الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

Université Badji Mokhtar - Annaba
Badji Mokhtar – Annaba University

جامعة باجي مختار – عنابـــة

**Faculté : Technologie**

**Département : Informatique**

**Domaine : Mathématique-Informatique**

**Filière : Informatique**

**Spécialité : systèmes informatiques**

## Mémoire

## Présenté en vue de l'obtention du Diplôme de Licence

## Thème

# Detection and recognition of faces wearing a mask using CNN

**Présenté par :** Fujo Dickson Baraka
Kimani Samuel Mbugua

**Encadrant :** Radia Amirouche M.A.A Université Badji Mokhtar - Annaba

**Année Universitaire : 2021/2022**

# Vote of Thanks

First of all, we thank God the Almighty for giving us the will and the perseverance to do this work.

To you Madam Radia Amirouche, our supervisor: This project wouldn't have been a success if it were not for you. We take this opportunity and address our deepest appreciation for your priceless advice, solid orientation, supervision, your presence, support throughout this project period to enrich it by your outstanding proposals and offering help whenever needed. Your support and kindness testify of your great generosity. Having crossed paths with you is the best thing ever happened to us.

We can't complete this project without extending our most sincere thanks to the people who have helped us and who have contributed in various ways to the development of this work as well as the success of this wonderful academic year. We thank all our teachers for their dedication, patience and contribution and dedication throughout our academic course.

Finally, we would like to extend our most sincere thanks to all our loved ones and friends, who have always encouraged us during the realization of this project.

# Dedications

To our dear parents, for whom we owe what we are today. Thanks to your prayers, support and sacrifices throughout our course. May God the almighty preserve you and give you a healthy and long life.

To our dear friends at the 19 Mai residence (Owen, Bongani, Fred, Courage and Tino), no dedication would be enough to let you know how we feel about you. We will just say thank you for your encouragement and co-operation whenever called upon. We wish you a life full of success and prosperity.

# Table of Contents

# Table of Figures



# List of Tables

# Introduction

## 1. Project Context

This work aims at developing a system that helps to determine the availability of face mask on human faces with the help of camera. If no mask is found, then it will recognize the face and note down the name of person with time in the excel sheet, which will only be accessed by admin.

## 2. Problems

The spread of COVID-19 can be limited if people strictly adhere to the use a facial mask. Sadly, people don't follow these rules properly and due to the huge population in the world, humans can't enforce the COVID precautions that's why the spread of this virus continues. Detecting the people not obeying the rules and giving details to relevant managing authorities can be a solution in reducing the spread of coronavirus.

## 3. Motivations

This encouraged us to explore face mask detection system to monitor people wearing masks in crowded places. For people who are already infected by a coronavirus, there are very high chances of spreading the disease when they talk or breathe. It is observed that coronavirus spread through fluid particles that are less than 0.0002 inches in diameter, which are usually emitted when people speak.

## 4. Objectives

Our objective is to develop of a Real-Time model that automatically detects whether the face mask is put on and correctly or not and possibly note names of people without a mask. It employs a Convolutional Neural Network (CNN), a technique that draws bounding boxes (red or green) around people's faces, depending on whether a person is wearing a mask or not, bring a proportion of the face covered to the recommended percentage, and keep the record of it accurately without generating overfitting.

The model is trained on a real-world dataset of images of faces with properly worn masks, wrongly masks, and without masks to achieve the desired accuracy.

## 5. Project Contents

This thesis comprises of three chapters:

Chapter 1: Detection and Recognition of faces wearing a mask: In this chapter we have face mask detection, face recognition and presented state of art of related works.

Chapter 2: Convolutional Neural Networks (CNN): we introduce the architecture and the different layers and hyper-parameters of a CNN network as well as CNN advantages and limitations.

Chapter 3: System Conception and Implementation: This chapter is dedicated to explaining the architecture of our system as well as the different stages of its design. It also illustrates the work environment as well as the results obtained from experimental study.

# Chapter 1 : Detection and Recognition of faces wearing a mask

## 1. Introduction

In this chapter we define the general concepts on face mask detection and face recognition as well as the process phases involved. We continue the chapter by presenting different methods on both mask detection and facial recognition under Convolutional Neural Network with the different approaches that can exist **[1-8]**. Finally, we close this chapter by a state of the art of related works on face mask detection and facial recognition in a general way, and more particularly those using CNN **[10].**

## 2. Face detection

Face Detection is the detecting faces from an image or video **[38]**. With the advancement of technologies, computers are becoming more and more intelligent day by day. In today's world, there is nothing for which machines cannot be trained for. Machines can be trained to interact with human in natural way by looking at people through cameras, they can be trained to understand and talk with humans by voice/natural language or chat and so can they be trained for face detection **[47]**.

## 3. Face detection methods of faces with or without mask

There are two major approaches for face detection technique:

- Feature Base Approach
- Image Base Approach

In Feature Base approach, machine detects face with help of extracting structural features like eyes; nose, mouth, etc while in Image Base approach machines rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non-face images **[37,21]**. One of the most popular face detection algorithms is Viola-Jones Algorithm. Other face recognition methods include: matching approach.

## 4. Face recognition

Artificial Intelligence research has introduced us with many great technologies earlier and is still doing continuously. One of the many wonders from AI research is Face Recognition. People usually make mistake of thinking face detection and face recognition as same things. But in reality, there is a lot of difference in both the terms. Face detection simply means detecting faces from the videos or images, but on the other hand face recognition is an advance learning which lets us know the identity of the individual detected face **[11-13]**.

**5. Face Recognition methods of faces with or without mask)**

Similarly to men, if we expect machine to tell us the identity of every person, it will also fail in doing so therefore, we create the datasets of the known faces and then we train the machine to recognize those persons easily. We first store the encodings of known faces in the dataset. Then, whenever the machine captures any face using image or video, it will first find the encodings of that image; and then it will compare captured image encodings with dataset encodings **[51]**. If from the stored encodings, any encoding matches with the captured image encodings, then the machine will tell us the identity of person. In this project, we have used "face recognition" module built with dlib using CNN in deep learning **[15]**.

**Occlusion removal approach**: In order to avoid a bad reconstruction process, these approaches aim to detect regions found to be occluded in the face image and discard them completely from the feature extraction and classification process. Segmentation based approach is one of the best methods that detect firstly the occluded region part and using only the non-occluded part in the following steps. For instance, Priya and Banu divided the face image into small local patches **[26]**. Next, to discard the occluded region, they applied the support vector machine classifier to detect them. Finally, a mean-based weight matrix is used on the nonoccluded regions for face recognition. Alyuz et al. applied an occlusion removal and restoration **[3]**. They used the global masked projection to remove the occluded regions. Next, the partial Gappy PCA is applied for the restoration using eigenvectors.

# 6. State of Art and Review

The trained architecture with multiple layers of convolution and max-pooling connected to dense neural network achieved **99.6%** accuracy on distinguishing people with and without a facial mask for previously unseen test data **[42]** [Table 2]. We will now examine several popular state-of-the-art.

Table 1. Sample of survey on related works

| Sr No. | Paper | Advantages | Limitations |
|---|---|---|---|
| **1.** | Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers **[1]** | The system is evaluated with different classifiers. Different classifiers like MobileNetV2, RESNET50, VGG16, each of them was compared with Optimizers like ADAM, ADAGRAD, SGD to yield the highest accuracy. ADAM gave maximum accuracy. | This framework might be incorporated with a framework which can coordinate with a framework actualizing social distancing which can make it a healthy framework that can welcome emotional effect on the spread alongside alarm framework interfacing |

| | | | |
|---|---|---|---|
| **2.** | Covid-19 facemask detection with deep learning and computer vision **[3]** | The system comprises of MobileNet as the spine which can be very well utilized for high and low calculation situations. In order to extract more robust features, learning is used to gain weights from a similar task face detection, which is trained on large datasets. By the advancement of face mask detection, it can be detected whether a person is wearing face mask or not and permit their entry would be of great help to the society. | In future studies, a more extensive facemask wearing dataset including images and videos will be collected and labelled with more details in order to improve the performance. |
| 3. | A Cascade Framework For Masked Face Detection **[4]** | This algorithm is based on a recently planned CNN course structure comprises of three CNNs.To defeat the overfitting issue due to lack of training samples, the model is pre-trained with WIDER FACE dataset, and finetuned them with MASKED FACE training set. This masked face detection algorithm is assessed on the MASKED FACE testing set, it achieves satisfactory results. | The system can be used in CCTV footages to identify whether a person is wearing a mask correctly so that he does not pose any hazard to others. |
| 4. | Multi-Stage CNN Architecture for Face Mask Detection **[5]** | A two-stage face mask detector was introduced. First stage uses pre-trained RetinaFace model for face detection, after comparing its performance with Dlib and MTCNN. Second stage uses NASNetMobile based model for classifying faces as “masked” or “unmasked”. Besides, centroid tracking is used to improve performance on video streams. | At present, the model gives 5 FPS inference speed on a CPU. We plan to improve this up to 15 FPS, making our model deployable for CCTV cameras, without need of a GPU. Stage 1 and stage 2 models can be easily replaced with improved models that would give better accuracy and lower latency. |
| 5. | Deep Learning Framework to Detect Face Masks from Video Footage **[6]** | In this work, methodology a profoundly successful face identification model is utilized for getting facial pictures and signals. A particular facial classifier is constructed utilizing deep learning for the errand of deciding the presence of a face mask in the facial pictures distinguished. The subsequent methodology is strong and is assessed on a custom dataset got for this work. The proposed approach was discovered to be successful as it depicted high exactness, review, and precision esteems on the picked dataset which contained videos with fluctuating occlusions and facial angles. | Mass screening is little difficult in crowded places like railway stations, bus stops, streets, schools, colleges, etc. so the system yielding better accuracy can be created. |
| 6. | A convolutional neural network (CNN) approach to detect face using tensorflow and keras **[7]** | A deep convolutional neural network (CNN) is utilized to extricate highlights from input pictures. Keras is utilized for actualizing CNN additionally Dlib and OpenCV for adjusting faces on information pictures. Face acknowledgment execution is assessed utilizing a custom dataset. | This innovation can be additionally evolved to be utilized in different avenues like ATMs, getting to private records, or other sensitive materials. This can make other safety efforts, for example, passwords and keys obsolete. Another way that innovators are looking to execute facial acknowledgment is inside subways and other transportation outlets. They are hoping to use this innovation to use faces as credit cards to pay for our transportation charge. |

| | | | |
|---|---|---|---|
| 7. | A Review on Face Mask Detection using Convolutional Neural Network **[8]** | Face mask identification can be done through different strategies. Essentially convolutional neural network technique is utilized quickly. The precision and decision making are exceptionally high in CNN contrasted with others through AI techniques in image processing. | Facial mask detection and nonmasked face detection accuracy provided high variations. |

## 7. Conclusion

We have in this chapter defined the general concepts on face mask detection and face recognition as well as the process phases involved, presented different methods and approaches on both mask detection and facial recognition under Convolutional Neural Network and finally, closed with a state of the art of related works more particularly those treating mask detection using CNN. In the next chapter we will discuss a general overview of Convolutional Neural Network (CNN) **[29,37].**

# Chapitre 2 : Convolutional Neural Network

## 1. Introduction

In the last few decades, Convolutional Neural Network architectures have proven themselves in many machine learning applications. Before discussing the detailed main phases of our work, we first present some basic concepts of Convolutional Neural Network **[34-36].** We later proceed on the training and testing phase of building our model then finalise with the advantages and limitations of CNN as an imagery processing technique.

## 2. Neural networks

Neural networks are the functional unit of Deep Learning which is a part of Machine Learning and falls under Artificial Intelligence. Neural Networks or Artificial Neural Networks (ANNs) or Simulated Neural networks (SNNs) are similar to neurons in our brain. We can simply denote Neural as neurons and network means they are connected to each other **[3,13,27]**. Neural Networks are similar to human brain, as brain learns from experiences and practice, we want the same to be implemented in machine.

We implement biological neuron into machine with the help of Artificial Intelligence **[75].** We then take input and initialize it with some weights and then perform some mathematical function and algorithms accordingly. Neural networks are widely used for research in Artificial Intelligence, Speech Recognition, Image Recognition and many more. And also, many applications like Google Translation, Google Assistant uses ANNs **[13].** The most common types of Neural networks are: Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

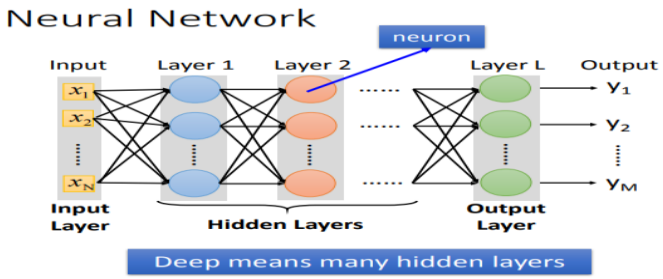



*Fig 1.Basic neural network structure.*

# 3. Convolutional neural networks (CNN)

CNN is a feed-forward neural network, a sub field of deep learning which is generally used for analysis of visual imagery by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image. This Neural Network uses the already supplied dataset to it for training purposes, and predicts the possible future labels to be assigned. A portion of the territories where CNNs are broadly utilized are image recognition, image classification, image captioning and object detection etc. Tasks where recommendation systems, contextual importance or natural language processing (NLP) is considered, CNNs come handy **[43-45]**. The key chore of the neural network is to make sure it processes all the layers, and hence detects all the underlying features automatically. A CNN is a convolution tool that parts the different highlights of the picture for analysis and prediction. During training, when the weights increases and the dimensions are lost, gradient descent is used to solve this. CNN hyper parameters affecting the efficiency of the CNN model include: filters, kernel size, stride and padding.

## 3.1 CNN overview

CNNs have several advantages over DNNs, including being more similar to the human visual processing system, being highly optimized in structure for processing 2D and 3D images, and being effective at learning and extracting abstractions of 2D features **[7-9].** The max pooling layer of CNNs is effective in absorbing shape variations. Moreover, composed of sparse connections with tied weights, CNNs have significantly fewer parameters than a fully connected network of similar size. Most of all, CNNs are trained with the gradient-based learning algorithm, and suffer less from the diminishing gradient problem. Given that the gradient-based algorithm trains the whole network to minimize an error criterion directly, CNNs can produce highly optimized weights **[18].**

## 3.2 CNN Architecture

### 3.2.1 Convolution Layer

Convolutional layer is the backbone of any CNN working model **[7].** This layer is the one where pixel by pixel scanning takes place of the images and creates a feature map to define future classifications. In this layer, feature maps from previous layers are convolved with learnable kernels. The output of the kernels

goes through a linear or non-linear activation function (such as sigmoid, hyperbolic tangent, Softmax, rectified linear, and identity functions) to form the output feature maps **[68]**. Each of the output feature maps can be combined with more than one input feature map.

### 3.2.2 Sub-sampling Layer

The subsampling layer performs the down sampled operation on the input maps. This is commonly known as the pooling layer. In this layer, the number of input and output feature maps does not change. For example, if there are *N* input maps, then there will be exactly *N* output maps **[9].** Due to the down sampling operation the size of each dimension of the output maps will be reduced, depending on the size of the down sampling mask. Two types of operations are mostly performed in this layer: average pooling or max-pooling. In the case of the average pooling approach, the function usually sums up over N×N patches of the feature maps from the previous layer and selects the average value. On the other hand, in the case of max-pooling 36], the highest value is selected from the N×N patches of the feature maps.

### 3.2.3 Classification Layer

This is the fully connected layer which computes into a single vector, the score of each class from the extracted features from a convolutional layer in the preceding steps. The final layer feature maps are represented as vectors with scalar values which are passed to the fully connected layers. The fully connected feed-forward neural layers are used as a soft-max classification layer. In LeNet, AlexNet and VGG Net two to four layers are used **[7,8,18,42]**. In the backward propagation through the CNNs, the fully connected layers update following the general approach of fully connected neural networks (FCNN) **[16].**

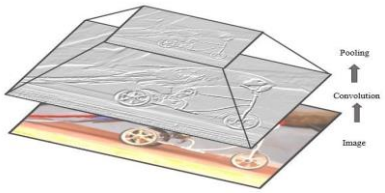



***Fig 2****. Example of convolution and pooling operation.*

### 3.3 Popular CNN architectures

We will now examine several popular state-of-the-art CNN architectures **[19].** In general, most deep convolutional neural networks are made of a key set of basic layers, including the convolution layer, the sub-sampling layer, dense layers, and the soft-max layer. The architectures typically consist of stacks of several convolutional layers and max-pooling layers followed by a fully connected and SoftMax layers at the end. Some examples of such models are LeNet, AlexNet, VGG Net, NiN and all convolutional (All Conv).

Other alternative and more efficient advanced architectures have been proposed including GoogLeNet with Inception units, Residual Networks, DenseNet, and FractalNet **[11]**. The basic building components (convolution and pooling) are almost the same across these architectures. However, some topological differences are observed in the modern deep learning architectures. Of the many DCNN architectures, AlexNet,

VGG, GoogLeNet, Dense CNN and FractalNet are generally considered the most popular architectures because of their state-of-the-art performance on different benchmarks for object recognition tasks **[10,6]**. Among all of these structures, some of the architectures are designed especially for large scale data analysis (such as GoogLeNet and ResNet), whereas the VGG network is considered a general architecture. Some of the architectures are dense in terms of connectivity, such DenseNet. Fractal Network is an alternative of ResNet.

## 3.4 CNN Algorithm

The structure of classical CNN is shown in Fig. 3. There are two parts in CNN: the first part is CPNN, and the second part is FCNN. In CPNN, the first layer is an input layer, and the following layers of CPNN are several pairs of convolutional layers and pooling layers [13]. In FCNN, the first layer is an input layer, and the second layer of (Fully connected Neural Network) **FCNN** is an output layer, both layers of FCNN are fully connected.

The relation and features of (Convolutional Pool Neural Network) **CPNN** and FCNN are as follows.

(1) The input layer of CPNN is the input layer of CNN.

(2) The last layer of CPNN is the input layer of FCNN.

(3) The output layer of FCNN is the output layer of CNN.

(4) The activation function of the convolutional layer in CPNN and the output layer in FCNN is sigmoid function.

(5) There are not any activation functions in the input layer and pooling layer of CPNN and the input layer of FCNN.

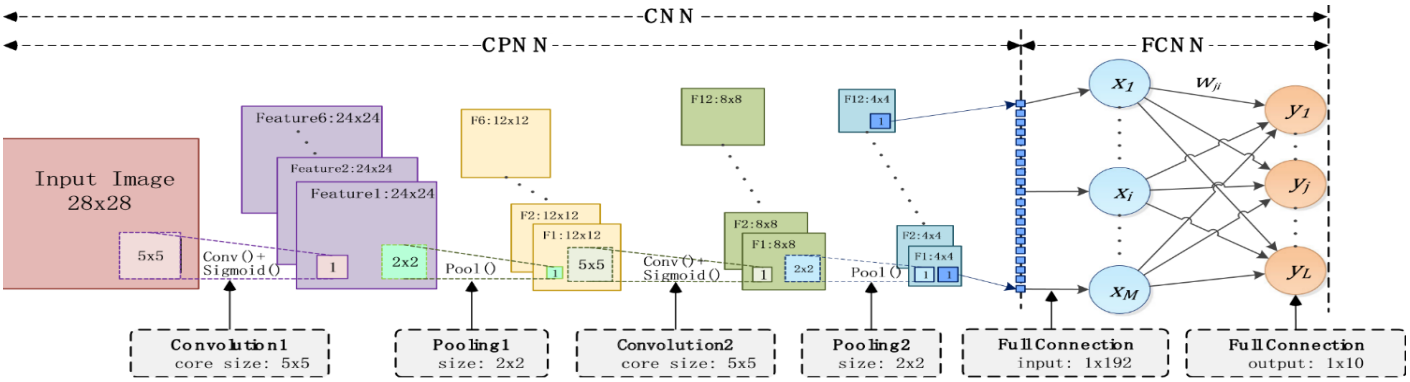



*Fig 3. Structure of CNN **[75]***

The algorithm of CNN can be described as follows:

(1) Initializing weights between layers and bias of neurons.

(2) Forward propagating.

(3) Calculating the mean square error (MSE) of all samples according to the loss function.

(4) Calculating back propagating errors for each layer, the results of derivation by the chain rule **[11]**.

(5) Applying gradient to adjust the weights and bias according to the back-propagated errors.

(6) Repeating the step (2) to step (5) until the MSE is small enough.

(7) Evaluating the accuracy, precision and efficiency.

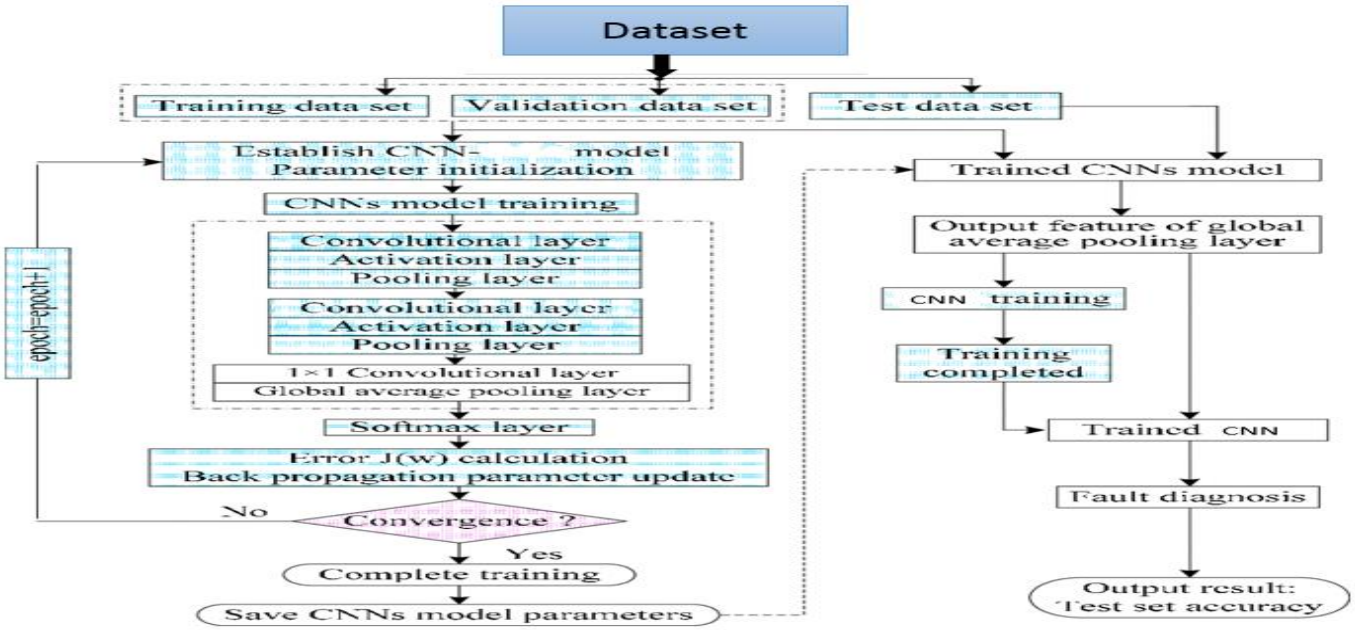



*Fig 4. CNN Algorithm [**20**]*

Training algorithm of CNN

**Input:**
***train_x, train_y:*** features and labels of Training Set
***test_x, test_y:*** features and labels of Test Set
**Output:**
$w_{ij}^{l}$, $b_{j}^{l}$,: weights and bias of **Convolution and Pooling Neural Network** (**CPNN**).
$w_{jk}$, $b_{jk}$: weights and bias of **Full Connection Neural Network** (**FCNN**, **FCNN** have 2 layers)
**Required parameters:**
***max_time*:** maximum value *n* of *n*ACs in every 1SP
***target_error*:** when the current training error is less than target error, the training is finished
$\eta\_CPNN$: **the** learning rate of CPNN
**Initialization work:**
$w_{ij}^{l}$, $b_{j}^{l}$, $w_{jk}$, $b_{jk}$: weights and scaling parameters of **CNN** (**CPNN**+**FCNN**) are set as random numbers.
***t*:** ***t*** is the current simulation time, which is initialized as ***t*=1** before the training loop**.**
$L(t)$**:** $L(t)$ is the mean square error at simulation time ***t.*** $L(t)$ is initialized as $L(1) = 1$>***target_error***.
**Begin:**
1: Set the required parameters and complete the initialization work
2: **while** $t <$ ***max_time*** and $L(t) >$ ***target_error***
3: **for all** *trainingSet*:
4: ***train_p*** (prediction of label) is calculated according to ***train_x*** and forward calculation formula 1-9
5: **end for**
6: $L(t)$ is re-calculated as $L(t) = \frac{1}{2}\sum_{n=1}^{N}(train\_p(n) - train\_y(n))^2$, $N$ is the total number of *trainingSet*.
7: $\Delta w^{l}$ , $\Delta b^{l}$ , $\Delta w_{ij}^{-1}$, $\Delta b_{j}^{-1}$ are updated according to the formula 20-23
8: $w^{l}(t), b^{l}(t), w_{ij}^{-1}(t), b_{j}^{-1}(t)$ are adjusted according to the formula 24-27
9: $t$++
10: **end while**
**End**

*Fig 5. CNN pseudocode **[23]***

## 3.5 Training phase:

In Deep Learning, the algorithm builds an "internal representation" for itself so that it can perform the required tasks (prediction, recognition, etc.), you must first enter a set of sample data so that it can train and improve, thus the dataset called the training set.

The construction of the CNN model in python consists of several layers of inputs, hidden and outputs, Keras offers as discussed in the conception section **[7]**.

### 3.6 Test phase:

Once the model is built, trained and saved we go to the next step to do this we will add a part for the learning and a part for the test that we had prepared and then we do the learning with the total of these data, train dataset and validation dataset the image training and validation data. We've defined our model and compiled it ready for efficient computation, now it's time to run the model on some data we can train or fit our model on our loaded data by calling the fit( ) on the model, training is done on:**Epoch** as seen in conception section **[48]**.

## 4. Merits of CNN

1. The usage of CNNs are motivated by the fact that they can capture/are able to learn relevant features from an image /video at different levels similar to a human brain.
2. CNN architectures are able to learn powerful features from a small amount of a dataset label using multilayer perceptrons that go far beyond the methods based on performance features.
3. CNNs automatically detect/extract important features without any human supervision. CNN models generally perform superior compared to the manual extraction (end to end) process.
4. CNNs are weight sharing. If you have a one layered CNN with filters, you can simply calculate parameters of such a CNN hence more efficient in terms of memory and complexity.
5. CNNs outperform NNs on conventional image processing image classification, object detection, segmentation and recognition tasks compared to the Inception model, Resnet50 **[47].**
6. For a completely new task / problem CNNs are very good feature extractors. This means that you can extract useful attributes from an already trained CNN with its trained weights by feeding your data on each level and tune the CNN a bit for the specific task. This is also called pre-training and CNNs are very efficient in pre-trained tasks compared to NNs which saves memory and time.
7. CNN or ConvNet takes in a fixed size input and generates fixed-size outputs.
8. CNN uses a connectivity pattern between its neurons and is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field.
9. CNN architectures are perfect for orthology tasks and characters such as emojis and byte.
10. CNNs are great for short texts (e.g headlines). Most likely can improve your text classifier model by adding a linear layer (like cbow) after for pooling layer.
11. A fully connected layer learns features from all the combinations of the features of the previous layer, where a convolutional layer relies on local spatial coherence with a small receptive field **[8]**.
12. Learning of accurate pattern and insights from the provided data.

13. Can provide better and accurate results than typical machine learning techniques/algorithms if tuned better and fed a good amount of data.
14. They provide translation equivariance, meaning that a shifting in the input data does not alter the representation of the input but rather linearly shifts the input in the latent space.
15. They yield themselves to be more readily parallelizable, hence why we had a GPU revolution in deep learning research hence giving dep learning research a huge boost.
16. They can be treated as backbone function approximators that can be implemented as visual/audio feature extractors for downstream applications that need to reason about image/audio data.
17. They closely resemble mammal visual cortex and there are neuroscientific links between biological visual cortex and artificial CNNs **[18].**

# 5. Limitations of CNN

1. While the structure of a ConvNet aims to mitigate over-fitting, you generally need a large amount of data depending on the complexity of the task at hand for a convolutional neural network to work effectively.
2. CNNs performance is very sensitive to architectural details and hardware configuration, in especially with the GPU processor to expedite the training process and as a result, it takes a very long time, especially with large datasets **[12]**. It sometimes employs complex architectures
3. CNNs are translation-invariant, they are generally bad at handling rotation and scale-invariance without explicit data augmentation.
4. CNN architectures can be more difficult with long texts when you really need a large vocabulary.
5. Fully connected layers are incredibly computationally expensive. That's why we use them only to combine the upper layer features **[55]**.
6. Requires more computing power: External GPU depending on network architecture and data size.
7. They are now being swept out by visual transformers that offer higher speeds of training.
8. Data Augmentation: CNNs are not still not quite robust to adversarial attacks, rotations and reflections.
9. They don't have a sense of memory state, and they are rather limited for sequential modelling (such as language models, speech recognition etc.)
10. They require a fixed size regular structure that renders it less useful for unstructured data with irregular geometry such as graph data.
11. Performance issues occur if you are noticing big data tools because CNNs capture hierarchical information unlike RNNs which capture temporal order, that is, order in time **[33]**.
12. You need a well-defined loss function for supervised learning in contrast with some other tools like genetic algorithms where an ordinal output you can have step changes in loss.

## 6. Conclusion

In this chapter, we have discussed the basic concepts of Convolutional Neural Network before presenting the detailed training and testing phase of building our model **[15].** We finally closed the chapter with the advantages and limitations of CNN as an imagery processing technique. In the next chapter we will present the detailed conception and implementation or our work.

# Chapter 3 : Conception and Implementation

In this chapter we define the approach established to design a robust system to assist in the in the development of face mask detection and facial recognition model by use of the famous mobileNetV2, a CNN architecture in order to ensure the diversity of the input data of the two classification models and to ensure the good learning, and accurate results without generating overfitting **[17,21]**.

## A. Conception

### 1. Introduction

This chapter is divided into two parts. We first present the conception process covering building of dataset. We will further proceed to present the methodology of our model, training and testing as well as UML diagrams to show the internal architecture of the system. The second part is devoted to the presentation of the development environment used as well as the integrated libraries and other possible tools used in the implementation of our CNN model, the discussion of the various experimental results obtained during the empirical analysis including the calculation of the matrix of confusion **[63,58]**.

### 2. Dataset Acquisition

**2.1 Definition:** **Data Set**: with-without-mask

It is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. We have constructed our database of images in form of a folder. In the main dataset folder there are two sub-folders which contain images with masks and without images. The images are coloured in the 3-RGB colour level. Masks play a crucial role in protecting the health of individuals from respiratory disease, as it is one of the few precautions available for COVID-19 in the absence of immunization **[2].** With this dataset, it is possible to create a model to detect people wearing masks, not wearing them, or wearing masks inappropriately (which will be classified as not wearing a mask). This dataset contains **1428** images **(703**

without mask and **725** with masks) belonging to different individuals having masks on different levels from least to perfection.

Fig 6**.** Data samples

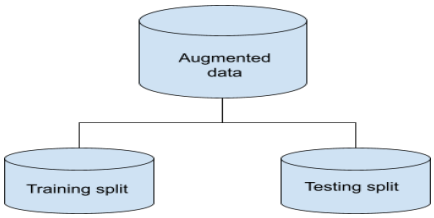



Fig 7**.** Splitting masked face image dataset into training and testing sets

### 2.2 Building the masked face recognition dataset

Our masked face image dataset consists of **1428 untrained images** (to be trained with CNN model) of different individuals without, fully and partially masked, each of which is 50×50 pixels.

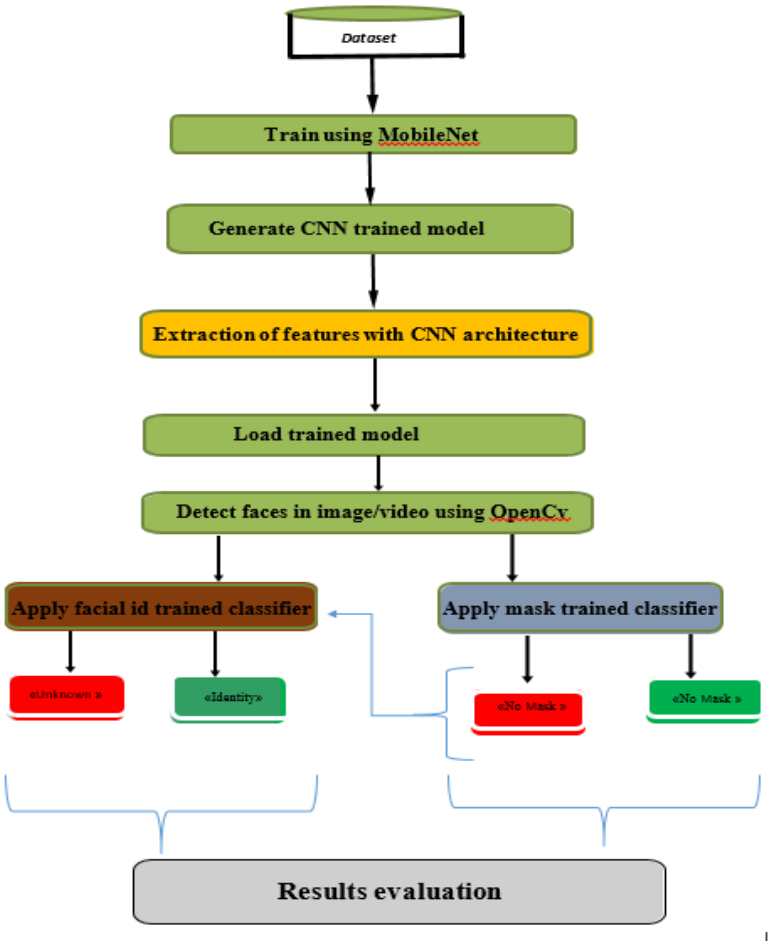
Loading this entire dataset in memory at once would require a lot of space so instead, we organized our dataset on disk so we can use Keras' ImageDataGenerator class to yield batches of images from disk without having to keep the entire dataset in memory.

## 3 Proposed system

In Mask Detection and Face Recognition System, face masks are detected and if person without mask is detected, then the person is recognizes and his/her name is noted **[72].** The UML diagrams in the next section show also demonstrates the internal working. When the camera captures video frames, face detection is performed first. Then the image processing of detected face is performed through model which decides whether the person wore mask or not, later with the help of face recognition module, person without mask is recognized and his/her name is noted in the excel sheets **[74,31].**

### 3.1Organigram

We are currently working on linking/combining facial recognition to face mask detection so that the name of the person captured without mask to be noted and printed in an excel sheet which can be accessed by admin. Modules like sklearn, NumPy, matplotlib**,** os are also used for small functions in the project **[30].**




This model will be prepared in two stages: **Stage1:** i). Loading of the dataset in the system

ii). MibileNetV2 classifier is used to train the data

**Stage2:** i). Load the classifier

ii). Detect faces in the input image or video stream

iii). Apply the trained classifier

iv). The final output is generated by classifying as masked face or unmasked face (and identity or unknown for the case of face recognition)

Fig 8. Application Architecture**.**

# 4 Methodology

The major focus of this project is to recognize the people without wearing face masks **[37].** Names of the people found will be noted in the excel sheet with time, which can be accessed only by admin after which the admin can take actions accordingly. This will certainly reduce the chances of people getting infected by virus in the organization, department or even in public places. Method implemented in this project are described below step by step. First the dataset containing images with mask and without mask is collected from various sources such as internet and manually captured willing friends. Then the model is trained for the dataset using MobileNet Neural Network which is using ImageNet as weights and algorithm used is Adam optimizer. From the dataset, **80%** of data was used for training and rest **20%** for testing. We trained our model with various numbers of epochs such as 5, 10, 15, 20, 50 and 100 each time automatically generating a model and a graph. From our observations, we concluded that the model with 100 epochs had the best results so we adopted it as our final model.

The model trained was **99.6%** accurate in detecting masks where OpenCV library was used to access camera and then the face detection technique based on CNN is performed. Also, the known face datasets, face encodings with their names were stored with the help of face recognition module. After the model is loaded, if the face captured by camera is wearing mask, then the bounding box with “Mask” written on its top left is shown on the person’s face each time returning a percentage of the region of interest covered. Otherwise, encodings of the faces are compared with the stored face encodings of known faces dataset then, if the

encodings match with stored encodings, a bounding box with the name of person on its top is shown, and if in case it does not matches then “UNKNOWN” is written on top of bounding box.
To implement our model, we used MobileNetV2 CNN to train the model with different layers of convolution and pooling operation **[13,60].** Before this, we are aiming to do data augmentation to increase our dataset for better performance **[28].** We have different optimizers in Keras like ADAM, ADAMAX, NADAM, SGD, ADAGRAD **[68].** Out of all these optimizers, it is found that the ADAM optimizer gives the best accuracy. Our model uses the convolutional neural network (CNN) under Deep Learning to quickly analyse and test machine learning capabilities to detect the face mask worn by people in given video or images. It takes an image data as input, captures all the data and sends it to its fully connected neural layers which processes the final output which represents a binary prediction on the image.
To train, validate and test the model, we used the dataset; a solid training set that allows us to achieve good precision and reduce bias which consisted of **725** images of masked faces and **703** exposed faces. The model was induced on images and video transfers live.
First, a collection of  software 'neurons' are created and wired together, allowing them to send messages to each other after which the network is asked to solve a problem, which it tries to do again and again, reinforcing to each time the connections that lead to success and diminishing those that lead to failure. Subsampling units use "Maxpooling" (MP) to become active if at least one of their inputs is active; their answers are insensitive to certain small image shifts **[39-43]**.
We used MobileNetV2 architecture as the framework for the classifier, which uses the one-shot multi-box detector as the face detector. It has a middle expansion layer which uses deep-light convolutions to filter out features as a source of nonlinearity. In addition, different hyper parameters are tried including batch size and epochs. The model uses TensorFlow, Keras, and OpenCV deep learning to fulfil the task of using the video stream to capture the frames in real time.

### 4.1 UML Diagrams

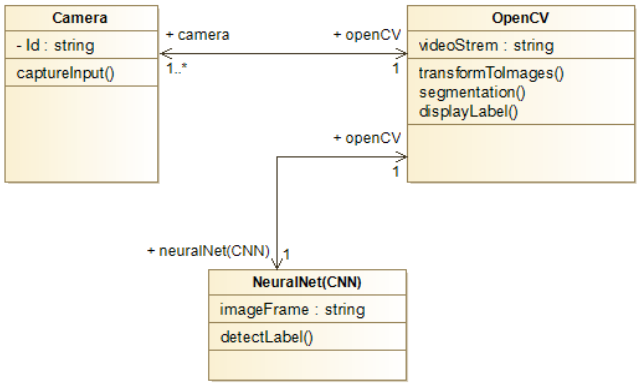



Fig 9: Class Diagram.

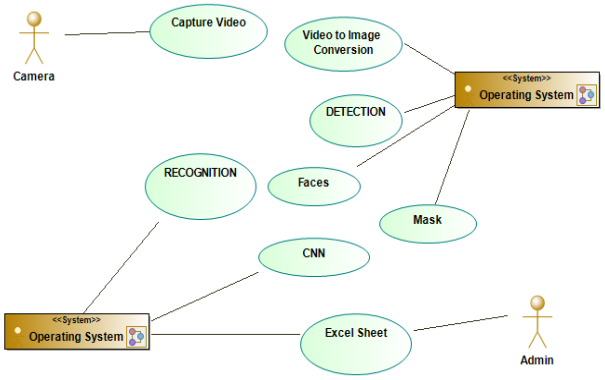



Fig 10.  Use Case Diagram.

Once the webcam is on, user’s face is captured on the real time camera via OpenCV video stream **[3].** Once more OpenCV transform the video to images then segments the images converted. From this point, CNN applies the trained model, detects/recognise the label then displays the output. The output generated is always binary (either mask/no mask in the case of facemask detection or person’s identity/unknown in the case of facial recognition) and displays text on the screen.

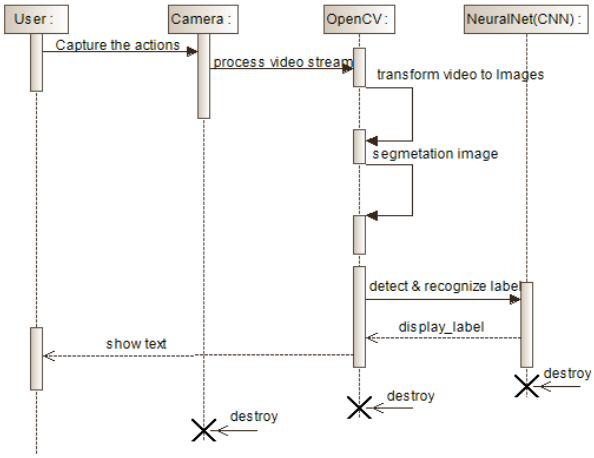



Fig 11: Sequence Diagram.

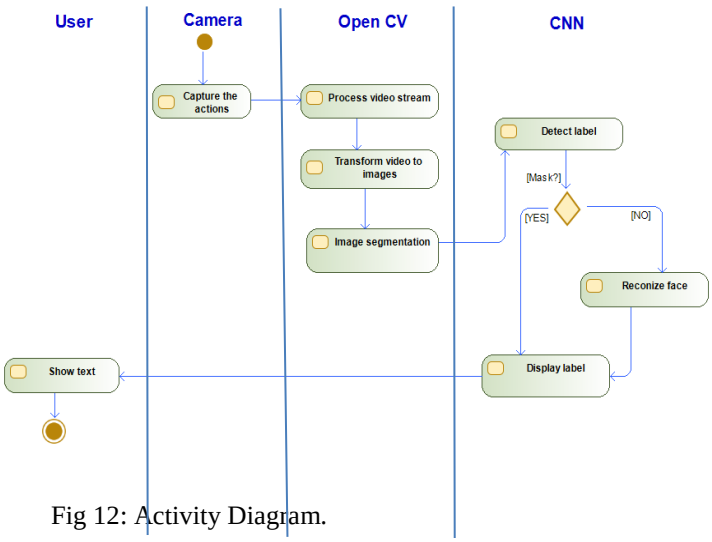



Fig 12: Activity Diagram.

# 5 Model learning phase

```
def création_model_avec_CNN(root):
 print("Étape 4: Construction du model CNN...")

 root = tf.keras.applications.MobileNetV2(weights="imagenet", include_top=False,
     input_tensor=tf.keras.layers.Input(shape=(256, 256, 3))) #three augments for RGB

 """ Construct le modèle de tête du modèle qui sera placé au-dessus du modèle de base: 'relu' est la couche d'activation et 'softmax' pour classification d'images"""
 tModel = root.output
 tModel = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), padding='same', activation="relu", input_shape=(256,256,3))(tModel)
 tModel = tf.keras.layers.MaxPooling2D(pool_size=(7, 7))(tModel)
 tModel = tf.keras.layers.Flatten(name="flatten")(tModel)
 tModel = tf.keras.layers.Dense(128, activation="relu")(tModel)
 tModel = tf.keras.layers.Dropout(0.5)(tModel)
 tModel = tf.keras.layers.Dense(2, activation="softmax")(tModel)
 global model
 model = tf.keras.models.Model( inputs=root.input, outputs=tModel)

 """ boucler et figer toutes les couches du modèle de base afin qu'elles ne soient pas mises à jour lors du premier 'Training' """
 for couche in root.layers:
     couche.Trainable = False
 return model
```

Fig 13: Building a CNN model.

The construction of the CNN model to perform the required tasks (prediction, recognition, etc.), you must first enter a set of sample data so that it can train and improve, thus the dataset called the training set **[76].** Python consists of several layers of inputs, hidden and outputs, Keras offers:

### 5.1 Conv2D Layer :

A first convolution layer will learn small patterns, a second convolution layer will learn larger patterns made up of features from the first layers, and so on. This allows ConvNets to effectively learn increasingly complex and abstract concepts **[25-58]**.

**Filter:** Sets the number of output filters (the number of neurons) used in the convolution operation. In our case is 64 filters.

**The size of the pattern**: the number of pixels it will contain also call kernel 3×3 pixels. Activation: (RelU) Rectified Linear Unit is a function that transforms all negative value data into 0, with the aim of not activating all neurons at the same time **[Web 19].**

**Input_shape:** This is the starting tensor that you send to the first hidden layer. This tensor should have the same shape as your training data. Example: input_shape (width, height, depth): input_shape(256,256,3).

**Width and Height:** Image width and height in pixels.

**Depth:** The number of channels for the image.

### 5.2 MaxPool2D Layer:

The MaxPooling is also a convolution layer. It extracts patterns and trends from data. But where Conv2D extracts features from an image to create feature maps, MaxPooling2D extracts the most important value from each pattern in feature maps. [Web 19]**Kernel:** a kernel of size (7,7).

### 5.3 Flatten Layer:

The Flatten layer makes it possible to flatten the tensor, to reduce its dimension. It takes as input a 3D-tensor and returns a 1D-tensor. It makes it possible to establish a connection between the convolution layers and the basic layers of Deep Learning **[44].** It allows to diffuse the data through the layers by reducing its dimension.

### 5.4 Dense Hidden Layer:

Define the number of nodes we want, we have chosen **128** neurons in the first hidden layer.

### 5.5 Dense Layer Output:

This layer is considered as the final layer of the model. After the model is defined, we can compile it before training the model then configure the learning process by calling the compile method which accepts three arguments:

**Loss:** this is the cost function that the model will use to minimize errors **[72].** It can be defined by its name (example: loss='binary_crossentropy').

Optimizer: aims to reduce the current error that is defined in loss (modify neuron weights).

**Metrics:** Metric functions are similar to loss functions, how accurate our neural network is on the predictions that make it.

## 6 The test phase (launch of learning):

The testing process for the CNN models is accomplished by using the testing data which are completely unknown to the model **[28]**. The classifier makes predictions on each image class and finally compares the calculated predicted class with the true class. After a classifier is trained, it can predict the class of a new entry from the testing set **[56]**. Once the model was built and saved we proceeded to the next step: to do this we added a part for the learning and a part for the test that we had prepared and then did the learning with the total of these data, train_dataset and test_dataset (the image training and test data).

We've defined our model and compiled it ready for efficient computation, now it's time to run the model on some data we can train or fit our model on our loaded data by calling the fit( ) on the model, training is done on:**Epoch :** We go through all the rows of the training dataset (the iterations) and each Epoch is composed of one or more Batches, depending on the size of the chosen Batch and the model is adapted to many epochs **[74].**

**Steps_per_epoch:** Integer.Total number of steps (sample batches) to produce from the generator before declaring an epoch over and starting the next epoch. It should usually equal sky (num_samples/batch_size). Optional for sequence: if not specified, will use them (generator) as number of steps. This is called model convergence.

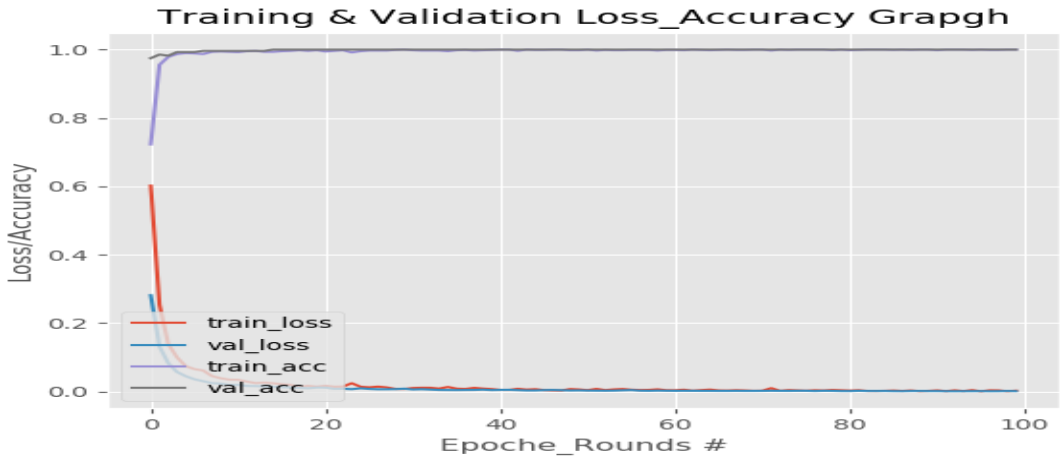



Fig 14**.** Train, loss and accuracy graph of our model

```
Étape 6:  Apprentissage de modèle construit avec 100 époques...
Epoch 1/100
35/35 [==============================] - 181s 5s/step - loss: 0.6154 - accuracy: 0.8153 - val_loss: 0.0723 - val_accuracy: 0.9860
Epoch 2/100
Epoch 7/100
35/35 [==============================] - 157s 4s/step - loss: 0.0069 - accuracy: 0.9991 - val_loss: 0.0104 - val_accuracy: 0.9965
Epoch 8/100
35/35 [==============================] - 152s 4s/step - loss: 0.0046 - accuracy: 1.0000 - val_loss: 0.0050 - val_accuracy: 0.9965
Epoch 9/100
35/35 [==============================] - 161s 5s/step - loss: 0.0087 - accuracy: 0.9991 - val_loss: 0.0038 - val_accuracy: 1.0000
Epoch 10/100
35/35 [==============================] - 165s 5s/step - loss: 0.0080 - accuracy: 0.9991 - val_loss: 0.0012 - val_accuracy: 1.0000
Epoch 11/100
35/35 [==============================] - 175s 5s/step - loss: 0.0034 - accuracy: 0.9991 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 12/100
35/35 [==============================] - 171s 5s/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0029 - val_accuracy: 1.0000
Epoch 13/100
35/35 [==============================] - 167s 5s/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.0031 - val_accuracy: 1.0000
Epoch 14/100
35/35 [==============================] - 163s 5s/step - loss: 0.0047 - accuracy: 1.0000 - val_loss: 0.0031 - val_accuracy: 1.0000
Epoch 15/100
35/35 [==============================] - 159s 5s/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0039 - val_accuracy: 0.9965
Epoch 16/100
35/35 [==============================] - 157s 4s/step - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.0069 - val_accuracy: 0.9965
Epoch 17/100
35/35 [==============================] - 158s 5s/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 0.0038 - val_accuracy: 1.0000
Epoch 18/100
35/35 [==============================] - 158s 4s/step - loss: 0.0104 - accuracy: 0.9973 - val_loss: 0.0056 - val_accuracy: 0.9965
Epoch 22/100
35/35 [==============================] - 159s 5s/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.0080 - val_accuracy: 0.9965
Epoch 23/100
35/35 [==============================] - 160s 5s/step - loss: 0.0022 - accuracy: 0.9991 - val_loss: 0.0089 - val_accuracy: 0.9930
Epoch 24/100
35/35 [==============================] - 159s 5s/step - loss: 6.8023e-04 - accuracy: 1.0000 - val_loss: 0.0084 - val_accuracy: 0.9965
Epoch 25/100
35/35 [==============================] - 158s 5s/step - loss: 6.7376e-04 - accuracy: 1.0000 - val_loss: 0.0050 - val_accuracy: 0.9965
Epoch 26/100
35/35 [==============================] - 158s 5s/step - loss: 4.0564e-04 - accuracy: 1.0000 - val_loss: 0.0037 - val_accuracy: 1.0000
Epoch 27/100
35/35 [==============================] - 158s 4s/step - loss: 8.2404e-04 - accuracy: 1.0000 - val_loss: 0.0026 - val_accuracy: 1.0000
```

Fig 15. Dataset training results with 100 epochs

The fit() function will return a list with four values **[63,57]** namely: the model loss on the train dataset, the model accuracy on the train dataset, the model loss on the validation dataset and the accuracy of the model on the validation dataset, each time printing the loss and accuracy, followed by the final evaluation of the trained model on the training dataset thought the **100 epochs** span.

## 7. Model evaluation

Once we had trained the model, we applied a new dataset pred_dataset that the model had never seen before, making predictions is as simple as calling the **predict()** function on the model. The purpose of the assessment is to best estimate the performance of a model on new data.

# B. Implementation

## 1. Implementation steps

This phase consists of different steps: Importing libraries, importing the dataset, data pre-processing, learning phase, test phase and finally, evaluation.

### 1.1 Importing libraries (package):

#### 1.1.1 Development environment

In this section, we will discuss the tools and programming language used to implement our software, evaluate the performance of the designed models and discuss the results obtained, as well as the challenges and obstacles we faced throughout the work.

**a). Programming languages**

- **Python**

Python is one of the most popular programming languages and is known for its simple syntax hence allowing developers to create applications writing fewer lines of code and making them more productive and its vast collection of libraries which make most tasks much easier and it is platform independent meaning it can run on any operating system. It is open source, available to all users thus making is undebatable choice for our task. Regarding the field of machine learning, Python stands out all particularly by offering a plethora of very high quality libraries, covering all types of learning available on the market including in the field of computer vision, object recognition and machine learning **[51]**. This entire project was done in Python for its simplicity of its syntax and the ease with which it can be learnt.

b**). Environment and tools**

- **Anaconda3**

Anaconda is a free and open-source distribution of the Python and R programming languages applied to the development of applications dedicated to data science and machine learning (large-scale data processing, predictive analysis, scientific computing), which aims to simplify package management and deployment **[7].**

- **Tensorflow**

TensorFlow is an end-to-end open source machine learning platform. It is one of the most widely used AI tools in machine learning **[65].** It offers a comprehensive and flexible ecosystem of community tools, libraries, and community resources that allow researchers to advance in the field of machine learning, and developers to easily create and deploy applications that use this technology **[46].** In this project, it was the main tool to build and train the model.

- **Keras**

Keras is a powerful and easy to use free open source Python library for developing and evaluating deep learning models, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research. Keras was developed and maintained by Francois Chollet, a Google engineer using four guiding principles:

**Modularity**: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

**Minimalism**: The library provides just enough to achieve an outcome, no frills and maximizing readability.

**Extensibility**: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
**Python**: No separate model files with custom file formats. Everything is native Python.

- **Numpy :**

Numpy is an open source library associated with the Python language. It is very useful for performing mathematical and statistical operations in Python. It works great for multiplying matrices or multidimensional arrays **[Web 45].**

- **OS :**

Python's OS module provides functions for interacting with the operating system. The operating system is part of Python's standard utility modules. This module provides a portable way to use operating system dependent features. The "os'' and "os.path" modules include many functions for interacting with the filesystem **[Web 47,62].**

- **CV2 (OpenCV) :**

Open Source Computer Vision Library implements a variety of tools for image interpretation **[25].** It is compatible with the Intel Image Processing Library (IPL), which implements low-level operations on digital images. It is a free graphics library, specialized in real-time image processing with matrix basic structure mainly implementing algorithms for calibration techniques, feature detection and tracking, shape and motion analysis, 3D reconstruction, segmentation and object Recognition, binarization, filtering, image statistics, pyramids **[81].**

- **Visual Studio Community**

Visual Studio Community for windows is a free, popular and powerful integrated development environment (IDE) for programing in python among other languages **[82].** It offers all python supported libraries. Microsoft Python extension makes VS Code an excellent Python editor, and works on any operating system with a variety of Python interpreters. It leverages all of VS Code's power to provide auto complete and IntelliSense, linting, debugging, and unit testing, along with the ability to easily switch between Python environments, including virtual and conda environments **[83].**

### 1.2 Importing the dataset:

In this project, we will use the image dataset, it is a set of images; with and without mask from various sources. In this stage, images in different sizes and/or formats are imported.

### 1.3 Data pre-processing:

The pre-processing of the data is a very important step which consists in transforming the smile formats of the images into 2D image format in order to be able to process them.

#### 1.3.1 Data separation (Split folder):

From this stage, the model goes through training and test and validation phases as discussed in the conception section.

| Layer (Type) | Size | Setting |
|---|---|---|
| CONV2D | 64 | Activation : Relu |
| MaxPooling2D | 7 | |
| Flatten | | |
| Dense | 128 | Activation : Relu |
| Dropout | 4 | |

| | | |
|---|---|---|
| Dense | 2 | Activation : Softmax |

Table 2: Summarized CNN model

## 2. Experimental study

To choose a base model, we had to consider the following state of art. Let's see the results of different classifiers with ADAM, ADAMGRAD and SGD optimizers.

Table 3: Results of the proposed system with MobileNetV2 classifier

| **Classifier** | **Epochs** | **Train/test size** | **Optimizer** | **Train loss** | **Train accuracy** | **Test loss** | **Test accuracy** |
|---|---|---|---|---|---|---|---|
| MobileNetV2 | 100 | 80/20 | ADAM | 0.0090 | 0.9964 | 0.0071 | 1.000 |
| | | | ADAGRAD | 0.2454 | 0.9148 | 0.1811 | 0.9819 |
| | | | SGD | 0.1549 | 0.9502 | 0.0216 | 0.9855 |

MobileNetV2 yields high accuracy with low storage space and running time **[17]**. Generally, it is observed that the performance of the model increases as it goes on becoming denser. MobileNetV2 does not require much depth as compared to other networks such as Inception-v3, DenseNet201, ResNet50, VGG19 so we built a system using ADAM optimizer and MobileNetV2 network as it is light weight and efficient to train [**49,43,54,39**]. In our architecture, we are going to train our model with different types of datasets because it is been observed that images captured in real-time. Images captured through CCTV cameras are wider in size and also with low resolution **[8].** We trained the MobileNetV2 model with different classes of datasets like low-resolution images, wider images.

To choose a base model, we evaluated the measurements such as precision, accuracy and recall and chose the MobileNetV2 architecture with the best exposure having an accuracy of **99%** and a **98%** recall. It is furthermore computationally efficient by using MobileNetV2, which simplifies the introduction of the model in the inserted frames **[22-33]**. By doing so, the proposed framework is able to correctly detect faces while precisely segmenting each face in an image with lucrative results.

## 3. Experimental Results

We have upto this point noted that our model works well under good light intensity. Bellow are some screeshots of the results obtained from both mask detection and facial recognition with ata the moment work independently.


Fig 16. Person in Mask.


Fig 17. Person without Mask.

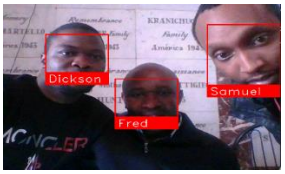



Fig 18. Known people in the same window.


Fig 19. Unknown person.

Fig 20. Person in Mask.


Fig 21. Known people in the same window.


Fig 22. Known person without mask


Fig 23. Known person without mask

Fig 24. Person mask 80%

Fig 25. Person without mask 100%

Fig 26. Person with mask

Fig 27. Person without mask 97%

### 3.1 Discussion of result

Figure 16 shows the output of the person captured in mask. Figure 17 is showing the output for the person captured without mask and is a known face. In Figure 18, the bounding box with display the identity of three people at once (known). In Figure 19, the bounding box with written "UNKNOWN" on its top is shown for a person captured without mask and is unknown in the department/organization. It should also be noted that on top of the bound box, it also displays a percentage of the face covered to the ratio of the region of interest (ROI) **[19].** Note the percentage difference between figure 16 & figure 20 and figure 26 & figure 27.

**Note:** Currently, face mask detection and facial recognition are working independently.

## 4. Evaluation (discuss the performance of the model)

### 4.1 Loss

The loss function quantifies the effectiveness or inability of the predictor in classifying the input data points in the data set. As stated earlier, the lower the loss, the better the classifier succeeds in modelling the relationship between the input data and the output class labels (Rosebrock, 2017) **[35].** The loss is cumulative per epoch. At the start of each epoch, the loss is zero. For each loss calculation the loss is added to the loss metric on the chart which is seen over time in the results plotted in figure 14, is the total loss (**0.03**) of training and validation which (usually) decreases. This means that the network weights (in this case, the classifier head fully connected over the convnet) become more accurate.

### 4.2 Precision (Accuracy)

Training accuracy increases with each epoch (Figure 15), however, a model that performs well on training data does not necessarily perform well on data it has never seen before (generalization). The accuracy of learning reaches **99.5** and that of validation **99.6.** According to our graph obtained (figure 15), we can already predict that our model is on the right track during its training. View that the curve is exponential. We also notice that the training and validation curves are both in the same direction. This reflects very good learning.

### 4.3 Confusion Matrix

A confusion matrix is a predictive analysis tool matrix that measures the quality of a classification model. On the rows of the matrix, we find the real classes **[64]**. The columns show the forecasts calculated by the model. It displays and compares the actual values with the values predicted by the model. In the context of machine learning, a confusion matrix is used as a metric to analyse the performance of a machine learning classifier on a data set. A confusion matrix visualizes parameters such as precision, accuracy, specificity, and recall. In our case, there are **1428** images where **703** belong to class 0 (Mask), and **725** to class 1 (No Mask).

```
#Matrix de confusion
matXconf = sklearn.metrics.confusion_matrix(testB.argmax(axis=1), predictionIndexes)
count = sum(sum(matXconf))
valid = (matXconf[0, 0] + matXconf[1, 1]) / count
hypersens = matXconf[0, 0] / (matXconf[0, 0] + matXconf[0, 1])
homogeneity = matXconf[1, 1] / (matXconf[1, 0] + matXconf[1, 1])

print(matXconf)
print("accuracy: {:.4f}".format(valid))
print("savoir_faire: {:.4f}".format(hypersens))
print("homogeneity: {:.4f}".format(homogeneity))

matX= sklearn.metrics.ConfusionMatrixDisplay(matXconf)
matX.plot()
```

Fig 28. Matrix of confusion function

- ✓ The model detected **730** images in class 0 (Mask), and **695** in class 1 (No Mask).
- ✓ The model correctly detected **722** images of class 0 (Mask), and **697** images of class 1 (No Mask).
- ✓ One can read the correctly predicted data by looking at the main diagonal of the matrix and the wrongly predicted on the lesser diagonal.
- ✓ The model predicted **6** images in class 1 (No Mask) while these are class 0 (Mask) data. We call it **false positives**.
- ✓ The model predicted **3** samples in class 0 (Mask) when it was class 1 data (No Mask). Here we are talking about **false negatives**.

| **N= 1376** | Predicted **No** | Predicted **Yes** | Out of |
|---|---|---|---|
| **No Mask** | 697 | 6 | 703 |
| **Mask** | 3 | 722 | 725 |
| Out of | 700 | 728 | **N=1428** |

Table 4: Matrix of confusion

# 5. Conclusion.

In this chapter we have presented and discussed the results obtained from the experimental study conducted to evaluate the performance of the proposed CNN face mask detection and facial recognition **[69]**. We have discusses the detailed training and testing phases as well as the accuracy and loss function as well as the confusion matrix. From the experiment performed in this work we can say that the results suggest that using high resolution images from a large dataset for training leads to good results **[71-73]**.

# General Conclusion and perspectives

We have demonstrated a real-time system of 2 models for the detection and recognition of people wearing a medical masks using CNN in real time using bounding boxes. We chose the MobileNet architecture for its lightness because our goal was to make a real-time mask recognition application that can be deployed on relatively weak systems: MobileNet can even be deployed on a Raspberry Pi and run in real time **[75,81]**. We proposed a pipeline as follows: We actively search the video input to find a face with the ResNet architecture and each time we find a face we launch our classifier which we have realized with the Transfer Learning based on a model pre-trained on the ImageNet database and which we coupled to our own networks and trained on the Face-mask-detection dataset. Our application can be deployed in public places to detect people who are not wearing a mask, which during these times is a threat to public health and helps the spread of this deadly virus. We are confident that our app can help fight the spread of COVID **[70].**

We found that CNN architectures are able to learn powerful features from a small amount of a dataset label that go far beyond the methods based on performance features **[25,63]**. Our training results indicate that performance is very sensitive to architectural details and hardware configuration, in especially with the GPU processor. However, having used the CPU processor for training, the testing results were not different **[27]**.

We also have concluded that in face recognition, CNN needs images of high resolution to extract important and unique features during training or else the model will be biased. In this project we used a 4K resolution.

This model can be implemented with surveillance cameras in places like malls, airports, banks, organizations, restaurants, departments, recreation areas, and other high traffic public places and crowded to screen people in general and prevent the spread of infection by checking who is following the essential rules and who is not **[3]**.

The main benefit of this project is, as the user will regularly instruct people without masks to wear it, the chances of people getting infected by the virus will reduce hence lowering the curve of global infection rate.

The project is based on Deep Learning for which the model is trained using Keras, Tensorflow, OpenCV and MobileNetV2 and for face recognition, face_recognition module is used. In future improvement the performance of Detection and Recognition pf faces wearing a mask system can be done using Segmentation instead of Bounding Box technique **[17].** In future we will try to add known face datasets using real-time technique, so that organizations with large number of people can also implement it. We are going to make the project more efficient by training the model in such a way that it can recognize and detect face in any angle. In addition we intend to combine facial recognition and mask detection to work simultaneously in one program and be able to generate notifications (in an excel sheet) of names of people without masks to the admin.

# Références


[1] *C.Jagadeeswari, M.Uday Theja, Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers,2020*

[2] *Vinitha.V1, Velantina.V2, COVID-19 facemask detection with deep learning and computer vision,2020*

[3] N. Alyuz, B. Gokberk, and L. Akarun. 3-d face recognition under occlusion using masked projection. *IEEE Transactions on Information Forensics and Security*,8(5) :789(802, 2013

[4] *Wei Bu, Jiangjian Xiao, Chuanhong Zhou, Minmin Yang, Chengbin Peng," A Cascade Framework for Masked Face Detection", 2017*

[5] *Amit Chavda, Jason Dsouza, Sumeet Badgujar, Ankit Damani ," Multi-Stage CNN Architecture for Face Mask Detection ",2020*

[6] *Aniruddha Srinivas Joshi, Shreyas Srinivas Joshi, Goutham Kanahasabai, Rudraksh Kapil, Savyasachi Gupta," Deep Learning Framework to Detect Face Masks from Video Footage",2020*

[7] *Reny Jose," A Convolutional Neural Network (Cnn) Approach to Detect Face Using Tensorflow And Keras",2019*

[8] *Adithya K1, Jismi Babu2," A Review on Face Mask Detection using Convolutional Neural Network", 2019*

[9] *"An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network",2019*

[10] *Zhongyuan Wang, Guangcheng Wang, Baojin Huang, Zhangyang Xiong, Qi Hong, Hao Wu, Peng Yi, Kui Jiang, Nanxi Wang, Yingjiao Pei, Heling Chen, Yu Miao, Zhibing Huang, and Jinbi Liang," Masked Face Recognition Dataset and Application",2020*

[11] *Mohamed Loey, Gunasekaran Manogaran b,c , Mohamed Hamed N. Taha d ,Nour Eldeen M. Khalifa,," Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection",2020*

[12] *A. Kumar, A. Kaur, and M. Kumar," Face detection techniques: a review,"Artificial IntelligenceReview,vol. 52,no. 2, pp. 927– 948, 2019.D.-H. Lee, K.L.Chen, K.-H. Liou, C.-L. Liu, and J.-L. Liu, "Deep learning and control algorithms of direct perception for autonomous driving,2019*

[13] *Y. Fang, Y. Nie, and M. Penny, Transmission dynamics of the covid-19 outbreak and effectiveness of governmentinterventions: A data-driven analysis,"Journal of medical virology, vol. 92,no. 6, pp. 645–659, 2020.*

[14] *M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510- 4520, doi: 10.1109/CVPR.2018.00474.*

[15] *Xin, M., Wang, Y. Research on image classification model based on deep convolution neural network. J Image Video Proc. 2019, 40 (2019).*

[16] *Sultana, F., A. Sufian, and P. Dutta. "A review of object detection models based on convolutional neural network." arXiv preprint arXiv:1905.01614 (2019).*

[17] *Adam: A Method for Stochastic Optimization - Diederik P. Kingma, Jimmy Ba*

[18] *Image Classification Using Convolutional Neural Networks Deepika Jaswal, Sowmya.V, K.P.Soman [18] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch", arXiv:1411.7923, 2014.*

[19] *Deng, J., Guo, J., Ververas, E., Kotsia, I., and Zafeiriou, S., RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild, 2020, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 5202-5211, doi: 10.1109/CVPR42600.2020.00525.*

[20] *He, K., Zhang, X., Ren, S., and Sun, J., Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.*

[21] *Bosheng Qin and Dongxiao Li, Identifying Facemask-wearing condition using Image super-resolution with classification network to prevent COVID-19, Sensors 2020*

[22] Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Essen, Abdul A S. Awwal, and Vijayan K. Asari: The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches

[23] Amit Chavda, Jason Dsouza, Sumeet Badgujar, Ankit Damani, "Multi-Stage CNN Architecture for Face Mask Detection" from ResearchGate in September 2020.

[24] V. Nair and G. Hinton, Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

[25] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554

[26] G.N.Priya and R.W.Banu.Oclusion invariant face recognition using mean based weight matrix and support vector machine. Sadhana, 39(2):303{315, 2014}.

[27] Fukushima, Kunihiko. "Neocognitron: A hierarchical neural network capable of visual pattern recognition." *Neural network 1.2 (1998): 1119-130.*

[28] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

[29] Krizhevsky, A., Sutskever, I, and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106-1114, 2012.

[30] Zeiler, M D. and Fergus, R. Visualizing and understandimg convolutional networks. CoRR, abs/1311.2901,2013. Published in Proc. ECCV,2014.

[31] Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." *arXiv preprint arXiv:1412.6806* (2014)

[32] He, Kiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[33] Szegedy, Christian, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, inception-resnet and the impact of residual connections on learning." *arXiv preprint arXiv:1602.07261* (2016)

[34] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[35] SImonyan, Karen, and Andrew Zisserman. "deep convolutional networks for large-scale image recognition" arXiv preprint arXiv:1409.1556(2014).

[36] Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400* (2013)

[37] Huang, Gao, et al. "Densely connected convolutional networks." *arXiv preprint arXiv:1608.06993* (2016).

[38] Ahn, Byeongyong, and Nam Ik Cho. "Block-Matching Convolutional Neural Network for Image Denoising." *arXiv preprint arXiv:1704.00524 (2017).*

[39] Ma, Shuang, Jing Liu, and Chang Wen Chen. "A-Lamp: Adaptive LayoutAware Multi-Patch Deep Convolutional Neural Network for Photo Aesthetic Assessment." *arXiv preprint arXiv:1704.00248*(2017).

[40] Cao, Xiangyong, et al. "Hyperspectral Image Segmentation with Markov Random Fields and a Convolutional Neural Network." *arXiv preprint arXiv:1705.00727* (2017).

[41] de Vos, Bob D., et al. "End-to-End Unsupervised Deformable Image Registration with a Convolutional Neural Network." *arXiv preprint arXiv:1704.06065* (2017).

[42] Wang, Xin, et al. "Multimodal Transfer: A Hierarchical Deep Convolutional Neural Network for Fast Artistic Style Transfer." *arXiv preprint arXiv:1612.01895* (2016).

[43] Babaee, Mohammadreza, Duc Tung Dinh, and Gerhard Rigoll. "A deep convolutional neural network for background subtraction." *arXiv preprint arXiv:1702.01731* (2017).

[44]Hou, Jen-Cheng, et al. "Audio-Visual Speech Enhancement based on Multimodal Deep Convolutional Neural Network." *arXiv preprint arXiv:1703.10893* (2017).

[45]Xu, Yong, et al. "Convolutional gated recurrent neural network incorporating spatial features for audio tagging." *arXiv preprint arXiv:1702.07787* (2017).

[46]Litjens, Geert, et al. "A survey on deep learning in medical image analysis." *arXiv preprint arXiv:1702.05747* (2017).

[47]Zhang, Zizhao, et al. "MDNet: a semantically and visually interpretable medical iamge diagnosis network" arXiv preprint *arXiv:1702.07787* (2017)

[48]Tran, Phi Vu. "A fully convolutional neural network for cardiac segmentation in short-axis MRI." *arXiv preprint arXiv:1604.00494*(2016).

[49]Tan, Jen Hong, et al. "Segmentation of optic disc, fovea and retinal vasculature using a single convolutional neural network." *Journal of Computational Science* 20 (2017): 70-79.

[50]Moeskops, Pim, et al. "Automatic segmentation of MR brain images with a convolutional neural network." *IEEE transactions on medical imaging* 35.5 (2016): 1252-1261

[51] Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs." *arXiv preprint arXiv:1605.05273* (2016).

[52] Jia, Xiaoyi, et al. "Single Image Super-Resolution Using Multi-Scale Convolutional Neural Network." *arXiv preprint arXiv:1705.05084* (2017)

[53]Alom, Md Zahangir, et al. "Optical beam classification using deep learning: a comparison with rule-and feature-based classification." *Optics and Photonics for Information Processing XI*. Vol. 10395. International Society for Optics and Photonics, 2017.

[54]Alom, Md Zahangir, et al. "Object recognition using cellular simultaneous recurrent networks and convolutional neural network." *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

[55]Alom, Md Zahangir, et al. "Handwritten Bangla Character Recognition Using The State-of-Art Deep Convolutional Neural Networks." *arXiv preprint arXiv:1712.09872* (2017).

[56]Arulkumaran, Kai, et al. "A brief survey of deep reinforcement learning." *arXiv preprint arXiv:1708.05866* (2017)

[57] Mnih, Volodymyr, et al "Human-level control through deep reinforcement learning " *Nature* 518.7540 (2015): 529-533.

[58] Mnih, Volodymyr, et al. "Playing Atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013)

[59] Raza Ali, Saniya Adeel, Akhyar Ahmed, Md Hasan Shahriar, Md Shohel Mojumder, Dr. Christoph Lippert, “Face Mask Detector”, in ResearchGate July 2020.
[60] Mohammad Marufur Rahman, Md. Motaleb Hossen Manik, Md. Milon Islam, Saifuddin Mahmud, JongHoon Kim, “An Automated System to Limit Covid-19 Using Facial Mask Detection in Smart City Network”, in IEEE.

[61] Amit Chavda, Jason Dsouza, Sumeet Badgujar, Ankit Damani, “Multi-Stage CNN Architecture for Face Mask Detection” from ResearchGate in September2020.
[62] Hong Zhao, Xi-Jun Liang, and Peng Yang, “Research on Face Recognition Based on Embedded System”, in Hindawi Publishing Corporation, 2013.

[63] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake,"Real- Time Human Pose Recognition in Parts from Single Depth Images," Computer Vision and Pattern Recognition, 2011.

[64] Marco Grassi, Marcos Faundez-Zanuy, “Face Recognition with Facial Mask Application and Neural Network”, in Computational and Ambient Intelligence, 2007, Volume 4507.

[65] Naveenkumar Mahamkali, Vadivel Ayyasamy, “OpenCV for Computer Vision Applications” br ResearchGate in March 2015.

[66]https://www.mygreatlearning.com/blog/real-timeface-detection/.
[67]https://towardsdatascience.com/understanding-facedetection-with-the-viola-jones-objectdetectionframework-c55cc2a9da14.
[68]https://rea/fvb lpython.com/traditional-face-detection-python/

[*69] Vinitha.V1, Velantina.V2, COVID-19 facemask detection with deep learning and computer vision,2020.*
*[70] Adithya K1, Jismi Babu2,” A Review on Face Mask Detection using Convolutional Neural Network”, 2019.*
[71] C.Jagadeeswari, M.Uday Theja, Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers, 2020.
[72] Martín ABADI et al. *TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL : https://www.tensorflow.org/.

[73] Orly Enrique APOLO-APOLO et al. « A cloud-based environment for generating yield estimation maps from apple orchards using UAV imagery and a deep learning technique ». In : *Frontiers in plant science* 11 (2020), p. 1086.
[74] François CHOLLET et al. *Keras*. https://github.com/fchollet/keras. 2015.

[75]CNN architectur.Retrieved from https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Conv_layer. png

[76] Pooling layer.Retrieved from weblink: https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Max_poolin g.png

[77] functionality of cnn. retrieved from Dabeer, S., Khan, M. M., & Islam, S. (2019). Cancer diagnosis in histopathological image: CNN based approach. Informatics in Medicine Unlocked, 100231.

[78] https://github.com/kjw0612/awesome-deep-vision

[79] Linear activation(ReLU). Retrieved from https://www.neuraldesigner.com/blog/perceptron-the-main-component-of-neural-netw orks#Linear%20activation

[80] Shin HC et al (2016b) Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging 35(5):1285–1298

[81] Claus NEBAUER. « Evaluation of convolutional neural networks for visual recognition ». In: *IEEE transactions on neural networks* 9.4 (1998),

[82]https://medium.com/two-minute-tech/getting-started-with-microsoft-visual-studio-community-for-python-programming 4b862107907

[83] https://code.visualstudio.com/docs/languages/python

# Annexe A

## 1. Applications of CNNs

**i). CNNs for solving Graph problem**

Learning graph data structures is a common problem with various different applications in data mining and machine learning tasks. DL techniques have made a bridge in between the machine learning and data mining groups. An efficient CNN for arbitrary graph processing was proposed in 2016 **[15].**

**ii). Image processing and computer vision**

CNN is used on different application domains including image classification, detection, segmentation, localization, captioning, video classification, major tasks like facial recognition, and image recognition and object identification using Single image super-resolution using CNN methods, Image de-noising using block-matching CNN, Photo aesthetic assessment using A-Lamp: Adaptive Layout-Aware Multi-Patch Deep CNN **[36,32, 5**]. DCNN for hyper spectral imaging for segmentation using Markov Random Field (MRF) **[9]**. Image registration using CNN. The Hierarchical Deep CNN for Fast Artistic Style Transfer Background segmentation using DCNN. Handwritten character recognition using DCNN approaches. Optical image classification using deep learning approaches. Object recognition using cellular simultaneous recurrent networks and convolutional neural network.

**iii). Speech processing**

CNN methods are also applied for speech processing: speech enhancement using multimodal deep CNN, and audio tagging using Convolutional Gated Recurrent Network (CGRN) **[32].**

**iv). CNN for medical imaging**

A good survey on DL for medical imaging for classification, detection, and segmentation tasks **[19]**. MDNet, which was developed for medical diagnosis with images and corresponding text description. Cardiac Segmentation using short-Axis MRI. Segmentation of optic disc and retina vasculature using CNN. Brain tumour segmentation using random forests with features learned with fully convolutional neural network (FCNN).

## 2. Challenges facing Convolutional Neural Networks (CNN)

They work quite well for some use cases like classifying images and segmentation (figuring out which group of pixels belong to and entity). But there are problems, for instance:

**i).** You can trick a neural network to think an image is something else by using some clever tricks. This can be a big problem for security and validation purposes. Below we see that if we add some carefully crafted noise to the image not changes are visually apparent to a human but can completely trick the CNN.

**ii).** Typical convolutional networks can be tricked by some rotations. If you did not train you network on rotations of an image it might classify two images that are exactly the same as different.

**iii).** It could be that the network only looks for elements being present in the image and does not care about their relative location. Hinton's team has proposed capsule networks in 2017 to eliminate this issue **[21]**. A CNN will classify both images as a face with similar confidence.

**iv).** There are some contexts that are hard to decipher humans know all about context, but CNNs need to be explicitly trained, which can be an exhaustive process (like in the case of interpreting feelings).

# Annexe B

## 1. Challenges

This study is very limited and requires a broader context of specification of images or videos. For future work such as face directions from 180 degrees of horizontal direction or vertical camera distances, higher resolution for cases industry real use cases, we will consider improving the speed of the proposed method by training on larger sets of data and then we can use it in real applications **[43-45]**.

Identifying mask misuse, difficulty in training these models, given the variety of camera angles in images and mask types, limitation of a large data set for the formation of the two categories; with and without a mask.

However, we have not yet been able to detect other types of problems, such as ("incorrect side adjustment“, “side fit” and "bezels below").

For these cases, it is necessary to find an alternative approach or to increase the number of training samples. For this reason, for errors of lateral misplacement like gum crossing, it would be good to teach the system to detect it with a lateral image.

# Résumé

To curb the spread of Covid19, we have demonstrated a Real-Time model that automatically detects whether a person has a mask on and correctly or not and possibly note names of people without a mask. It employs a Convolutional Neural Network (CNN), to draw bounding boxes (red/green) around people's faces, depending on whether a person is wearing a mask or not and bring a proportion to the Region Of Interest (ROI). It can be embedded in an institution where the admin can take action accordingly on people recognized without face masks thus reducing the chances of People getting the virus infection in that institution, by protecting everyone in the circle. It is a project based on Deep Learning and Artificial Intelligence using OpenCV, Keras/Tensorflow and MobileNet.

**Key words**. [CNN, OpenCV, Deep Learning, Face Mask Detection, Face Recognition, Keras, Tensorflow, MobileNet]