model based on a collection of connected nodes, the "artificial neurons", which roughly represent the neurons in the brain. The approach attributed to the human brain lies in the fact that the network receives knowledge from the environment through a learning process, and that the power between the connections of neurons is called synaptic weight where the acquired knowledge is stored. The accuracy of an ANN depends on the number of neurons and the type of activation function used. There is still no rule for the number of layers that an ANN must include in order to perform at its maximum or for the activation function, the only limitation is that the ANN must include at least two layers (Sharma et al., 2020).

The neurons of an ANN are organized in different levels of parallel arrangement. These levels are organized as follow:

- **Input layer.** This is the first level of an ANN. At this level the data enters the ANN and the number of variables is equal to the neurons.
- **Hidden layer.** The number of hidden layers may differ, and the number of neurons increases as the hidden levels increase.
- **Output layer.** This is the last level of an ANN. At this level, the results are the outputs, and the number of neurons is equal to the possible output variables.

Figure 3. Schematic representation of a multilevel ANN.



The three main features of an ANN model are the following:

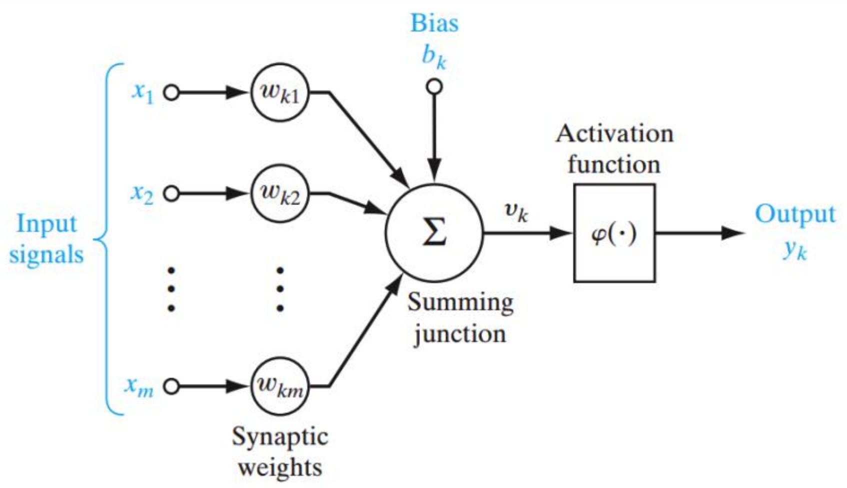
1 A set of synapses, each one has a specific weight. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. The values of the weights can be either positives or negatives.

2 An adder for summing all the input signals, weighted by the corresponding synaptic strengths of the neuron.

3 An activation function, which limits the amplitude of the output of a neuron. The normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1] or [-1,1].

(Simon, 1992)

Figure 4. Nonlinear model of a neuron *k*.



The neural network shown in Figure 4 indicates b_k bias. Bias increases or decreases the input value of the activation function, depending if the value is positive or negative, respectively.

In mathematical terms a neuron k can be described along these lines:

$$\mu_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$\Upsilon_k = \varphi(\mu_k + b_k)$$

Where $x_1, x_2, x_3, \dots, x_m$ are the signal inputs, the $w_{k1}, w_{k2}, w_{k3}, \dots, w_{km}$ are the corresponding synaptic weights of the neuron k, μ_k is the output of the linear combination of the input signals, b_k is the bias and $\varphi(\cdot)$ is the activation function of the neuron. Finally, the y_k is the output signal (Ferreira et al., 2019).

ANNs are categorized into two major categories, Feed Forward Neural Networks (FFNNs) and Feedback Neural Networks (FBNNs). In FFNN, the signal is transmitted from input nodes to the output nodes, but the reverse process is not allowed. Also, FFNNs are divided into single-tier and multi-tier networks, this separation results from the number of hidden layers. A network in which inputs are directly connected to the outputs is called a Single-Layer Neural Network or Perceptron Network. FBNNs contain at least one feedback loop, so there is at least one level of neurons that feeds the outgoing signal back to the inputs of all other neurons (Abiodun et al., 2018).

1.2.2.1.1. Deep Neural Networks

DNNs are powerful ML tools and they are commonly used to solve large-scale real-word problems, including automated image classification, natural language processing and human action recognition tasks (Samek et al., 2017). They are ANNs with multiple layers between the input and the output layers and they are able to localize objects in images (Bengio, 2009). They have the ability to learn more complex models and can achieve object detection without the need of hand designing features on the images (Hadidi et al., 2014).

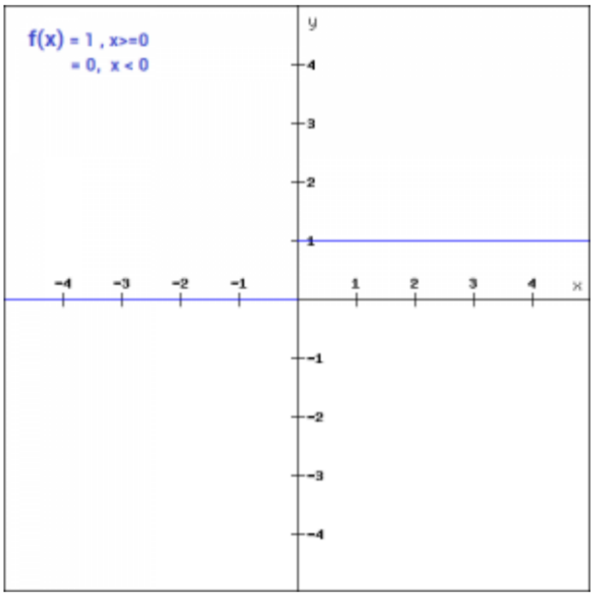
1.2.2.2. Activation functions

Activation functions are used in ANNs to convert the input signal into an output signal which in turn is fed as input to the next layer. It plays an important role on the accuracy of the ANN prediction thus its selection must be careful. If an ANN does not have an activation function, the output signal would be a linear function and the network acts as a Linear Regression

Model. The result of this is a network with limited performance. Some of the most important activations functions are Binary Step Function, Linear, Sigmoid, Tanh, ReLU, Leaky ReLU, Parametrized ReLU, Exponential Linear Unit, Swish and SoftMax.

1.2.2.3. Binary step function

Figure 5. Representation of binary step function (Sharma et al., 2020).

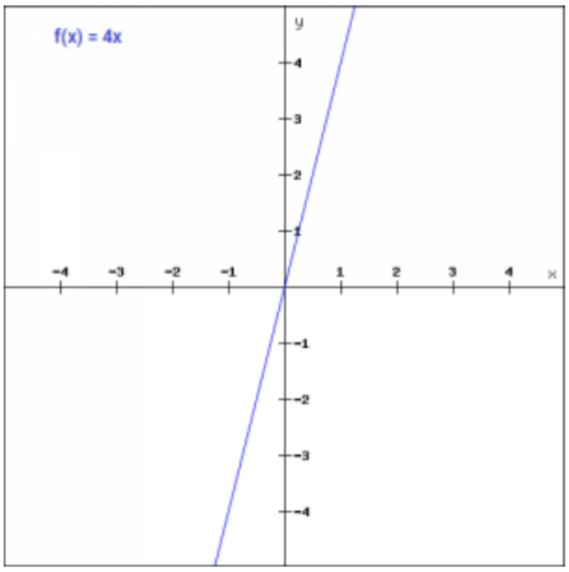


The Binary Step Function is the simplest type of an activation function. It can only be used in binary classification and cannot be used in case of multiclass classification problems. It has as output the value 1 if the set condition is met or the value 0 if the opposite is true.
The Binary Step Function can be described as follow:

$$y_k = \begin{cases} 1 \; if \; v_k \geq 0 \\ 0 \; if \; v_k < 0 \end{cases}$$

1.2.2.4. Linear activation function

Figure 6. Linear activation function (Sharma et al., 2020).



The Linear activation function is directly proportional to the input. It can overcome the major disadvantage of the Binary Step Function, the zero gradient because of the absence of the x component. The linear function can be described in mathematical terms as follow:

$$F(x) = ax$$

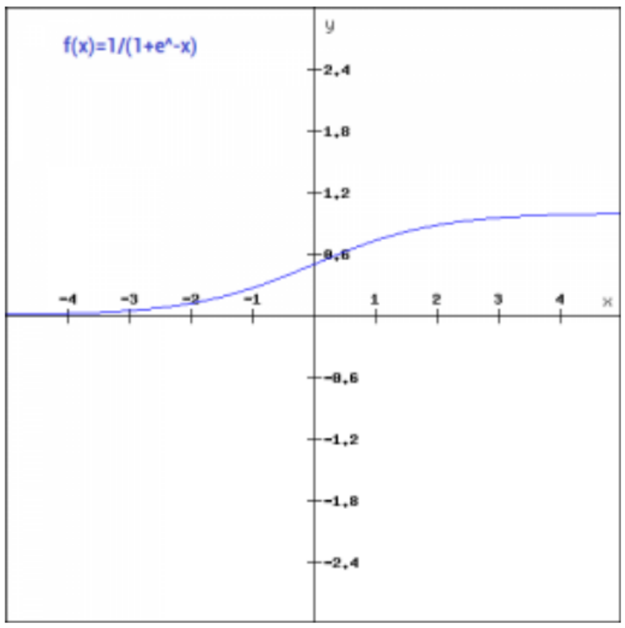
Where α is the slope, which is a constant value and it is selected based on the network.

Its derivative is not zero, but is equal to the constant α. The slope α is not zero, but a constant value which is independent of the input value x. The α updates the biases and the synaptic weights during backpropagation process.

The linear activation function does not benefit the ANN because it does not improve the error in each iteration due to the constant value of the slope. Finally, they do not have the ability to recognize complex patterns from the data thus it is used for simple tasks.

1.2.2.5. *Sigmoid activation function*

Figure 7. Sigmoid activation function (Sharma et al., 2020).



The sigmoid activation function is the most used kind of activation function because it is a non-linear function and it transforms values in the closed unit interval [0,1]. The sigmoid activation function can be described as follow:

$$sigmoid(x) = F(x) = \frac{1}{1 + e^{-\alpha x}}$$

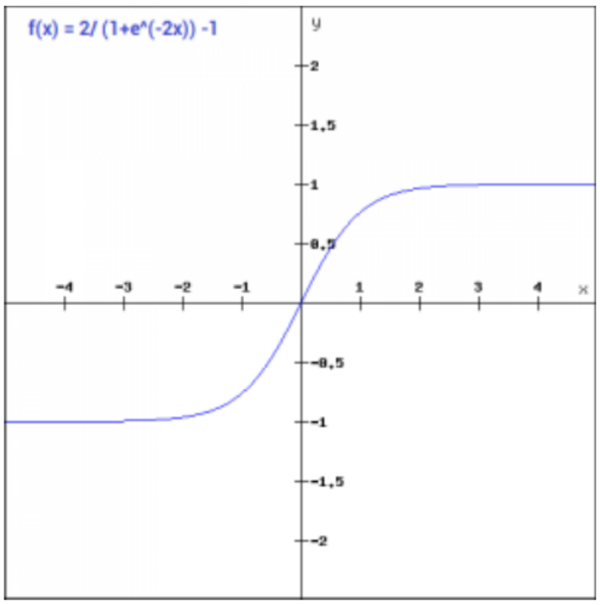
It is continuously differentiable, for its value α=1 the result of the first derivative is:

$$sigmoid'(x) = sigmoid(x)\big(1 - sigmoid(x)\big)$$

The value of α tending to infinity results in the conversion of the sigmoid activation function to a threshold function.

1.2.2.6. Tanh activation function

Figure 8. Tanh activation function (Sharma et al., 2020).

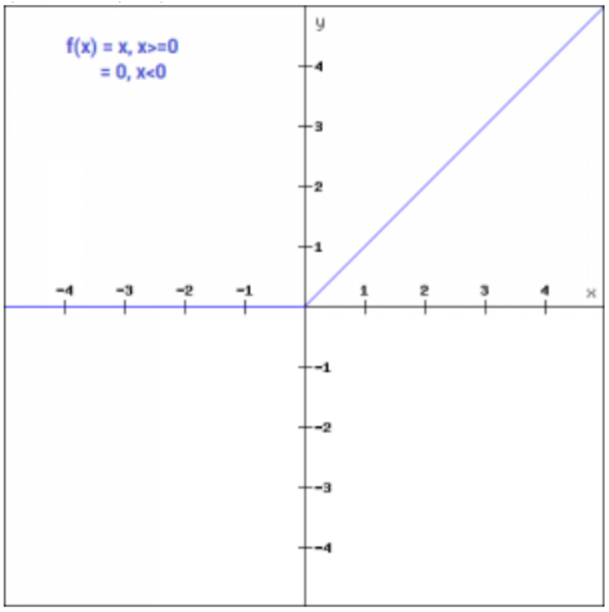


Tanh is similar to the sigmoid function. The differences lie on the symmetry, where Tanh is symmetric around the origin, and the gradient, the gradient of Tanh is more steep. This results in different signs of outputs from previous layers which will be fed as input to the next layer. The Tanh function can be defined as:

$$F(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

1.2.2.7. ReLU activation function

Figure 9. ReLU activation function (Sharma et al., 2020).

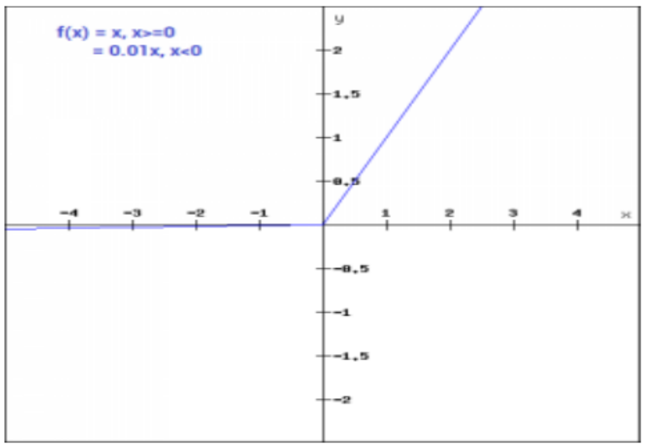


ReLU stands for Rectified Linear Unit. It works more efficiently compared to other activation functions because the neurons of the NN are not activated at the same time, only certain neurons are activated at a time. This implies that a neuron will be deactivated only when the output of linear is zero. It can be expressed mathematically as:

$$F(x) = \max(0, x)$$

1.2.2.8. Leaky ReLU activation function

Figure 10. Leaky ReLU activation function (Sharma et al., 2020).

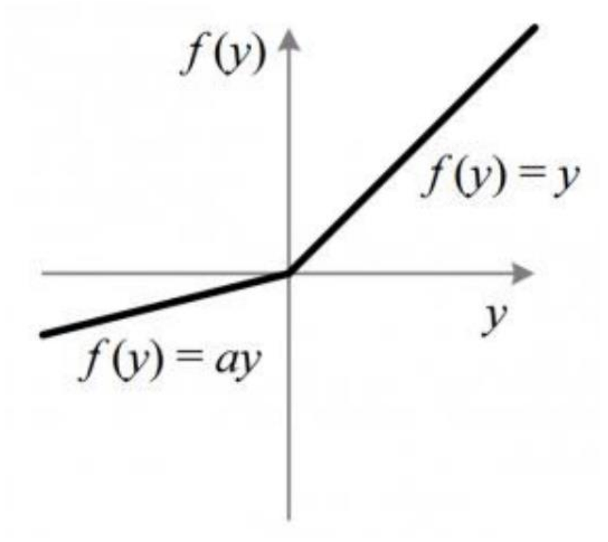


Leaky ReLU is an improved version of ReLU activation function. For negative values of x, Leaky ReLU is defined as very small linear component of x. It can be defined as:

$$F(x) = 0.01x, \qquad x < 0$$
$$F(x) = x, \qquad x \geq 0$$

1.2.2.9. Parametrized ReLU activation function

Figure 11. Plot of parametrized ReLU activation function (Sharma et al., 2020).

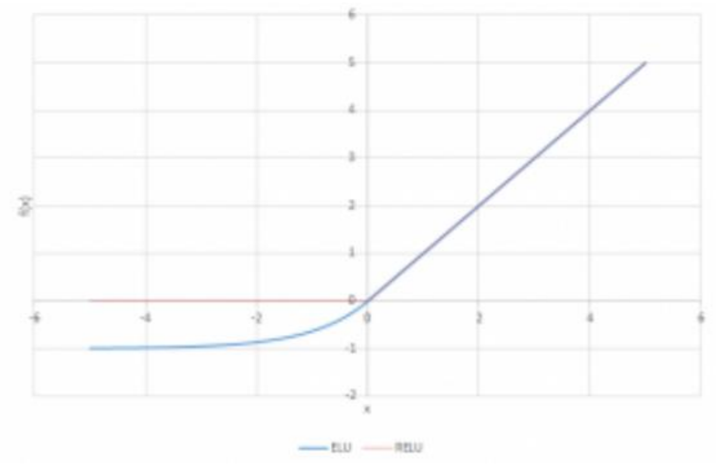


Parametrized ReLU function is another version of ReLU activation function which outperform the ReLU function because it resolves the problem of gradient of ReLU becoming zero for negative values if x by introducing a new parameter of the negative part of the function. It is described as:

$$F(x) = x, \qquad x \geq 0$$
$$F(x) = ax, \qquad x < 0$$

1.2.2.10. Exponential linear unit (ELU) activation function

Figure 12. Plot of ELU activation function (Sharma et al., 2020).

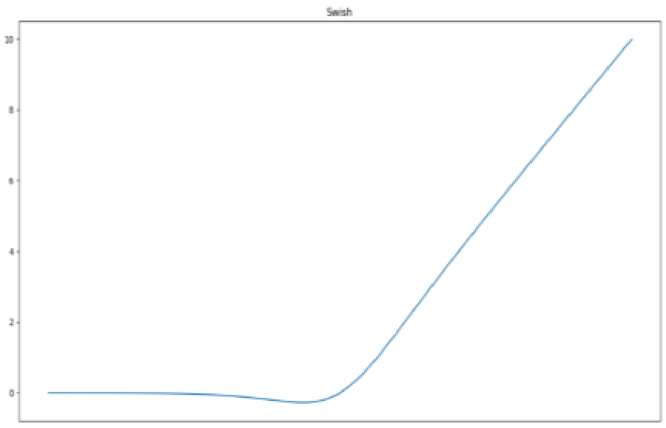


ELU introduces a parameter slope for the negative values of x and it uses a log curve for defining the negative values.

$$F(x) = x, \qquad x \geq 0$$
$$F(x) = a(e^x - 1), \qquad x < 0$$

*1.2.2.11. Swish activation funct*ion

Figure 13. Plot of swish activation function (Sharma et al., 2020).



The main characteristic of Swish activation function is that the value of the function may decrease even though the values of inputs are increasing. In some tasks, Swish outperforms the ReLU function. It can be defined as:

$$F(x) = x, \qquad x \geq 0$$
$$F(x) = a(e^x - 1), \qquad x < 0$$

*1.2.2.11. Softmax activation funct*ion

Softmax function combines multiple sigmoid functions. This type of activation function can be used for multiclass classification problems. The sigmoid functions' return values, 0 to 1, can be used as probabilities of a particular class' data points. It can be expressed as:

$$\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}} \qquad \text{for } j=1,\ldots,K.$$

(Sharma et al., 2020)

1.3. Deep Learning

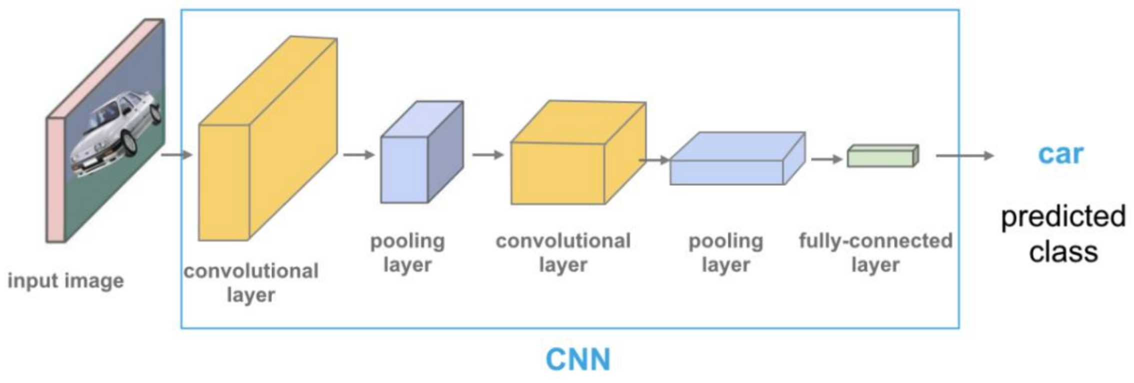
Deep Learning (DL) is a subset of Machine Learning. Its name derives from the fact that it consists of neural networks that form a complex of multilayered layers. DL is different from ANNs because it has more complex ways of connecting between levels, has more neurons, and requires more computing power for training and automatic feature extraction. Therefore, DL is defined as an ANN with a wide range of variables and levels based on a network architecture of unsupervised pre-trained networks, Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN). DL has proven to be quite useful in large-scale data studies, it has been hugely successful in applications such as speech recognition, computational vision, pattern recognition, recommendation systems and natural language processing. Today DL innovates in the fields of image recognition, image classification, face recognition, as well as object detection (Albawi et al., 2018).

2. Convolutional Neural Network

CNN is an ANN and it is inspired by the natural visual perception mechanism of living creatures. Specifically, the ability of the cells in animal visual cortex to detect light in receptive field (Zhang et al., 2019). CNN is a type of feed-forward NN and works by weight sharing (Pathak et al., 2018). It consists of multiple layers including fully connected layer, pooling layer, convolutional layers and non-linearity ones as presented at Figure 14. The fully connected layers and the convolutional layers have parameters, while the non-linearity layers and the pooling ones do not have parameters (Abiodun et al., 2018). Problems which are solved by CNNs should not have features which are spatially dependent. For example in a face detection problem, there is no need of detecting where the faces are located but to detect them despite their location (Albawi et al., 2017).

As regards the training process, CNN learns through backpropagation algorithm, by regulating the weights according to the target. Deep CNNs have been used extensively for this purpose. The main idea of how a CNN works in object detection can be described by an image which is convolved with an activation function to get feature maps. The feature maps are treated with pooling layers to get abstracted ones in order to reduce the spatial complexity of the network. The process is repeated for the desired number of filters and consequently feature maps are created. Finally, these feature maps are processed with fully connected layers to get output of image recognition showing confidence score for the predicted class labels. For improving the network's complexity and reduce the number of parameters, CNN employs a variety of pooling layers (Pathak et al., 2018).

Figure 14. Representation of CNN (Camacho et al., 2018).



2.1. Convolution

In mathematical terms, convolution is defined as:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da$$

This operation indicates the mapping of how a signal $x(a)$ is weighted with the signal $w(t)$. Function $w(t)$ can be considered as a filter applied to the signal $x(t)$. In case of images, the input signal is a two-dimensional and sampled. For deep learning purposes the function meant by convolution is the cross-correlation, it has similar properties as convolution, but the main difference is that it is not commutative. The cross-correlation function is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(m,n)K(i-m, j-n)$$

S is the output layer, K is the filter kernel, I is the input image, m and n are the position in the filter, I and j are the position in the image (Kapur, 2017).

2.2. Pooling

Pooling is the operation performed after the convolution. Its main feature is dimension reduction of the input into a patch. Two common functions used in the pooling operation are:

- Average Pooling: Calculate the average value for each patch on the feature map.
- Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.

The network gains two things:

1. Reduces the number of training parameters and computation cost, thus control overfitting.
2. Makes the network invariant to certain distortion.

(Browniee, 2019)

2.3. Fully connected layer

The output feature maps of the final convolution or pooling layer is transformed into a 1-Dimension array of numbers and they are connected to one or more fully connected layers, in which a connection takes place between every input and every output by a weight. Once the features extracted from the convolution layers and passed through the pooling layer they are mapped by a set of fully connected layers to the final outputs. A typical final fully connected layer consists of the same number of output nodes as the number of classes given to the network, the layer also is followed by a non-linear function (Patil & Rane, 2021).

3. Object Detection

Object detection is a supervised ML procedure of determining the instance of the class to which the object belongs and at the same time, estimating the location of the object by outputting the bounding box around the object. Each image in the training dataset must be accompanied with a file that includes the boundaries and the classes of the objects it contains. Object detection is applicable in a wide range of tasks, such as human computer interaction, defense, robotics and transportation (auto-pilot) (Pathak et al., 2018).

3.1. Performance Metrics

The performance metrics are useful to capture the performance of the model. Before we introduce more complicated metrics, we denote four basic variables:

1. True Positive (tp): The model predicts that an item belongs to a class and that is correct.
2. True Negative (tn): The model predicts that an item does not belong to a class and that is correct.
3. False Positive (fp): The model predicts that an item belongs to a class and this is incorrect.
4. False Negative (fn): The model predicts that an item does not belong to a class and this is incorrect.

Accuracy is used as a statistical measure of how well a model correctly detects or excludes a situation. That is, accuracy is the ratio of actual results (both TP and TN) to the total number of cases examined. Although a common metric, it is not widely used in object detection problems as the number of TN elements is much larger and predominant. It is defined as:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

Specificity is the ratio of the true negatives to the sum of true negatives and false positive predictions. Specificity indicates the ability of the algorithm to correctly identify negative results out of the real negative ones.

$$specificity = \frac{tn}{tn + fp}$$

Precision or Positive Predictive Value equals the number of true positive results divided by the total positive results and it shows the number of positives that are correctly identified by the model out of the total positive records.

$$precision = \frac{tp}{tp + fp}$$

Sensitivity or Recall is the ratio of the true positives predictions to true positives and false negative results, and it presents the positive results that are correctly identified by the algorithm out of the actual positive ones.

$$recall = \frac{tp}{tp + fn}$$

F1 score or F-measure is a metric who take into consideration the precision and the recall to measure the not correctly predictions.

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall}$$

Mean average precision (mAP) is defined as the mean of average precision across all K classes.

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

(Caelen, 2017; Dhananjay & Sivaraman, 2021; Sachdeva & Kumar, 2021; Vakili et al., 2020)

3.2. Transfer Learning

Transfer learning is a machine learning technique which works by reusing a pre-trained model that was originally built for another dataset. Transfer learning then reuses that model as a starting point for a new, most of the times, smaller dataset. Transfer learning can be applied for speeding up the model's training time and solve the problem of insufficient data. Usually, while building the CNN models, a lot of time is spent on building and connecting convolutional layers. Transfer learning works by using the previous neural network model to identify edges in the earlier layers, structures in the middle layer, and high-level features in the later layers (Burugupalli, 2020).

3.3. Detection Paradigms

The most used state-of-the-art object detectors for deep learning are separated into two main categories: two-stage detectors and one-stage detectors.

Two-stage frameworks divide the detection procedure into the region proposal and the classification phase. These models first propose several object candidates, known as regions of interest (RoI), using reference boxes (anchors). In the second step, the proposals are classified and their localization is refined. Two-stage detectors work more efficiently in terms of performance when compared to one-stage detectors.

One-stage detector contain a single feed-forward fully convolutional network that directly provides the bounding boxes and the object classification. They are much faster and are most used for real-time object detection applications. In the current work, we deal with one-stage detectors (Carranza-García et al., 2021).

3.3.1. YOLOv1

YOLO stands for "You Only Look Once" and the first version of YOLO algorithms, YOLOv1, is used for multiple object detection and is created by Joseph Redmon. Its uniqueness is the approach of the detection as a regression problem. It is a one-stage detector and runs simultaneously the evaluation and detection (Redmon et al., 2016). It is a single convolutional network which simultaneously predicts multiple bounding boxes and class probabilities for those boxes. The YOLOv1 uses Darknet, which is an open source neural network framework written in C and CUDA.

The main idea of how this network works is that it divides the input image into $S\ x\ S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the box is that it predicts.

The main drawbacks of the YOLOv1 was that the network struggles with small objects that appear in groups, also it doesn't perform well in generalizing the objects in new or unusual aspect ratios (Redmon et al., 2016)

3.3.2. YOLOv2

The newer version of YOLO, the YOLOv2 version, overcomes the limitations of YOLOv1 and it was published by Joseph Redmon and Ali Farhad at the end of 2016. The main changes that made YOLOv2 a better model in terms of performance is the following:

1. Batch normalization: Adding batch normalization to all the convolutional layers improved mAP by 2%. Also, it helped to tune the model and thus reduce any kind of overfitting.
2. High resolution classifier: Firstly the model is tuned at 448 × 448 resolution on ImageNet, this change give to the model the time to adjust its filters and increases mAP by 4%.
3. Anchor Boxes: More bounding boxes are used and k-means clustering is used to define the input dimensions of the anchor boxes.
4. Fine-grained features: It predicts detections on a 13 × 13 feature map, which is smaller than YOLOV1's. This improved in localization of small objects while remaining efficient for larger objects.
5. Multi-scale training: The YOLOv1 performed weak while detecting the objects with different image sizes. YOLOv2, randomly chooses the image dimension sizes with the minimum being 320 × 320 and the maximum being 608 × 608.

6. Darknet-19: Uses 19 convolutional layers and 5 max pooling layers, on top of the last convolutional layer will be a softmax layer for classification.

(Redmon & Farhadi, 2017)

3.3.3. YOLOv3

YOLOv3 published in 2018, it is more accurate but a little bigger than the previous versions (YOLO and YOLOV2). The main changes refer to:

1. Bounding Box Prediction: YOLOv3 uses logistic regression to predict an objectness score for each bounding box. Only assigns one bounding box prior for each ground truth object.
2. Class Predictions: YOLOv3 uses independent logistic classifiers for each class instead of a regular softmax layer. This change helps more complex classification cases.
3. Predictions across scales: To support detection YOLOv3 predicts boxes at 3 different scales.
4. Feature Extractor: Feature extraction is a hybrid approach. It uses successive 3 x 3 and 1 x 1 convolutional layers with some shortcut connections, it has 53 convolutional layers thus its name is Darknet-53.

(Redmon & Farhadi, 2018)

3.3.4. YOLOv4

Yolov4 released in 2020 and it is an improved version of YOLOv3 algorithm by having an improvement in the mAP by as much as 10% and the number of frames per second by 12%. The authors of YOLOv4 include a series of contributions in their paper titled a "bag of freebies". These are a series of steps that can be taken to improve the model's performance without increasing latency at inference time. Because they cannot affect the model's inference time, most of these make improvements in the data management and data augmentation of the training pipeline. These techniques improve and scale up the training set to expose the model to scenarios that would have otherwise been unseen. Also another improvement is “bag of specials” techniques, they change the network architecture and sometimes increase the cost of the output process (Bochkovskiy et al., 2020).

3.3.5. YOLOv5

YOLOv5 released only on GitHub in 2020 without a paper to accompany it. It is different from all other prior releases, as this is a PyTorch implementation rather than a fork from original Darknet. The major improvements includes mosaic data augmentation and autolearning bounding box anchors. YOLOv5 is extremely fast and lightweight than YOLOv4, while the accuracy is equivalent to the YOLOv4 benchmark.

With each training batch, YOLOv5 passes training data through a data loader, which augments data online. The data loader makes three kinds of augmentations: scaling, color space adjustments, and mosaic augmentation. The most novel of these is mosaic data augmentation, which combines four images into four tiles of random ratio.

The PyTorch framework allows the ability to half the floating point precision in training and inference from 32 bit to 16 bit precision. This significantly speeds up the inference time of the YOLOv5 model.

4. Implementation of Object Detection

An object detection implementation with YOLOv3, YOLOv4 and YOLOv5 networks is conducted in order to detect if an individual wears a face mask or not. In addition, a real time object detection is applied through webcam.

4.1. Data

Machine learning models depend mainly on data. Without applying high-quality training data even, the most well-functioning computer algorithms can be impractical. Hence, no other element is more fundamental in machine learning techniques other than quality training data.

Training data indicates the original data that is used to improve the model. The model uses the training data to build and improve itself. The quality of this data has deep effects on the model's succeeding development, which helps in setting a powerful model for any future applications that may use the same training data. Training data in machine learning involves a human contribution to examine and develop the data for machine learning procedures.

In this thesis, supervised learning was used to train the model with labelled training data. When the data is labeled, the dataset usually is marked with fundamental identification features. These features are very beneficial for the model to train and learn. Therefore, the accuracy of the model and its ability to identify the outcomes and perform high-quality predictions are largely affected by the extracted features from the dataset as well as the quality of labeling.

Therefore, the main purpose of this thesis is to train a model to detect whether a person wears a medical mask and locate it. This system could be used for surveillant purposes. In addition, it can perform detection at various places, such as airports and malls.

4.1.1. Face Mask Detection Dataset

The initial dataset used in this study is publicly available on Kaggle. The dataset has been enriched with images of people who wear mask from Google. The images have been labeled by using the tool LabelImg for object localization. The dataset consists of two classes, persons who wear a face mask (face_mask) and persons who do not wear a face mask (no_face_mask). The final dataset consists of 1694 RGB images. Figure 15 and 16 show a sample of the final dataset with images of people who wear a face mask and a sample of people who do not wear a mask, respectively.

Figure 15. Dataset sample of people wearing medical mask.

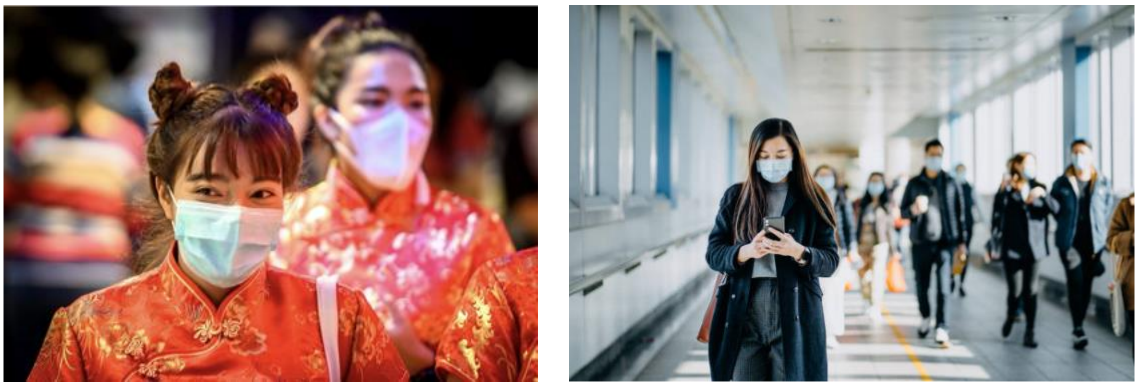


Figure 16. Dataset sample of people without wearing medical mask.

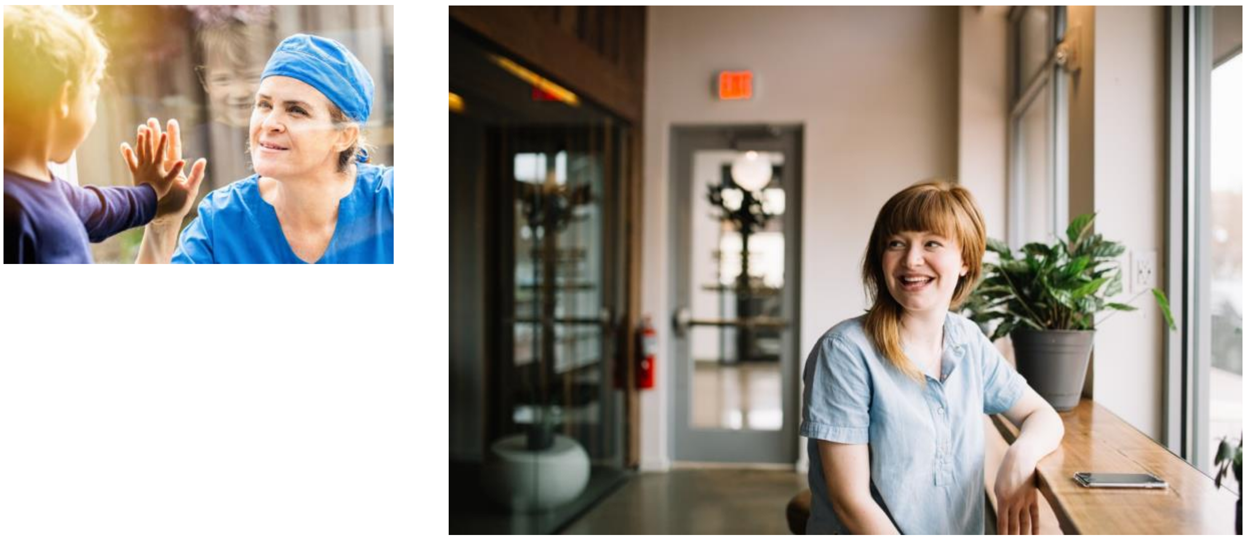


Figure 17. Distribution of classes.

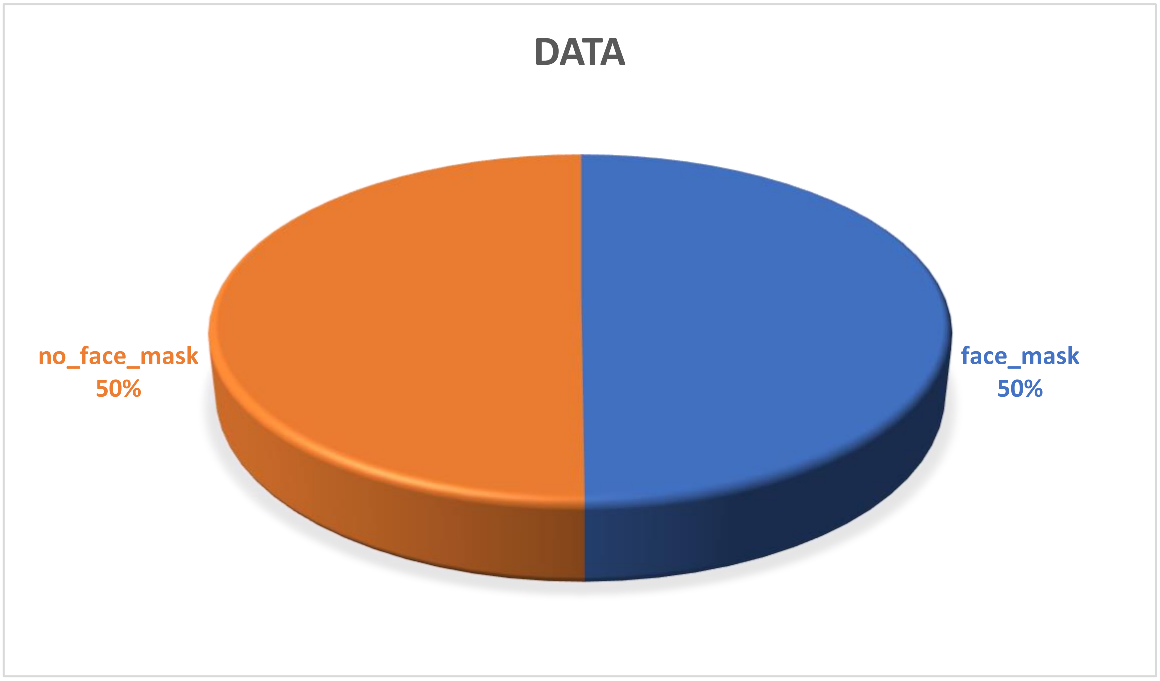


Figure 17 represents the distribution of the two classes. Specifically, there are total 7791 labels, 3905 (50.1%) correspond to people who wear a face mask and 3886 (49.9%) to people who do not wear a face mask.

4.3. Methods

Firstly, the dataset was randomly divided into training and test set which are composed of 1356 (80.0%) and 338 (20.0%) pictures, respectively. Three models have been used: YOLOv3, YOLOv4 and YOLOv5.

Some changes have been made for YOLOv3 and YOLOv4 implementations as regards the YOLOv3.cfg and YOLOv4.cfg configuration files. Specifically, the changes are presented below.

- Batches = 2
- Subdivisions = 8
- Max Batches = 4000
- Steps = 3800, 4200

- Filters = 21
- Classes = 2

As regards the YOLOv5 model, library wanb has been used. This is a library which provides a live monitoring of the training process. The YOLOv5 has 100 epochs of 16 batches.

All the above approaches were implemented in Python 3.8 programming language on Google Colab with Tesla P100-PCIE-16GB graphics card. As for the training process, transfer learning is applied, and the model was adjusted to the needs of the current work. YOLOv5 model is trained in the PyTorch framework, and YOLOv3 and YOLOv4 in the Darknet framework.

5. Results

In Table 1, the three models YOLOv3, YOLOv4 and YOLOv5 that are tested with the dataset produced the shown metrics.

Table 1. Performances of YOLOv3, YOLOv4 and YOLOv5 models.

	YOLOv3	YOLOv4	YOLOv5
Precision	0.91	0.86	0.92
Recall	0.52	0.85	0.88
F1-score	0.66	0.85	0.90
mAP_0.5	0.70	0.88	0.90

From Table 1 is evident that the YOLOv5 model outperforms the YOLOv3 and YOLOv4 implementations. YOLOv3 has a high precision but its recall is low, and that shows the YOLOv3 model needs improvements. YOLOv4 and YOLOv5 precisions and recalls are balanced, therefore their F1 score are higher compared to YOLOv3, although YOLOv3 has high precision.

Figure 18. YOLOv3 test images.

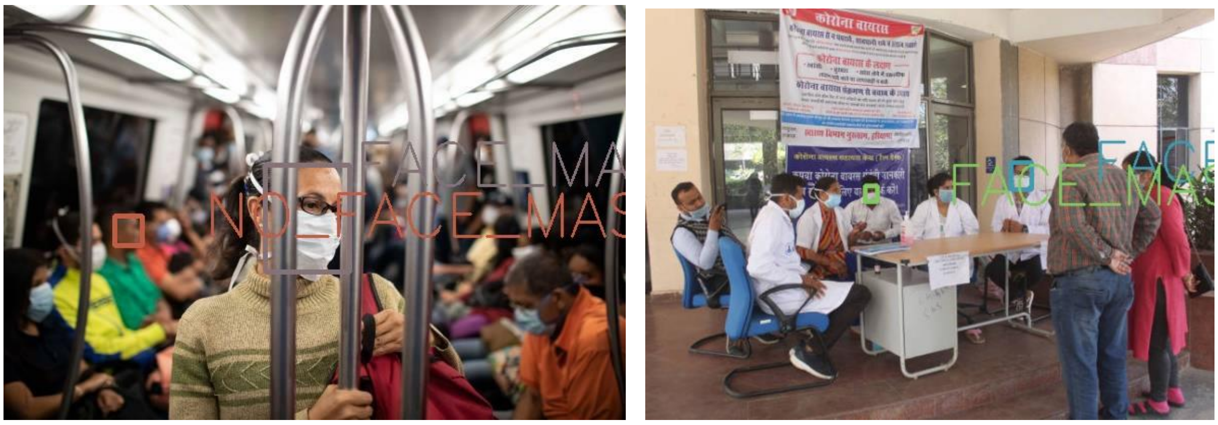


Figure 19. YOLOv4 test images.

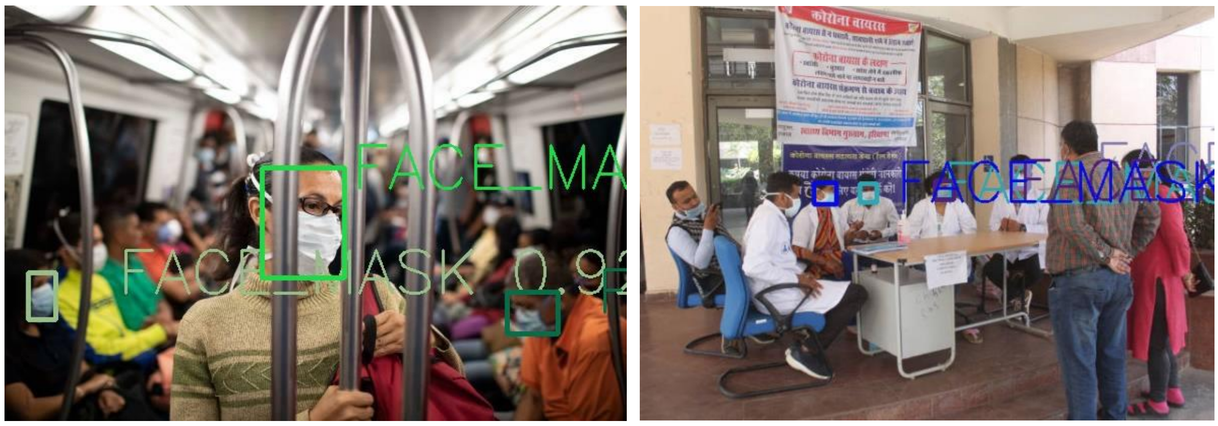
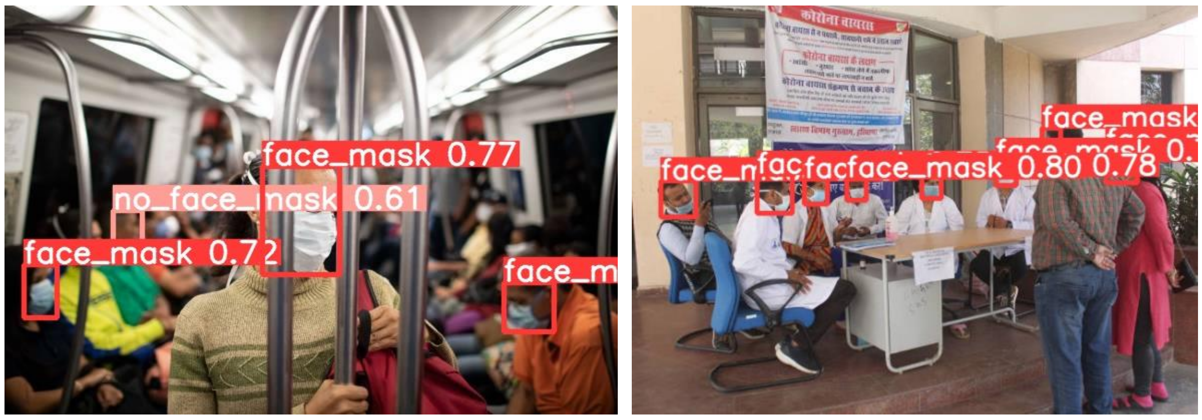


Figure 20. YOLOv5 test images.



Figures 18, 19 and 20 show the detections of YOLOv3, YOLOv4 and YOLOv5, respectively. It is evident that YOLOv5 achieves the highest number of detections compared to the other models.

Figure 21. YOLOv3 test video.

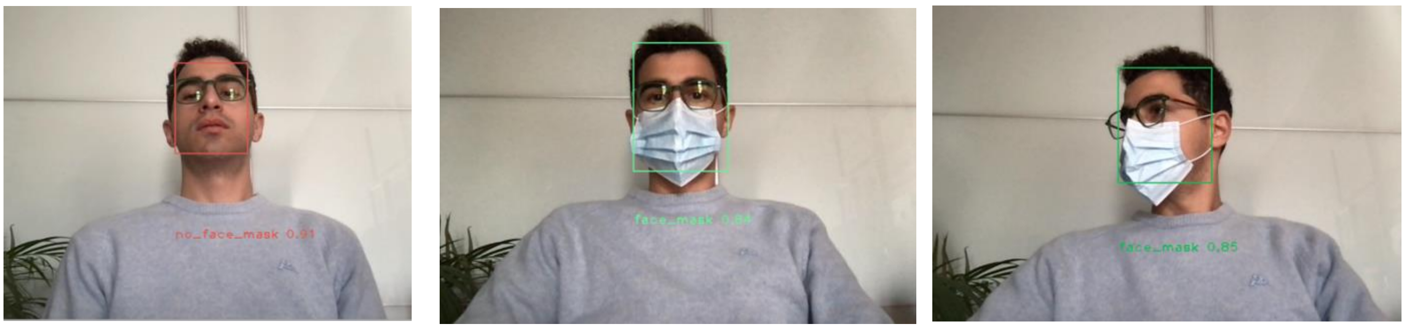


Figure 22. YOLOv4 test video.

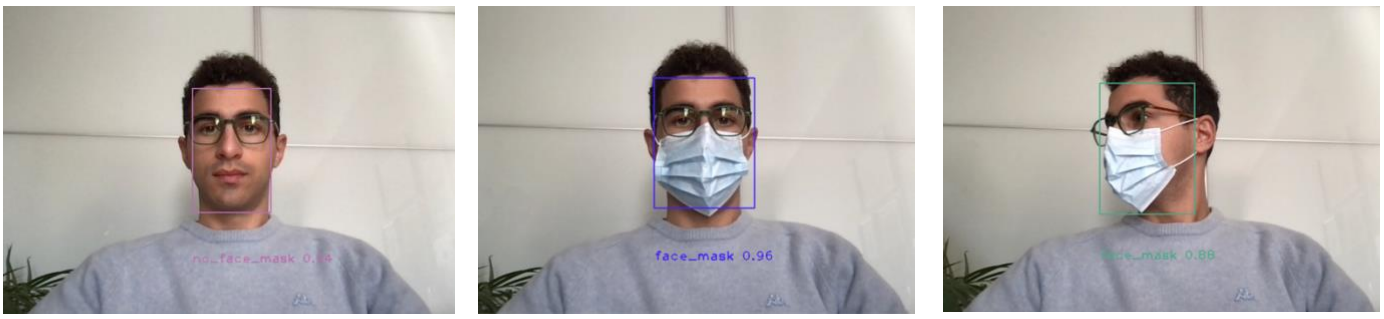
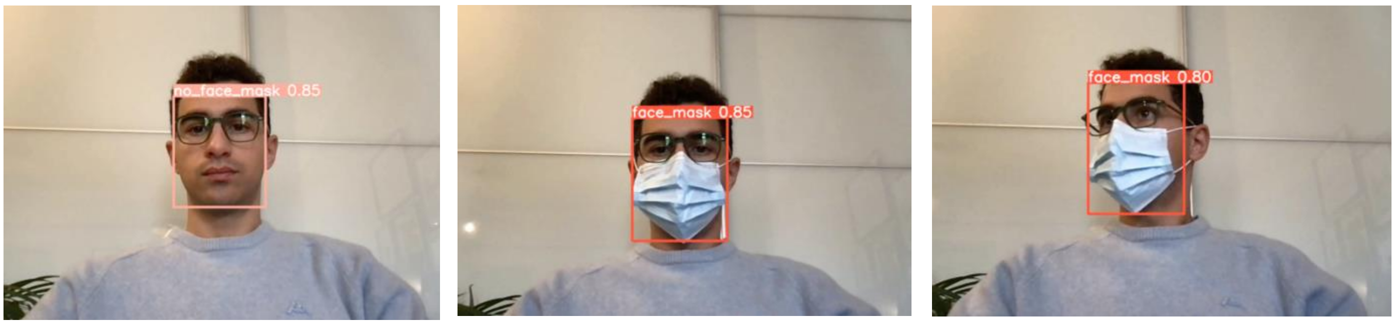


Figure 23. YOLOv5 test video.



Figures 21, 22 and 23 show the detections while the algorithms are implemented on a video through webcam.

Table 2. Size of the models.

	YOLOv3	YOLOv4	YOLOv5
Size (MB)	246,4	256	42,2

From Table 2, YOLOv5 has the most lightweight model size in comparison to YOLOv3 and YOLOv4 models.

6. Conclusions

This research aimed to assist in the abidance of the health protection measures against the spread of the virus that causes COVID-19 disease.

In the present work, we are interested in developing an object detection module that will detect either an individual wears a mask or not in real-time. Thus, YOLOv3, YOLOv4 and YOLOv5 models were selected due to their good detection speed and accuracy in real-time applications in order to find the most suitable object detection algorithm. From the results of this study, presented in Table 1, 2 and Figures 18-23, we conclude that YOLOv5 model claims to be very fast, it has a very lightweight model size, detects the most objects, and finally it satisfies the requirements of the current detection problem.

Bibliography

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. E., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186

Albawi, S., Mohammed, T. A. M., & Alzawi, S. (2017). Layers of a Convolutional Neural Network. *Ieee*, 16.

Batta, M. (2020). Machine Learning Algorithms - A Review . *International Journal of Science and Research (IJ*, *9*(1), 381-undefined. https://doi.org/10.21275/ART20203995

Bengio, Y. (2009). Learning deep architectures for AI. In *Foundations and Trends in Machine Learning* (Vol. 2, Issue 1). https://doi.org/10.1561/2200000006

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

Burugupalli, M. (2020). *IMAGE CLASSIFICATION USING TRANSFER LEARNING AND CONVOLUTION NEURAL NETWORKS* (Issue July) [North Dakota State University Graduate School]. https://doi.org/10.4324/9781315721606-101

Caelen, O. (2017). A Bayesian interpretation of the confusion matrix. *Annals of Mathematics and Artificial Intelligence*, *81*(3–4), 429–450. https://doi.org/10.1007/s10472-017-9564-8

Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P., & García-Gutiérrez, J. (2021). On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, *13*(1), 1–23. https://doi.org/10.3390/rs13010089

Dhananjay, B., & Sivaraman, J. (2021). Analysis and classification of heart rate using CatBoost feature ranking model. *Biomedical Signal Processing and Control*, *68*(December 2020), 102610. https://doi.org/10.1016/j.bspc.2021.102610

Ferreira, J., Callou, G., Josua, A., Tutsch, D., & Maciel, P. (2019). An artificial neural network approach to forecast the environmental impact of data centers. *Information (Switzerland)*, *10*(3), 1–20. https://doi.org/10.3390/info10030113

Hadidi, N. N., Cullen, K. R., Hall, L. M. J., Lindquist, R., Buckwalter, K. C., & Mathews, E. (2014). Functional magnetic resonance imaging as experienced by stroke survivors. *Research in Gerontological Nursing*, *7*(5), 200–205. https://doi.org/10.3928/19404921-20140820-01

Kumar, M., & Choudhary, R. (n.d.). *' Emerging Trends in Big Data , IoT and Cyber Security .'*

Lin, J.-W. (2017). Artificial neural network related to biological neuron network: a review. *Advanced Studies in Medical Sciences*, *5*(1), 55–62. https://doi.org/10.12988/asms.2017.753

Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of Deep Learning for Object Detection. *Procedia Computer Science*, *132*(Iccids), 1706–1717. https://doi.org/10.1016/j.procs.2018.05.144

Patil, A., & Rane, M. (2021). Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition. *Smart Innovation, Systems and Technologies*, *195*, 21–30. https://doi.org/10.1007/978-981-15-7078-0_3

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. http://arxiv.org/abs/1804.02767

Sachdeva, S., & Kumar, B. (2021). Comparison of gradient boosted decision trees and random forest for groundwater potential mapping in Dholpur (Rajasthan), India. *Stochastic Environmental Research and Risk Assessment*, *35*(2), 287–306. https://doi.org/10.1007/s00477-020-01891-0

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., & Müller, K. R. (2017). Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(11), 2660–2673. https://doi.org/10.1109/TNNLS.2016.2599820

Sharma, S., Sharma, S., & Athaiya, A. (2020). Activation Functions in Neural Networks.

International Journal of Engineering Applied Sciences and Technology, *04*(12), 310–316. https://doi.org/10.33564/ijeast.2020.v04i12.054

Simeone, O. (2018). A Very Brief Introduction to Machine Learning with Applications to Communication Systems. *IEEE Transactions on Cognitive Communications and Networking*, *4*(4), 648–664. https://doi.org/10.1109/TCCN.2018.2881442

Simon, H. (1992). Neural networks and learning. In *Institute of Physics Conference Series* (Vol. 127).

Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K. L. A., Elkhatib, Y., Hussain, A., & Al-Fuqaha, A. (2019). Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges. *IEEE Access*, *7*, 65579–65615. https://doi.org/10.1109/ACCESS.2019.2916648

Vakili, M., Ghamsari, M., & Rezaei, M. (2020). *Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification*. http://arxiv.org/abs/2001.09636

Zhang, Q., Zhang, M., Chen, T., Sun, Z., Ma, Y., & Yu, B. (2019). Recent advances in convolutional neural network acceleration. *Neurocomputing*, *323*, 37–51. https://doi.org/10.1016/j.neucom.2018.09.038