MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

ABDELHAMID IBN BADIS UNIVERSITY - MOSTAGANEM

Faculty of Exact Sciences and Computer Science

Department of Mathematics and Computer Science

Sector : Computer Science

END OF STUDIES THESIS

For obtaining the Master's Degree in Computer Science

Option : **Information Systems Engineering**

Presented By :

Benhenneda Abdelmadjid Amine

Larbi Youcef Sid Ahmed

THEME :

One to many faces detection on mask-wearing individuals

Supported on : 19/06/2021

In front of the jury composed of :

Dr.Henni Fouad	MCB	Mostaganem University	President
Dr.Hamami Dalila	MCB	Mostaganem University	Examinator
Dr.Bentaouza Mérièm Chahinez	MCB	Mostaganem University	Supervisor

Acknowledgments

Benhenneda Abdelmadjid Amine's Acknowledgments

First of all, I thank the good Lord who brought me help to do this job well. I also thank my parents and my brothers who were always by my side ... Encouraging me throughout the years, nothing would have been possible without their lifelong support.

To my grandparents who always believed in me and pushed me further in pursuing my dreams and line of work.

To my passed away friend, Oussama "Chikoko" Elazhar Djoudi may his soul rest in peace, and to the "GG EPIC" group for being there through the hardest of times.

I would also like to express my gratitude to all the team at the Neo-Fintech startup that supported me throughout the year and were so keen on helping.

I would also like to express the honor to the members of the jury that have accepted and took the time to evaluate our work, and to our supervisor that poured so much time and efforts in guiding us.

At the end, I thank all people who believed in me to realize this work.

Larbi Youcef Sid Ahmed's Acknowledgments

I would like to start by thanking Almighty God for giving me the opportunity to do what I am passionate about.

To my parents, whom gave me unconditional love and gratitude and helped me on my journey and supported my choices no matter what.

To my grandmother, aunts, and uncles who keep giving me life long advice.

A big thank you to the teachers for helping us and guiding us throughout this project.

Finally, I want to express my gratitude towards anyone who helped and believed in me since day one.

ملخص

هذا المشروع عبارة عن بحث ببليوغرافي حول اكتشاف الوجه للأشخاص الذين يرتدون أقنعة الوجه. يتمثل اكتشاف الوجه في تحديد وجه موجود في صورة ولكن مع ارتداء قناع واقٌ ، يصبح هذا الكشف صعباً لأن القناع سيخفيٍ أجزاء مهمة من الوجه مثل الأنف والفم. تستخدم أنظمة الدوائر التلفزيونية المغلقة شخصاًا لكثير لالتقاط وجوهٍ عالية الجودة دون معرفة المشاركين من أجل التحكم في السلامة أو حتى السلامة. بناءً على ذلك ، توصلنا إلى نتيجة استخدام أساليب التعلم الآلي لتحقيق أقصى قدر من الدقة في اكتشاف الوجوه المقنعة.

الكلمات الدالة : كشف الوجه ، الذكاء الاصطناعي ، التعلم العميق.

Résumé

Ce projet est une recherche bibliographique sur la détection faciale de personnes portant des masques faciaux. La détection de visage consiste à repérer un visage présent sur une photo mais avec le port de masque de protection, cette détection devient difficile à gérer car le masque va cacher des parties importantes du visage tel que le nez et la bouche. Les systèmes de vidéosurveillance utilisent one-to-many pour capturer des visages de qualité sans la connaissance de personnes concernées afin de contrôler la sécurité voire la sûreté. En se basant sur cela, nous sommes arrivés au résultat qui consistait à utiliser des méthodes d'Apprentissage Machine pour obtenir une précision maximale dans la détection des visages masqués.

Mots clés : Détection Faciale, Intelligence artificielle, , Apprentissage Profond.

Abstract

This project is a bibliographic research on the facial detection of people wearing face masks. Face detection consists in locating a face present in a photo but with the wearing of a protective mask, this detection becomes difficult to manage because the mask will hide important parts of the face such as the nose and the mouth. CCTV systems use one-to-many to capture quality faces without the knowledge of those involved in order to control safety. Based on this, we came to the result of using Machine Learning methods to achieve maximum accuracy in detecting masked faces.

Keywords : Face detection, Artificial Intelligence, Deep learning.

List of Figures

List of Tables

List of Abreviations

AI : Artificial Intelligence.

ML : Machine Learning.

CNN : Convolutional Neural Network.

ANN : Artificial Neural Network.

PCA : Principal Components Analysis.

Adaboost : Adaptive Boosting.

API : Application Programming Interface

NLP : Natural Language Processing

ReLU : Rectified linear activation function

PIL : Python Imaging Library

BS : Batch Size

DNN : Deep Neural Network

BAIR : Berkeley Artificial Intelligence Research

Contents

General introduction

As of 2021, the COVID-19 pandemic is still very active and everyone should protect themselves at all cost. The first and most important tool of protection is the mask that we wear daily now. While becoming such a trivial object that many of us has gotten used to, sometimes, masks are removed by many individuals, and this can cause health hazard and safety issues.

In the subject of computer engineering, a method that could help us treat this matter is face detection. Ideally, there would be a method that could automatically tell apart faces wearing masks and faces that are not. Using artificial intelligence, it is indeed possible to get an image's information and detect what this image contains. In order to understand how this works, we will need to know about the different artificial intelligence notions and concepts, their applications and where they facial detection in used. Also, The face detection technology is relatively new, and can detect faces from images. We will go through its history's highlight and how it came to be. The problem that arises then is, the impossibility to detect if an individual is wearing a mask with artificial intelligence, this hindrance can stop maximum protection and prevention against the pandemic. And to treat this problem, we are going to dive in the subject of face detection on masked individuals. As we advance, we'll treat the different methods used, and then compare them based on their accuracy and their efficiency on coming up with a solution for this problem.

Finally, we will try to come up with a solution of our own that will be a trial to get the best and most optimized method to detect whether a human face has a mask on or not.

To summarize, this paper, we will proceed to do a scientific research on the subject of face detection on mask-wearing individuals. This research will contain four (4) chapters :

- The first chapter will contain definitions of artificial intelligence notions along with a summary of the face detection's history.
- The second chapter's goal is explaining the methods already used concerning face detection on mask-wearing individuals. A comparison between those methods will be made in terms of accuracy and the method's reliability.
- The third chapter will be a theorization of the method that will be used in order to detect masked faces.
- The fourth chapter represents an attempt in implementing what we have covered in the third chapter with an emphasis on the different outcomes that yielded through trial and error with as a final outcome, a model we think is optimal and a functioning one-to-many live feed detection.

Chapter I

Facial detection

I.1 Introduction

First and foremost, in order to go through our research which consists of looking into face detection on mask-wearing individuals, we have to understand what face detection actually is, and see what steps it went through along with its core concepts.

Facial detection has a lengthy history, that goes from the birth of the concept itself to its usage in our daily life. Knowing that, we'll go into its history, detailing its rise and how it came to be in the following parts. Naturally, in order for us to understand how facial detection works, we must also categorize the different methods to know which is the best to proceed with. Along with the theoretical concepts of artificial intelligence that will be used.
Thus, in this chapter, we will go through the major highlights of facial detection history while categorizing its methods and giving the prime concepts of artificial intelligence where face detection can be used.

I.2 The history of facial detection

I.2.1 What is facial detection ?

Facial detection, in technology, means solely detecting if a face is present in an image (or video) or not. It falls into an application of object-class detection. Its concept consists in detecting the face's different shapes (mouth, nose, lips, ears, eyes) to tell apart a face from another structure present in an image.

I.2.2 The first occurrences of a facial detection concept

Automatic facial recognition is a relatively new concept, the first semi-automated facial recognition system was developed in the 1960s and it's earliest pioneers of facial recognition were Woody Bledsoe, Helen Chan Wolf and Charles Bisson , it required the administrator to locate the eyes, ears, nose and mouth in the photo and enter the calculated distances and ratios to a common benchmark, which were then compared to the benchmark data [1].

Fast forward a decade after that, Goldstein, Harmon and Lesk used 21 specific markers such as hair color and lip thickness to automate recognition. The problem with these first two solutions is that the measurements and locations were calculated manually [2].

I.2.3 PCA : A major step

Then in the 1980s Kirby & Sirovich applied principal component analysis (PCA), a standard technique of linear algebra, that is used for reducing the number of variables in face recognition. In PCA, every image in the training set is represented as a linear combination of weighted eigenvectors called eigenfaces. These eigenvectors are obtained from the covariance matrix of a training image set. This was in some way considered an important step because it showed that at least a hundred values were needed to properly and accurately encode an aligned and normalized image.

However in 1991 Turk & Pentland discovered that when using the Eigenfaces technique (PCA), the residual error can be used to detect a face in an image [3] as seen in Figure I.1. A discovery that enabled automatic facial recognition in real time and was somewhat limited by environmental factors, it nevertheless created a significant interest in promoting the development of automatic facial recognition technologies. This technology was put to the test in January 2001 during the final of the SUPER BOWL American football championship by capturing surveillance images and then compared to a digital database [4].

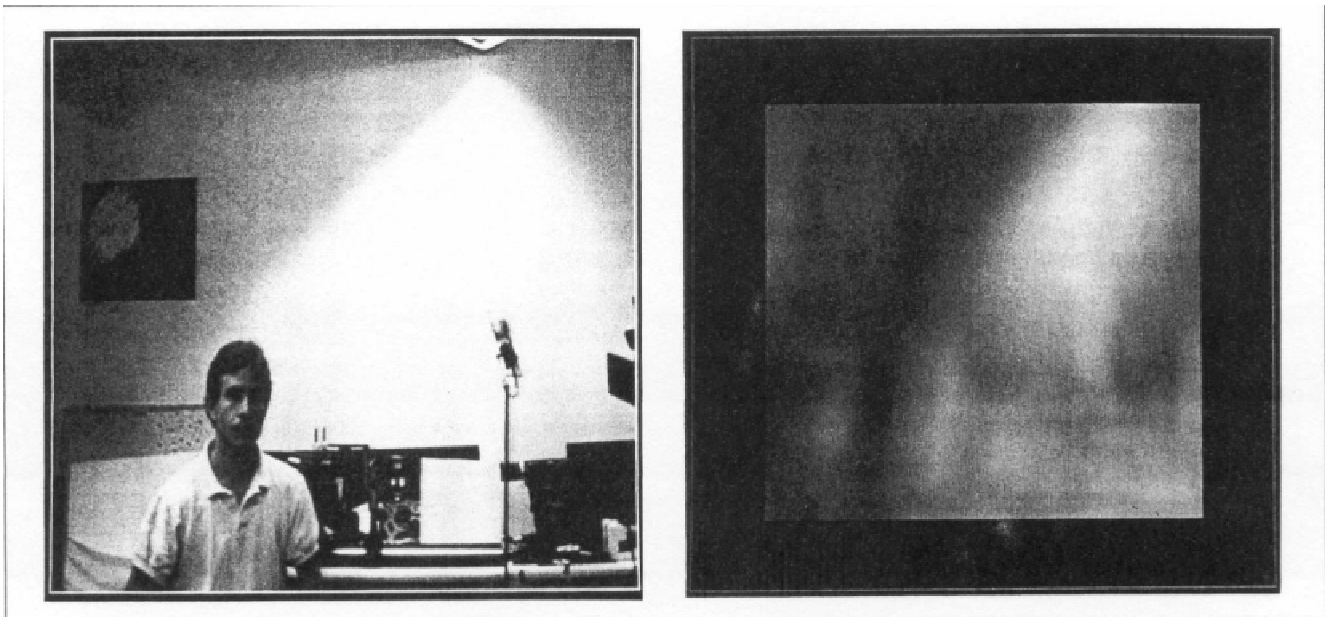


Figure I.1: An image and its corresponding face map, where low values indicate the presence of a face [3].

The commercial aspect of the facial identification and reconnaissance niche started to gain traction when the Defence Advanced Research Projects Agency (DARPA) rolled out it's face recognition program called "FERET", as seen in Figure I.2. The goal of the FERET program was to develop automatic face recognition capabilities that could be employed to assist security, intelligence, and law enforcement personnel in the performance of their duties [5].

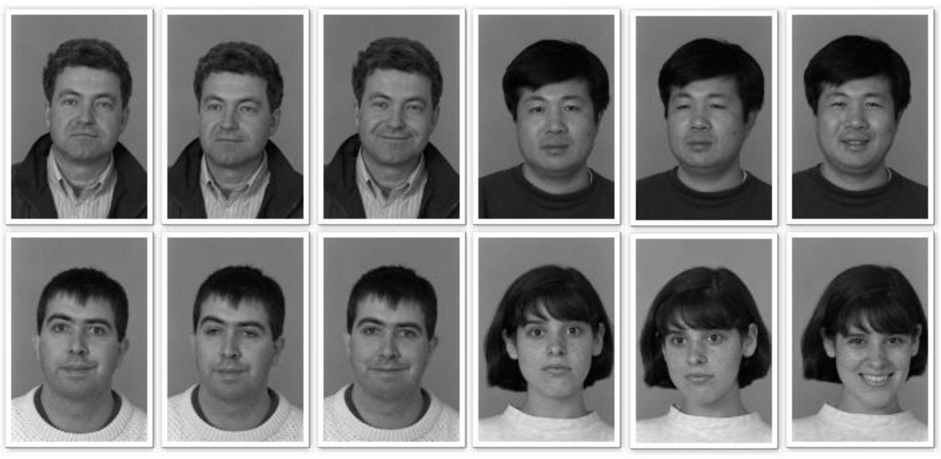


Figure I.2: FERET face database sample [5]

I.2.4 The Viola-Jones revolution

In 2001, two researchers by the name of Paul Viola and Micheal Jones joined forces to conceive the most successful face-detection method. The Viola-Jones method consisted in combining different techniques, each in one step :

1. **The Haar Feature Selection** Named after Alfred Haar, a mathematician originating from Hungary. This technique consists of selecting the most prominent face features as edge, line and four-sided features. Thanks to lightning, and edges characteristics, things such as the nose and the eyebrows can be selected and sorted from image to image.

2. **The Integral Image** The integral image is computing a value for each pixel in an image by adding the sum of the pixel values above and to the left of the said-pixel. This is a method widely used in Computer Vision and makes the calculation of a specific feature with Haar way less intensive, with saving much more time in the process.

3. **Adaptive Boosting** Adaptive Boosting (Adaboost) is a learning alogirthm that helps in taking apart false positivies and true negatives in the images given, thus giving way more accuracy to the method.

4. **Cascading Classifiers** At last, the final optimizing that has to be done in the Viola-Jones method is the Cascading Classifiers, where the features are classified and sorted in many subwindows, and those get compared to other subwindows and so on. When a subwindow doesn't have the feature searched for, it is simply discarded. This comparison is done at a very fast rate when comparing two sole subwindows, but when there are lots of features, it can take a lot of time.

In the Early 2000s the National Institute of Standards and Technology (NIST), an agency of the Department of Commerce for United States responsible, among other things, for evaluating algorithms and define standards provided evaluations that were designed to help law enforcement agencies and the U.S. government with information necessary to determine the best ways to deploy facial recognition technology, FRVT 2000 consisted of two components: the Recognition Performance Test and the Product Usability Test. The Recognition Performance Test was a technology evaluation. The goal of the Recognition Performance Test was to compare competing techniques for performing facial recognition [6].

I.3 The different types of facial detection methods

After seeing how facial detection came to be, and where it is now, it seems natural for us to sort its methods in order to know where to go next in our research. A survey by Yang, Kriegma, and Ahuja categorized face detection approaches in four big categories [7] :

I.3.1 Knowledge based

These methods are based on the knowledge of the various elements which constitute a face and the relations which exist between them. Thus, the relative positions of various key elements such as the mouth, nose and eyes are measured and then used for the face or non-face classification. The problem with this type of method is that it is difficult to accurately define a face in a single way. If the definition is too detailed, some faces will be missed, while if the description is too general, the false positive rate will increase

I.3.2 Feature based

These approaches are used mainly for face localization. The algorithms developed aim to find the existing structural features even if the pose, point of view, or lighting condition changes. Then they use these invariable characteristics to locate faces. We can cite two families of methods belonging to this approach : The methods based on the color of the skin, and the methods based on the characteristics of the face, they consist in locating the five characteristics (two eyes, two nostrils, and the nose / lip junction) to describe a typical face. As seen in the Viola-Jones method, combining the feature based technologies to the other can give promising results.

I.3.3 Template Matching

Face detection is done by learning standard examples of faces or frontal images containing faces. The procedure is done by correlating the input images and the recorded examples (templates) and the result gives the final decision of whether or not a face exists. There are 2 types of correlation depending on the type of templates: predefined face templates and deformable models. However, deformable templates have been proposed to deal with these problems.

I.3.4 Appearance based

The difference between this method and template matching is that the models are read from the training images which must be representative and made in different positions of the face. Generally the appearance-based approach is based on statistical analysis techniques (percentage of existence of models in the image) and machine learning to find significant characteristics.

I.4 Theoretic notions used in facial detection

After talking about the history of facial recognition, we're going to further explain the recent methods and theoretic notions used for general facial detection.

I.4.1 Neural Networks

1. **Artificial Neural Networks** Artificial Neural networks (ANN) are models that have share a similar structure with the brain's neural network, with connected layers of neurons. A neural network can learn from data, and can also predict future events.

2. **Convolutional Neural Networks (CNN)** A convolutional neural networks (CNN) is class of deep neural network, used mostly in image detection. The difference between ANNs and CNNs is that an CNN needs a lot of raw data in order to give an accurate result since it cannot detect features as easily as ANN. On the other hand, the CNN is much more accurate when the data is proliferate.

I.4.2 Machine Learning

Machine learning (ML) is an application of artificial intelligence (AI) that can give the possibility for a machine to learn and improve automatically through experience, without being specifically programmed to do that. Machine Learning is applied when classic algorithmic and software engineering reaches its limits. Its core concept consists of observing data thoroughly, be it through experience, or instruction. It mainly looks at recurring patterns of a given data and extracts it to detect it easily on every data given.

Machine Learning methods

Machine learning is generally classified into four categories : supervised, unsupervised, semi-supervised and reinforcement learning.

1. **Supervised Machine learning** Supervised algorithms use labeled information that have been learned before in order to predict future matters. It basically studies known training datasets and produces a function that will predict the outgoing values. The algorithms can also come up with new input data after enough training, it is also able to compare its outgoing values with the correct ones so as to achieve a higher success rate and modify the function accordingly.
2. **Unsupervised machine learning** In the unsupervised algorithms, the information is neither classified nor labeled. With this method, functions go on their own to try to extract a pattern and discover structures in the unlabeled data.
3. **Semi-supervised machine learning** As their name suggests, semi-supervised algorithms are between supervised learning and unsupervised, where a mix of both labeled and unlabeled data are used. It is typically used in order to improve the accuracy of learning.
4. **Reinforcement machine learning** Reinforcement machine learning algorithms interact and discover error or rewards. So as its name obviously suggests, it reinforces the function so the results are optimal.

Machine Learning and Face Detection

As we have seen earlier, PCA was one of the pioneer in facial detection algorithms, and it very well goes into the unsupervised algorithms of machine learning [7]. SVM (Support Vector Machines), and LDA (Linear Discriminant Analysis), supervised algorithms are also commonly used when treating the face detection matter.

I.4.3 The birth of Deep Learning

Deep learning's history can be traced back to 1943 when a computer was used to model the human brain using neural networks, however it was only until the 2000s that it started to gain some popularity, the particularity of these networks is that they are organized in layers and in a hierarchical manner.

Deep learning has managed to solve the problem of the feature construction algorithm, his idea is to skip the design stage of this algorithm by making a big neuron network with lots of layers as seen in figure I.3.

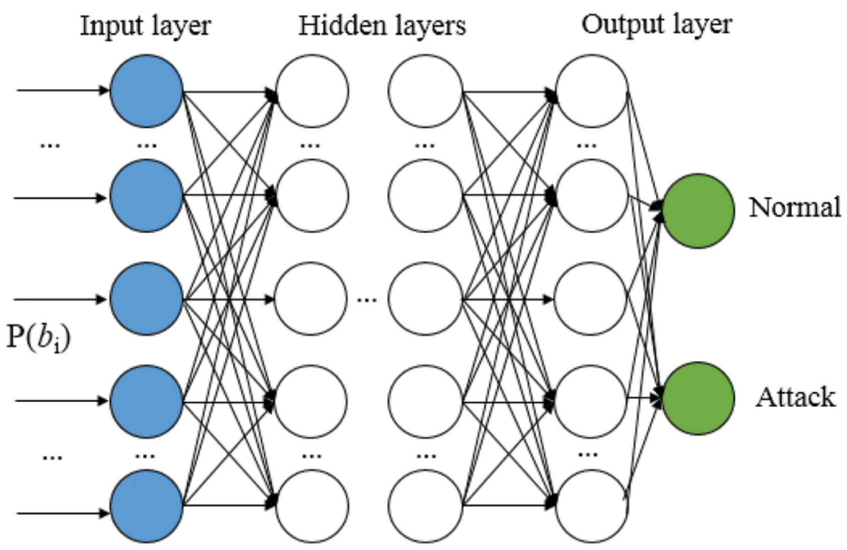


Figure I.3: Deep Learning operating principle. [8]

Although the learning phase was very difficult with a large network of neurons, the great pioneers of Deep Learning like Yann LeCun, managed to build algorithms which are able to determine quite complex relationships between inputs and outputs. So the little miracle that occurs with Deep Learning is that the lower layers of the neural network first seek to process the simplest elements present in a given data and the upper layers extract the more complex characteristics which will facilitate the learning phase. Deep learning is a great revolution in artificial intelligence. Large companies such as Facebook,Google,IBM..etc have been benefited greatly by this new technology and have decided to integrate it into many of their applications.

I.4.4 Deep Learning

Deep learning is a ML technique that resolves problem with a greater abstraction level. Deep Learning's concept is pretty straightforward : it uses many layers of neurons in order to let the machine learn known shapes from our data. The first layer interprets the image, or video as data and carries on learning the shapes in every layer. After the first layer, the algorithm goes through every layer until the last one. Once a system has been trained with enough images, we receive the model. We then proceed to test new data to check its accuracy.

I.5 Conclusion

Face detection is a broad concept with many nuances and loads of information. In this first chapter of our scientific research, we tried to make a clear summary of what is face detection, citing its history and its main methods. Naturally, it is needed for us to define theoretic notions that concern artificial intelligence so we can know where to put different technologies. After having a good grasp on the basics of face detection, we can carry on in the next chapter to explore methods used in the detection of masked faces, then we can try to compare them by accuracy and efficiency.

Chapter II

Face detection on mask-wearing individuals

II.1 Introduction

The current health crisis has generalized the compulsory wearing of a hygiene mask in the public space. As with road and urban surveillance systems, or face detection systems, the question of automated mask wearing control arises. From a technical point of view, a real revolution has taken place in recent years in particular in the field of object detection through deep learning and with a sufficient image database, it is possible to train networks of neurons to recognize the presence, position or even the type of objects in an image with excellent precision.

Once trained, some networks are also able to analyze in real time. Face masks have become an essential accessory to fight against the spread of the COVID-19 pandemic, thus the need to remotely and automatically identify individuals that follow this health protection fabric emerges as a necessity instead of an option. It is obviously faster, easier, and safer to monitor people walking through a space to verify they have face masks on than to manually do,meaning that the flexibility and easy deployment of the solution make it useful in various scenarios where multiple people pass by, such as transportation hubs, offices, warehouses..etc

In a scenario where face masks are mandatory or even recommended, this provides an easy way to monitor the situation or even remind people of the rules,but before we dive in how we're going to detect if a person is wearing a face mask or not we must first lay out what is the actual detection process looks like with it's different methods as well as portray the different tools that are available to us and that can be used in our research.

II.2 AI frameworks

Artificial intelligence frameworks have been the backbone for machine learning for a while now and their use represents a big leap into the masked faces detection spectrum. They enable the development of AI applications to be achieved more rapidly and conveniently. An example of these applications is neural networks or natural language processing(NLP). These frameworks serve as a blueprint for the whole enterprise of the artificial intelligence infrastructure.

This simplifies creation, implementation and the governance while also allowing each system to seamlessly work together. As frameworks are built on top of a robust AI architecture, any possible security breaches or quality management problems are then eliminated and it one of the main reasons they're considered a standard in the machine learning world.

II.2.1 Computer Vision frameworks

Custom vision is a service offered by Azure, a branch of cloud computing owned by Microsoft.It basically is an image recognition service that lets you build, deploy, and improve your own image identifiers. An image identifier applies labels (which represent classes or objects) to images, according to their visual characteristics [9] .

The basic premise that we'll use is simply uploading our dataset into the Azure owned branch, then train and test our model by having an accurate precision rate of the output. Finally we'd be able to optimize our model by choosing from several varieties of the Custom Vision algorithm that are optimized for images with certain subject material.

II.2.2 Deep Learning frameworks

TensorFlow is a machine learning library, with a toolkit that is able to solve extremely complex mathematical problems with ease. It allows researchers to develop experimental learning architectures and transform them into software.It also brings together a large number of Machine Learning and Deep Learning models and algorithms.

Table II.1: A comparison between AI Frameworks

Custom Vision	**Tensorflow**
cloud deployment comes out of the box (including a rest API)	Full control over your network and how you train it
Labelling tool to add data and label them	Manual model deployment
Difficulty to run our model locally/completely for free	Manual data management
difficulty to run our model locally/completely for free	difficulty to run our model locally/completely for free

II.3 Graphic libraries and face detection

Detection of face masks is a very difficult task for the current projected models of face detectors; this is often as a result of faces with masks having varied accommodations and numerous degrees of obstructions, and varied mask sorts. They're accustomed to facilitate self-focusing, the interaction between humans and computers,and managing image info Even when having such extraordinary and exceptional ends up in the prevailing face detectors, there's still high rising scrutiny within the development of additional advanced face detectors as for existing models, event analysis and video streams is usually a difficult job.

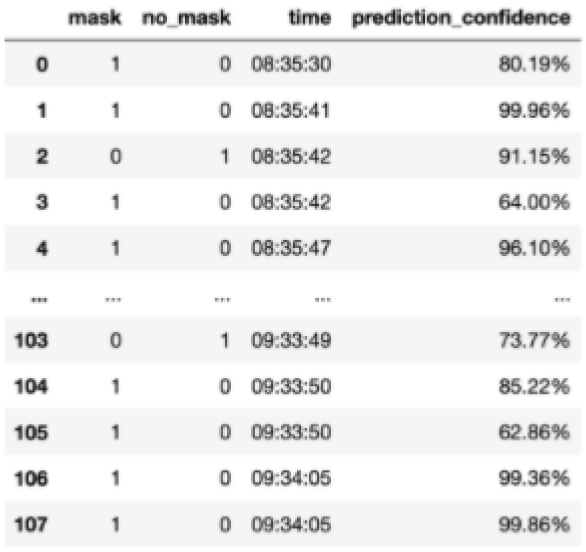
Many reasons were found for the poor action of existing mask detection models as compared to the conventional ones, two of them were because of the lack of appropriate datasets with properly disguised faces and automatic face detection,as well as the presence of masks on the face brings a definite quiet noise, that more deteriorates the detection method, thus the use of a library containing multiple algorithms can come in handy which currently is the case for OpenCV.

II.3.1 OpenCV

OpenCV (Open Computer Vision) is a graphics library. It specializes in image processing, whether for photo or video, OpenCV can be reused freely, in whole or in part, to be integrated into another project. It is this notion that makes OpenCV very popular and the basis of many image / video processing software.

This graphic library provides a set of over 2,500 computer vision algorithms, accessible through API(Application Programming Interface). This allows a lot of processing to be carried out on images (color extraction, detection of faces, shapes, application of filters, etc.). These algorithms are mainly based on complex mathematical calculations, mainly concerning the processing on the matrices (because an image can be considered as a matrix of pixels).

The premise of this method is creating a virtual environment where we'll install all of our necessary packages, then pick an object detection method such as Haar Cascade classifiers. We start by reducing the computation time by converting our set of images to grayscale, this step is carried out to also reduce the amount of noise an image/video may imit.



	mask	no_mask	time	prediction_confidence
0	1	0	08:35:30	80.19%
1	1	0	08:35:41	99.96%
2	0	1	08:35:42	91.15%
3	1	0	08:35:42	64.00%
4	1	0	08:35:47	96.10%
...	...	...	...	...
103	0	1	09:33:49	73.77%
104	1	0	09:33:50	85.22%
105	1	0	09:33:50	62.86%
106	1	0	09:34:05	99.36%
107	1	0	09:34:05	99.86%

Figure II.1: Prediction confidence with OpenCV for frontal face positions [14].

II.4 Masked faces dataset construction

Having a reliable dataset to train our model is a crucial step in detecting individuals wearing masks. This can be achieved by first collecting a set of images that can be found in the internet simply by researching keywords such as a face mask on google, or using various image providers such as unsplash. However we should aim towards adding images that do not exceed the 180 pixels in order to minimize the overall size of our dataset.

II.4.1 Need for filtration

After collecting the images containing face masks we'll start by filtering them by occlusion,this means that the image that contains only faces without the occlusion will be removed. Finally, once our images are now all filtered and ready we can move along by either manually cross validating our images or use an automatic provider such as custom vision, this latter can be done by simply annotating the service what we're looking for by inputting the image and teaching the algorithm what a masked face and is not.

In the phase of annotating our images, it is empirical that we use a set of perimeters that will overlay the attribute we're looking for, these will be as following :

- **The location of the face** This will allow us to label faces as valid or ignored,the location of each face will be annotated by a square,once detected we'll be looking for various variants such as the amount of blurring that is present or the noise the image may be emitting.
- **The location of the eyes** : For each face, the coordinates of eye centers are manually marked.
- **The location of the masks** : Face masks are usually worn in three different way, covering half of the face(chin to nose), worn from chin to mouth, or simply placed on the chin,this latter will also be annotated by a rectangle.
- **The facial orientation** : In our research we'll simply use images/streams with frontal faces only.
- **The occlusion degree** : In order to measure the occlusion degree we will divide the face into four major regions:the eyes, the nose, the mouth and as well as the chin.We can conclude three types of occlusion degrees from the major regions we've mentioned above,weak which means only one or two regions are visible (i.e chin nose), a medium occlusion (three regions) or a significant occlusion (which includes all of the four regions).
- **The type of mask the individual is wearing** : The model of masks can differ from person to person that is why it is necessary to consider that while picking our dataset and creating our model,we usually distinguish three types of masks, the simple surgical mask that comes in most cases with a pure color, a complex mask that can contain graphics or logos, and a hybrid one which is a combination of the two sorts of masks or one in all the aforesaid mask sort with eyes occluded by the glasses.

The purpose of computer vision is to enable machines to see and perceive the physical world in the same way that we humans do. This includes the ability to accurately and reliably perform a multitude of image-based tasks such as:

- Analyze and classify objects (Is this a photo of masked person or not ?).
- Object recognition (Can you recognize and label the individuals ?).
- Media recreation (What would the image look like in a different lighting or environment ?).

II.5 A necessity for artificial neural networks

Artificial neural network or Neural Network is a computer system that takes inspiration from the workings of the human brain to learn.Typically, a neural network relies on a large number of processors operating in parallel and organized in thirds. The first third receives raw information input, much like a human's optic nerves when processing visual signals.
Subsequently, each third receives the information output from the previous third. The same process is found in humans, when neurons receive signals from neurons near the optic nerve.

The last third, on the other hand, produces the results of the system [16]. Through an algorithm, the artificial neural network enables the computer to learn and form new data. The computer with the neural network learns to perform a task by analyzing examples to practice, and these examples have been tagged beforehand so that the network can know what it is and where it is originating from.

In our scenario, a neural network can be used to train the computer to recognize objects such as masks in an array of faces. A large number of objects of the same category are presented to the neural network, and the computer learns to recognize this object in new images by analyzing recurring patterns within the example images. Thus, by analyzing thousands of masked faces photos, the Neural Network will learn to recognize the individual in any given photo.

One of the many neural networks available to us in this day and age is the Convolutional Neural Networks(CNN), since it will mostly represent the theme of our thesis in terms of masked faces detection it is necessary that we define it and understand it's various nuances. A convolutional neural network is an algorithm that can take an image as input, assign importance (in the form of trainable weights and biases) to aspects or characteristics of the image, and produce a decision or other form logic based on what it saw in the picture. The required data preprocessing / preparation is very minimal when it comes to CNN architectures because the characteristics of your data are the actual pixel values and with enough training data, CNN will automatically define which of these pixels are most important in it's decision making.

However like mentioned above CNN is not the only option available to us, there are many artificial neural networks at the tip of our fingers, such as modular NN(neural network),feed forward NN, recurrent NN and much more. CNN is a feed forward neural network that is generally used for Image recognition and object classification. While RNN works on the principle of saving the output of a layer and feeding this back to the input in order to predict the output of the layer.

II.6 Feed Forward or Convolutional neural network ?

Another notable artificial neural network is Feed-Forward neural networks, a feedforward neural network is an artificial neural network where connections between the nodes do not form a cycle.As such, it's different from its descendant: recurrent neural networks. The feedforward neural network was the primary and simplest sort of artificial neural network devised. during this network, the data moves in precisely one direction—forward—from the input nodes, through the hidden nodes (if any) and to the output nodes. There are not any cycles or loops within the network. Any saved image is quite simply an array of pixel values ranging from 0-255 , with values in between this range indicating the amount grayness of the pixel, hence the closer we are to the value 0, the grayer we're reaching.

So why not just flatten this matrix into a single vector of pixels and feed it directly into a multi-layered perceptron neural network, where each pixel corresponds to an input neuron? Well simply because it is easier said than done. In the case of a standard feedback NN, each input neuron would be directly mapped to a feature of the dataset, and the assumption here is that each neuron (and therefore feature) is completely independent of one of the others. However this does not apply for image data,pixels in an image have both spatial and temporal dependencies - if you imagine an image of the moon in the night sky, you would expect all pixels near the moon to have similar pixel values (they should all be around 200-255) and the farther away the pixel is from the moon, the darker the pixel becomes (and its value approaches 0).

A standard feedforward neural network would not be able to preserve this kind of spatial and temporal information, and its performance is limited to the information it can get from each individual pixel in the image regardless of other pixels nearby. The convolutional neural network, on the other hand, can capture these dependencies through the application of appropriate filters. The architecture performs a better fit to the image dataset due to the reduction in the number of parameters involved and the reusability of the weights. In other words, the network can be trained to better understand the sophistication of the image.

II.7 Masked face detection classifiers and optimizers

We cannot mention the most famous face mask detecting methods that are available to us and neural networks, without the need to bring out the use of classifiers and optimizers, both of these algorithms are crucial elements in our clustering tasks in minimizing or maximizing an objective function.

II.7.1 Classifiers

Classifier is an algorithm that automatically sorts or categorizes data into one or more "classes". One of the most prominent examples is an email classifier, which checks emails and filters them according to whether they are spam or not. In our cases, the classifiers will be able to tell if a mask is present or not.

- **MobilenetV2**

  MobileNetV2 is a system for mobile visual recognition that includes classification, object identification, and semantic segmentation. This classifier employs Depth wise Separable Convolution, which was implemented to drastically reduce the network's complexity cost and model size, making it ideal for mobile devices or devices with limited computing capacity. Inverted residual structure is another excellent module implemented in MobileNetV2. Non-linearity is removed from narrow layers. The best results for object identification and semantic [10].

- **ResNet 50**

  ResNet50 enables us to train deep neural networks with more than 150 layers. Due to the issue of vanishing gradients, training very deep neural networks was difficult before ResNet. The skip link definition was first introduced by ResNet.

- **VGG16**

  VGG16 is a CNN architecture that is regarded as one of the best vision models available today. The most distinguishing feature of VGG16 is that instead of a large number of hyper-parameters, it focuses on convolution layers of 3x3 filters with a stride of 1. Throughout the design, the convolution and max pool layers are arranged in the same way. Finally, there are two FC (fully connected layers) for output, followed by a softmax. The 16 in VGG16 stands for 16 weighted layers. To test the system's accuracy, three optimizers are applied to each of the classifiers.

II.7.2 Optimizers

Optimizers are algorithms or techniques for altering the characteristics of a neural network, such as weights and learning rate, to minimize losses and they aid in obtaining quicker results.

- **ADAM**

  Adam is a first-order gradient-based stochastic objective function optimization algorithm based on adaptive estimates of lower-order moments. This method is computationally effective and can run on a small amount of memory. It is a diagonal rescaling invariant of the gradients that is well suited for problems with a lot of data and/or parameters. The hyper-parameters have intuitive representations and require little tuning in most cases. Adam performs well in practice and compares favorably to other stochastic optimization approaches, according to empirical evidence [11].

- **ADAGRAD (Adaptive Gradient Algorithm)**

  Adagrad is a parameter-specific learning rate optimizer that adapts to the frequency at which a parameter is modified during training. The smaller the updates get when a parameter receives further updates [12].

- **SGD**

  For each iteration of Stochastic Gradient Descent, several samples are randomly chosen rather than the entire data set. The term "batch" in Gradient Descent refers to the total number of samples from a dataset that are used to calculate the gradient for each iteration. The entire dataset can be used to get to the minima in a less noisy and random manner. The issue occurs when the datasets become excessively large. If the dataset contains a million samples, a standard Gradient Descent optimization technique will require using all one million samples to complete one iteration while performing the Gradient Descent, and this would have to be done with each iteration until the minima was reached.

II.7.3 A comparison of classifiers in masked faces detection

As classifiers proliferate, they naturally differ by their core functioning. Hence,it is natural for us to benchmark them in order to see the best performance in detecting a masked face. As seen in [13], a performance evaluation has been done on a masked face dataset to see the best classifier.
The basic premise of our testing is selecting what we call epochs, epochs are simply the number of passes the machine learning algorithm has made through the whole testing dataset, where the number of epochs equals the number of iterations, if the batch size is the entire training dataset, then a selection of the ratio of 90 trainee has been made as well as on 10 tested ones, then an application of the optimizers was made so that the accuracy and loss of each one of them can be noticed.

Table II.2: MobileNetV2 classifying test

Classifier	Epochs	Train/Test size	Optimizer	Train loss	Train Accuracy	Test Loss	Test accuracy
MobileNet V2	20	90/10	ADAM	0.0090	0.9981	0.0071	1.0000
			ADAGRAD	0.2454	0.94148	0.1811	0.9819
			SGD	0.1549	0.9502	0.0216	0.9855

From our analysis of the performance of the MobileNetV2 classifier in the table II.2, it is observed that performance of ADAM optimizer is good in both training and testing when compared with other two optimizers ADAGRAD and SGD.

Table II.3: ResNet50 classifying test

Classifier	Epochs	Train/Test size	Optimizer	Train loss	Train Accuracy	Test Loss	Test accuracy
ResNet50	20	90/10	ADAM	0.0068	0.9975	0.0557	0.9856
			ADAGRAD	0.1087	0.9693	0.0019	1.0000
			SGD	0.1114	0.9693	0.0100	1.0000

Our observations of the Resnet50 Classifier comparisonn, as seen in Table II.3, shows convincing results for the ADAM optimizer compared with both ADAGRAD and SGD for the preparation and testing, also, both test accuracies are good.

Table II.4: VGG16 classifying test

Classifier	Epochs	Train/Test size	Optimizer	Train loss	Train Accuracy	Test Loss	Test accuracy
VGG16	20	90/10	ADAM	0.2145	0.9826	0.0006	1.0000
			ADAGRAD	1.7911	0.8425	0.4243	0.9638
			SGD	0.5133	0.9536	0.1055	0.9928

In the other hand, as seen in table II.4, the testing of the VGG16 classifier shows promising results in both SGD and ADAM as they are nearly equal in accuracy, but the training and testing is still superior for the ADAM classifier.

II.8 Conclusion

Wearing a face mask has become a necessity whether we like it or not, and the need for systems that detect these objects is now more of an obligation instead of an option.From automated ready-to-go services such as custom vision to manually doing everything, the implementation of a face mask detector is now an essential byproduct of almost all the industries, these detector must be biased by their optimal outcome and their level of accuracy, one that results from studying all the available options to us by gathering all the data that is available to us and carefully testing what works what doesn't like we've mentioned in the sections above.

We also covered the importance of having a dataset, and how to structure that dataset in order to be able to extract a good model out of it, and this can be done by looking at various criterias.Finally from the results we've yielded, it is an obvious choice that using a graphic library such as OpenCV will allow more flexibility in the long term, like mentioned above it will allow us to choose an optimal method, one that can be based on the haar-like feature.

On the classifier spectrum it is relevant that the efficiency of the ADAM optimizer is very high, and the test accuracy of SGD is roughly equal to ADAM for all three classifiers considered above, according to the results of the different classifiers. During research, it was discovered that the MobileNetV2 classifier produces the best results with the highest accuracy.

From the acquired informations, it is now natural for us to use the cited library, classifier and face-detecting method in the conception part that will come next in our research.

Chapter III

Face mask system conception

III.1 Introduction

When looking at the computer vision technology that is widely applicable to the coronavirus's global health problem, the aim is to replicate that concept as automatically possible through the growth, training, and deployment of this project, the goal is to automate the task of determining whether or not someone is wearing a protective mask, and this can be achieved using the optimizers and classifiers mentioned in the second chapter.

In this part we'll go through the technicalities of detecting the masked faces as an image as well as deepen our understanding in terms of live feeds such as webcam. We opted for the Convolutional Neural Network since it is a common machine learning and allows to have a magnitude of raw data in order to give an accurate result.

III.2 The process of detecting a face on masked faces

Wearing a mask has become a very common action since the coronavirus epidemic. As face detection technologies become more and more popular smartphones are having more problems with identifying individuals.

To better understand why masks often prevent face detection, we need to take a look at how these systems work. Like we've mentioned in the above chapter, in order to identify a person, the algorithm will place points in certain places on the face and will calculate distances between them which will be specific to each person. The problem is that the areas of the face where the markers are most numerous are at the level of the eyes, nose and mouth. If a mask therefore hides the nose and mouth, the algorithm will no longer have enough information to effectively recognize that individual. Before precising those methods and those nuances, we would need to precise the phases a face detection on face mask passes through. In total, there are two phases that need to be done :

- **Phase 1 :**
  - Loading a dataset into the system.
  - Classifying it to generate a model through classifiers. (like MobileNetV2, ResNet50 and VGG16).
- **Phase 2:**
  -Loading the face mask classifier model.
  -Detecting the faces in the images (through one of the methods we've talked about in chapter 1).
  -Apply the chosen classifier that will give "With Mask" and "Without Mask".

Those phases are represented in Figure II.1 through a simple diagram :

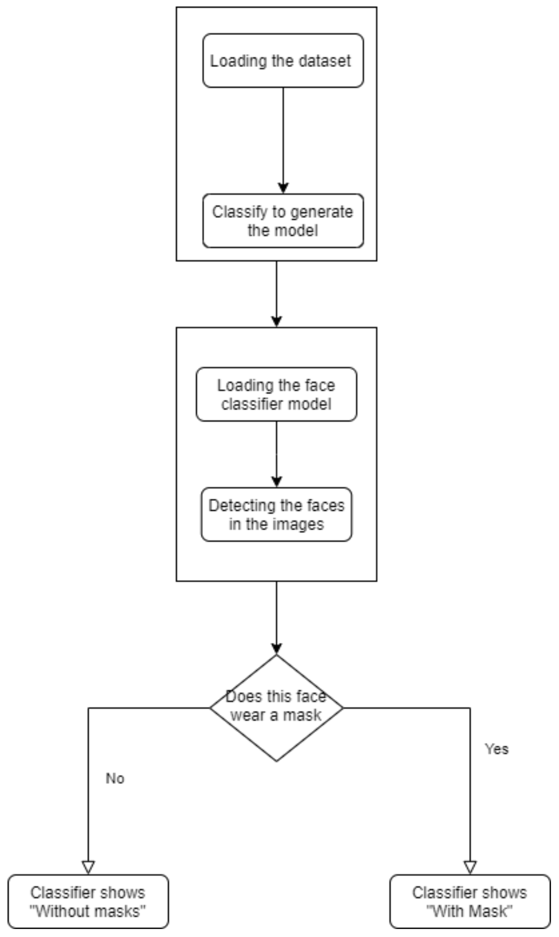

Loading the dataset
Classify to generate the model
Loading the face classifier model
Detecting the faces in the images
Does this face wear a mask
No
Yes
Classifier shows "Without masks"
Classifier shows "With Mask"

Figure III.1: Diagram representing face detection on mask process

III.3 The Procedure

For the realization of our work, we will follow the stages of a process of recognition of the forms namely:

- **Acquisition of the data set.**
- **Learning and modeling.**
- **Development, training and testing.**
- **Deployment of the model for live webcam feeds.**

III.3.1 Acquisition and exploration of the dataset

There were 1376 images in the initial data collection that was collected. The remaining 686 photographs were of individuals who were wearing a face mask, with 690 images of people who were not wearing a face mask. We will use predisposed data , then greatly extend this dataset with our own images to enhance the model's performance and generalization potential. The cumulative number of photographs in the final dataset was 4,814: 2,408 for the "With Mask" class and 2,406 for the "No Mask" class.

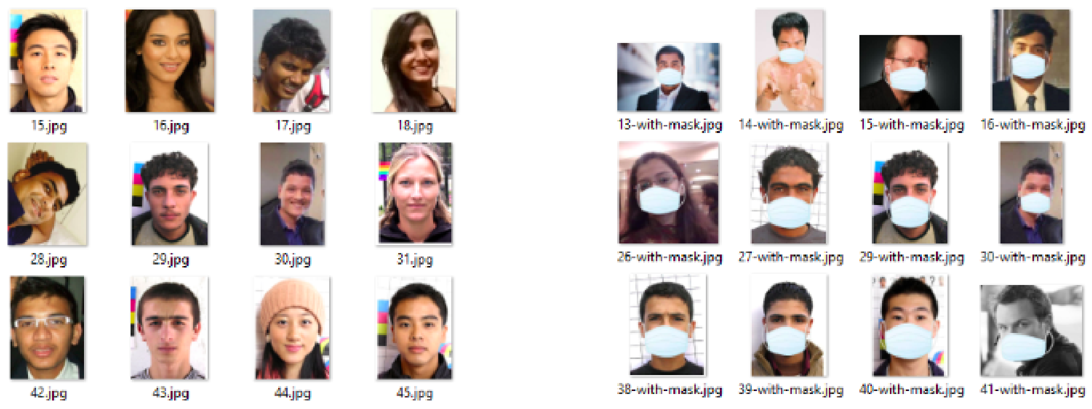


Figure III.2: Image dataset, two classes - "With mask" and "Without mask".

III.3.2 Data preprocessing

As previously stated, the preprocessing steps required in the field of computer vision are very minimal. For this project, we only took two preparation steps. They were:

1. First, each image was converted from RGB (red, green, blue) to a grayscale image. Color images have three channels (one for each color) while grayscale images have only one color channel (the “ grayness ” of each pixel)

2. The images will then be resized to help reduce the complexity of the CNN model, thereby reducing the computational power required to train the model.

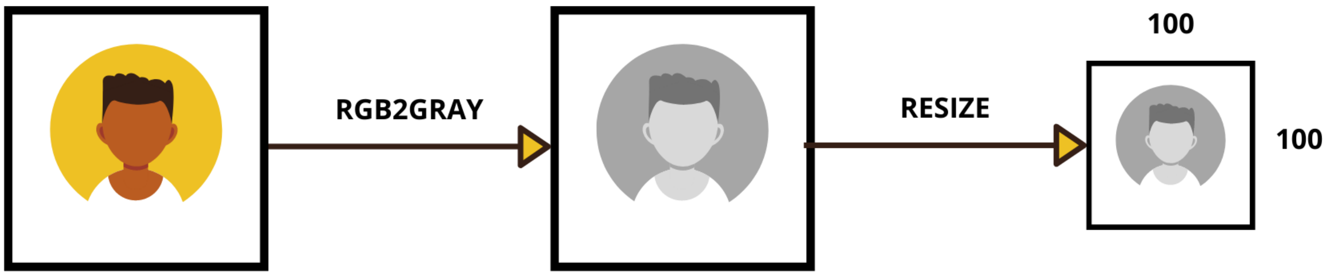


Figure III.3: Data preprocessing steps

For each image in the project folder, the use of the color function will be implemented to convert the BGR image to a grayscale image as seen in Figure III.3. A resize function will be implemented to reduce the dimension of the image (i.e from a 500x500 image to a 100x100 one). The image size parameter will be set to 100, and so each image becomes a 100 x 100 square image. The prepared image is added to the data list and the class label will be added to the target list.
We can reduce the model’s complexity and training time by translating the data list to a more efficient numpy array, then dividing the array by 255, which will normalize the pixel range between 0 and 1.

III.3.3 Development, training and testing of the CNN architecture

To start developing the convolutional neural network, it is empirical to start by importing the required functions from the Keras ML(Machine learning) library. A sequence of models are then used to describe the CNN architecture as seen in Figure III.4. Hence the overall architecture of the project will contain the following :

1. Both layers willinclude a three by three convolutional window and a ReLU activation feature(rectified linear activation function). The first convolutional layer will yield 200 function maps, while the second one will yield around 100. This initiative will keep the stride and padding untouched.
2. Two max Pooling layers, both of which will have window sizes of two by two.
3. Two fully connected layers (one hidden layer, one output layer). The hidden layer will contain around50 neurons and the output layer will only have two (one for each class). The Softmax activation function will also be used to calculate the class probabilities. Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector [14]
4. The model will also take use of abandonment to avoid overfitting the model to the training data.

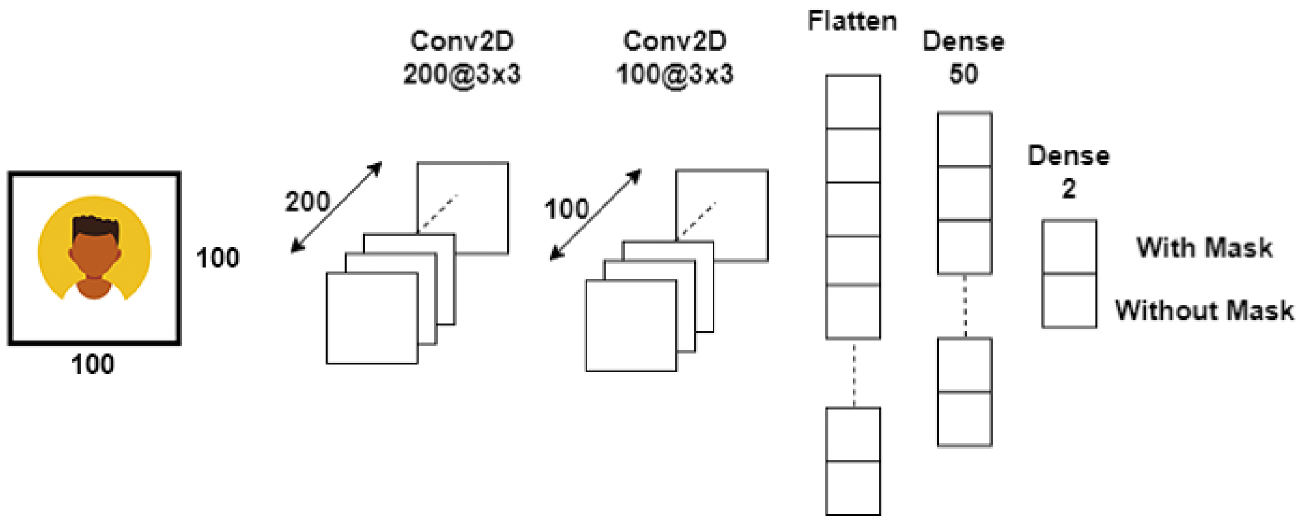


Figure III.4: CNN Architecture - 2 Conv layers, 2 Max Pooling, 2 FC (including output layer)

The model will be trained for a total of 20 epochs, with the loss and precision being transmitted through the programming interface. To test the model, 20 percent of the data will be used. The model with the highest results is then saved to our project list.

For visual purposes, the failure value for and training epoch will be plotted in a graph. A reduction in the training will suggest that the model is beginning to overfit our training results. While there are a few improvements that may be made to better counteract overfitting, the model would still reach an excellent accuracy score on validation results even without them.

Along with plotting the model's failure in each epoch, we'll also plot the model's accuracy value. If the metric averages about 97 percent across the validation collection, then that's a great score considering the dataset's scale.

III.3.4 Deployment of the model for live webcam feeds

Once that the model has been trained and its performance has been evaluated on the invisible images of the test set, we can think about how we want to feed our live webcam images into the model for prediction, and how to feed our live webcam images into the model for prediction as seen in Figure III.5.

The way we want the prediction to be returned to the user in an appropriate format is that, all images in the webcam feed will need to go through the same data pre-processing steps that we will be done on the training data, with the added step of identifying the location of a face in the image of the webcam. The visual below shows how the webcam feed will be processed from the original image to a cropped "face only" image, to the fully pre-processed image ready for classification.

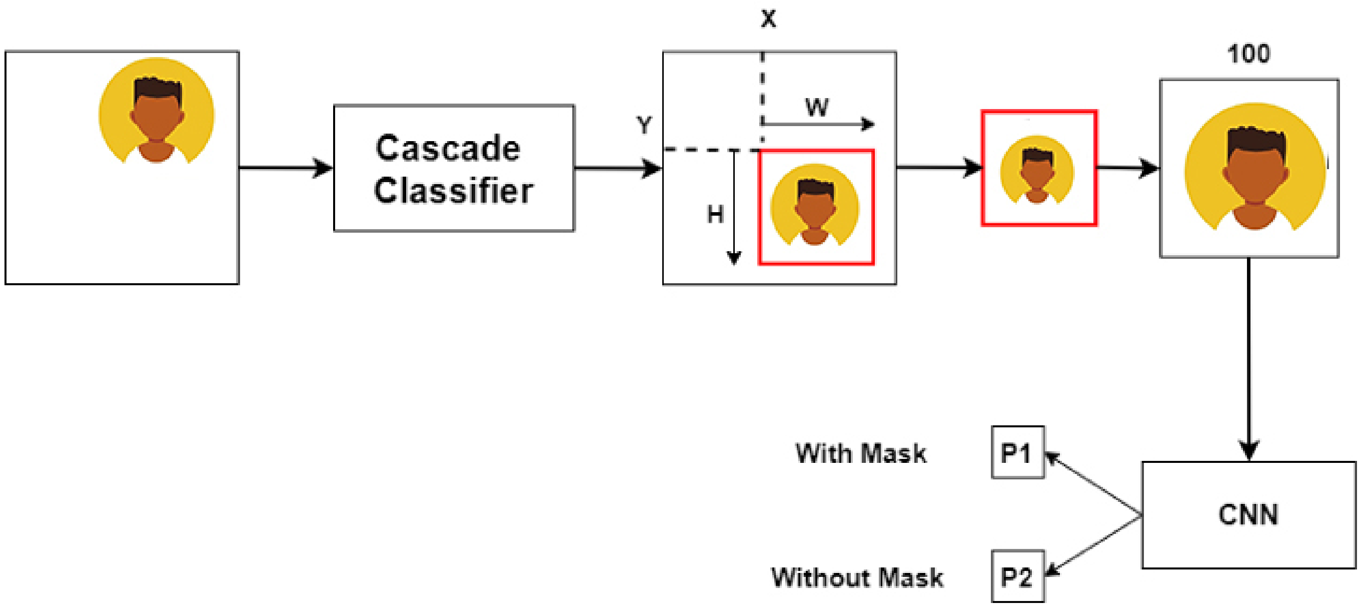


Figure III.5: Deployment process from model to live webcam feeds

We determined that the visual input from the classification challenge would be in the form of two rectangular boxes (one will have a green outline if a mask is present, and a red outline if a mask is not.) The class mark will be on one side, and the class label will be on the other ("Mask" or "No Mask"). These boxes will appear around our cropped "face only" picture on the live webcam feed.
To detect faces in a picture, we'll first use the Haar Cascade Frontal Face technique. This classifier, as we've seen, is a widely used algorithm for detecting one or more faces in an image. Then we'll describe the color dictionary and the mark dictionary (Mask or No Mask) (in example black if the individual is not wearing a mask and green if they are wearing a face mask).

We'll use a read feature to take each individual frame of the video while the webcam stream is live, then translate the image from BGR to grayscale, and then split all observed faces into their individual data objects. We crop the image to include only the face with each x, y, weight, and height of faces

observed, then scale it to 100 x 100, the same dimensions as our training images.
Once the image has been fully prepared and is ready to be classified by the algorithm, our model function will be the line of code that evaluates the image and receives a classification decision. Then the visual feedback of the rectangle and the output label are chosen based on the classification result and are overlaid on the webcam feed in the forms of live black or green rectangles.

III.4 Conclusion

Adding a Fully-Connected layer is a (typically) low-cost way of studying non-linear combinations of high-level features represented by the convolutional layer's performance. In that space, the fully-connected layer is learning a possibly non-linear function. And while the procedure of creating our model to later then implement with Keras, can be time consuming, we must also note that it is an important step that will allow us to seamlessly use our dataset with our system.

On our mask/no mask dataset, we will fine tune MobileNetV2 and obtain a classifier that is accurate. We will then use this face mask classifier to classify faces and since we used the MobileNetV2 architecture, our face mask detector is reliable and computationally effective, making it easier to deploy the model in various systems and environments.

Chapter IV

Implementation of the face mask detection system and results

IV.1 Introduction

We've concluded in the third chapter of our research the importance of undergoing the necessary steps to come up with an approximately accurate model that will allow us to detect masked individuals, as well as knowing how important it is to follow the outlined procedures of our conceptual system.

In this chapter, we will oversee the steps to build a deep learning model for face mask detection using Python, Keras, and OpenCV. We will first undergo the creation of the face mask detector model for detecting that will allow us to know whether a person is wearing a mask or not. The model will then be qualified with network engineering using Keras. After defining the several tools and libraries we will be using, we will go over the three distinct steps of our implementation.

The first step of the project will be processing our dataset, the second will be generating our model while bearing in mind the results we've achieved in the third chapter, the third and final step will be live checking for face masks using our webcam with the help of OpenCV. Afterwards, we will review our progress made through the several attempts made and showcase our final result, these processes can be explained through the diagram represented in the figure below.

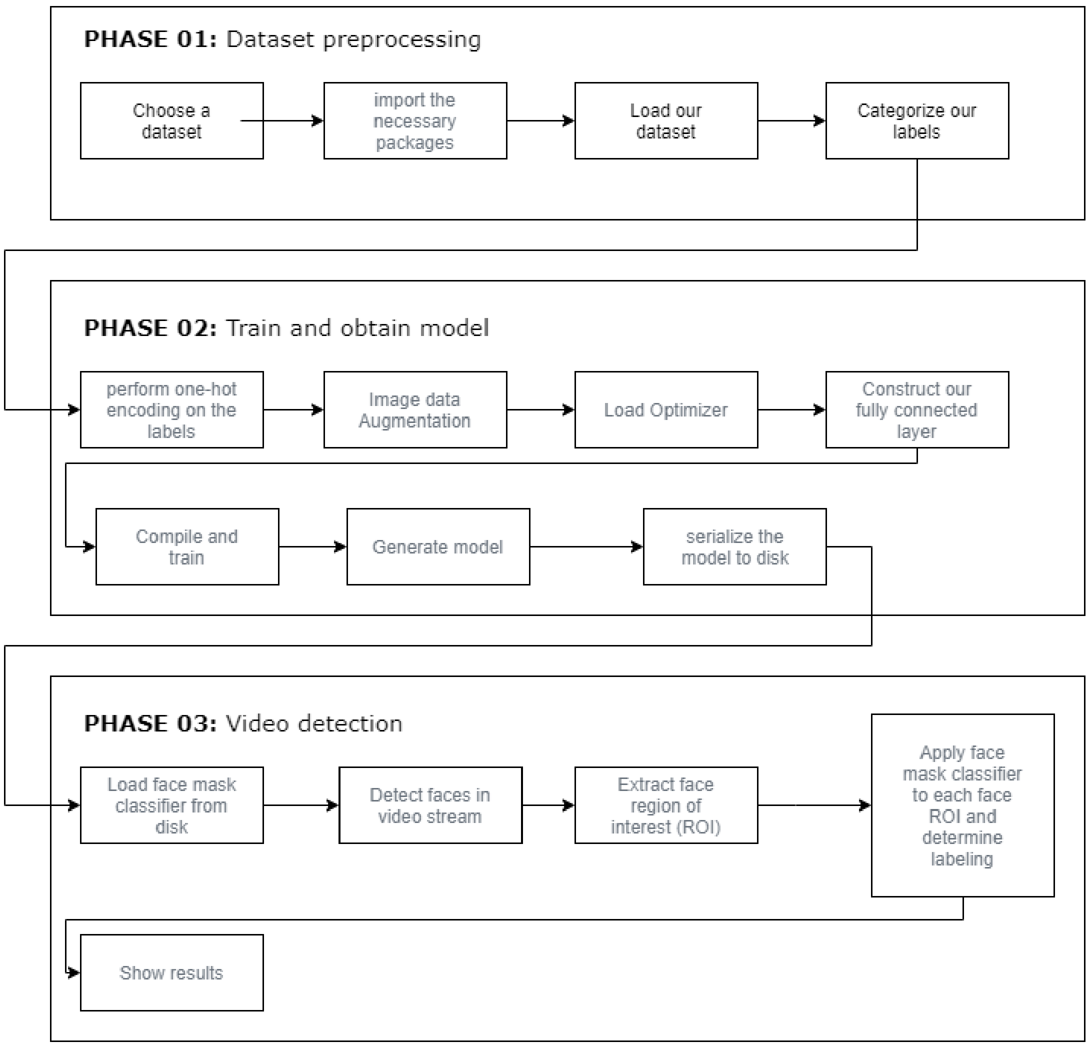

PHASE 01: Dataset preprocessing
Choose a dataset
import the necessary packages
Load our dataset
Categorize our labels
PHASE 02: Train and obtain model
perform one-hot encoding on the labels
Image data Augmentation
Load Optimizer
Construct our fully connected layer
Compile and train
Generate model
serialize the model to disk
PHASE 03: Video detection
Load face mask classifier from disk
Detect faces in video stream
Extract face region of interest (ROI)
Apply face mask classifier to each face ROI and determine labeling
Show results

Figure IV.1: Three phase diagram portraying each major phase

IV.2 Tools Used

IV.2.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together [17]. It is the language we will use for our project, our choice was made according to its very strong popularity in the ML field, and especially the availability of the libraries needed.

IV.2.2 GitHub

GitHub is a website and cloud service that helps developers store and manage their code, as well as track and control the changes made to it. In order to store our project we used a Github repository.

IV.2.3 Jupyter Notebook

The Jupyter Notebook is a web-based interactive computing platform. The notebook combines live code, equations, narrative text, visualizations, interactive dashboards and other media. Jupyter Notebook is an important part of the ML Python ecosystem, since it gives the possibility to illustrate the analysis process and organize it in different steps. Results can also be captured after the execution which will help us tremendously in showcasing what we need.

IV.2.4 Google Colab

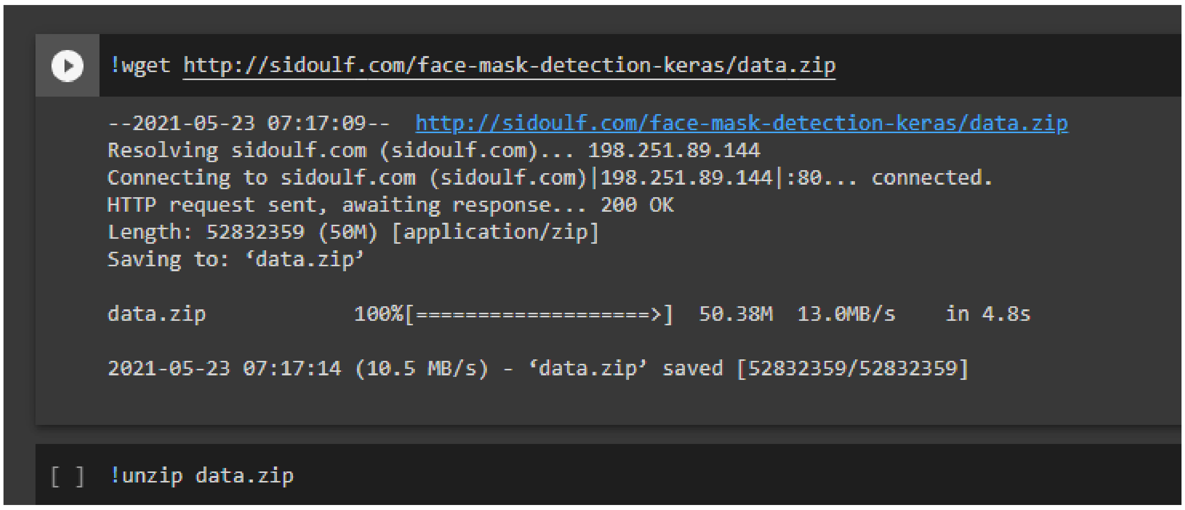
Google Colab provides the user with a Jupyter notebook for an interactive coding experience. It connects seamlessly with GitHub and all the notebooks we create can be stored in our Google Drive. This also makes our Jupyter Notebook available over the internet by sharing it (either by a link or by email), so anyone can execute it in real time without the need of a physical machine.

Before we start coding,it is imperative to have a quick idea of how we're going to proceed with the overall process on Google Colab.

Firstly, we will create a notebook, it simply is a Jupyter notebook but it will be hosted by Colab, it is obvious that the file extensions differs from the usual ones programmers tend to see, indeed the ".ipynb (Formerly known as the IPython Notebook) the IPython Notebook is now known as the Jupyter Notebook. It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media". Secondly, we will mount our google drive account on the Colab project then lastly, we will use our notebook to extract and process the dataset as well as save it to our Google Drive.

IV.3 Processing the chosen dataset

At first, we will download the dataset we've mentioned in chapter 3, where a total of 1376 photographs were retrieved from the initial data collection. Of the 690 images, 690 were people without the face mask, and the remaining 686 were people with the face mask. Prajna Bhandary had this dataset kindly created. and we've divided it with labels such as "With mask" and "No Mask". So far all we've covered is a small portion of how we'll use the service provided by Google to train our masked and unmasked individuals dataset.

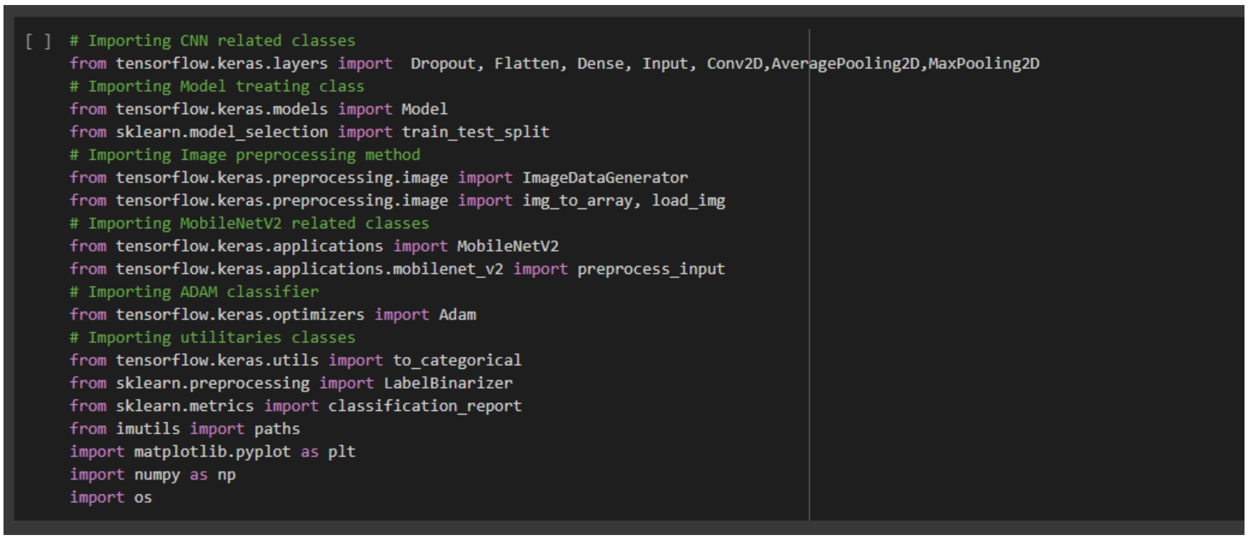


```
!wget http://sidoulf.com/face-mask-detection-keras/data.zip

--2021-05-23 07:17:09--  http://sidoulf.com/face-mask-detection-keras/data.zip
Resolving sidoulf.com (sidoulf.com)... 198.251.89.144
Connecting to sidoulf.com (sidoulf.com)|198.251.89.144|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 52832359 (50M) [application/zip]
Saving to: ‘data.zip’

data.zip            100%[===================>]  50.38M  13.0MB/s    in 4.8s

2021-05-23 07:17:14 (10.5 MB/s) - ‘data.zip’ saved [52832359/52832359]

[ ] !unzip data.zip
```

Figure IV.2: Downloading our dataset in Google Colab and unzipping it

IV.4 Libraries used

Firstly we will need to import all of the relevant libraries and modules required for our project. These libraries for example are Keras and Sklearn, before diving further into each subsection of these respected categories we must first understand and assess what they are and how they'll be relevant to our project. We will explain the classes used when showcasing their utilization, for the time being let's focus on the libraries.



```
[ ] # Importing CNN related classes
    from tensorflow.keras.layers import  Dropout, Flatten, Dense, Input, Conv2D,AveragePooling2D,MaxPooling2D
    # Importing Model treating class
    from tensorflow.keras.models import Model
    from sklearn.model_selection import train_test_split
    # Importing Image preprocessing method
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.preprocessing.image import img_to_array, load_img
    # Importing MobileNetV2 related classes
    from tensorflow.keras.applications import MobileNetV2
    from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
    # Importing ADAM classifier
    from tensorflow.keras.optimizers import Adam
    # Importing utilitaries classes
    from tensorflow.keras.utils import to_categorical
    from sklearn.preprocessing import LabelBinarizer
    from sklearn.metrics import classification_report
    from imutils import paths
    import matplotlib.pyplot as plt
    import numpy as np
    import os
```

Figure IV.3: Importing the required libraries

IV.4.1 Keras

Keras is an open source library containing Python-written neural network components. It will run on TensorFlow and other open source end-to-end learning platforms. The library has been designed to be modular and user friendly, so that its basic principles can easily be assimilated and understood.

IV.4.2 Sklearn

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

IV.4.3 numpy

Numpy is a widely used library in the Python community, its usage consists in providing a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more [18]. It will then naturally come in handy since converting our data into arrays is primordial in order to achieve our results.

IV.4.4 Categorizing our dataset

Image, or data processing in general is our first step towards making our program, in this step we will convert all of our images (With mask and without mask) into arrays so that we can create a deep learning model. This is done by simply reading our directory which is the dataset in itself. We then proceed to create two lists, one that will append all of the image and another one that will serve as a labelling parameters to differ between “With Mask” and “Without Mask”. Once that has been done we will start looping through the two categories by first going through the With mask images then once it's complete, it will start appending the Without mask images.

The next step is using a pre processing function nested within Keras's image pre processing libraries called load img which will allow us to load the image into PIL(Python Imaging Library) format in other words, it allows us to load the images path where we can also specify our target size which is the height and width of the image, this can come in handy with big datasets because it will allow us to save some memory in the long run. Once the images are saved we will convert them into an array by using the img to array function which allows us to convert a python image library instance into a numpy array.

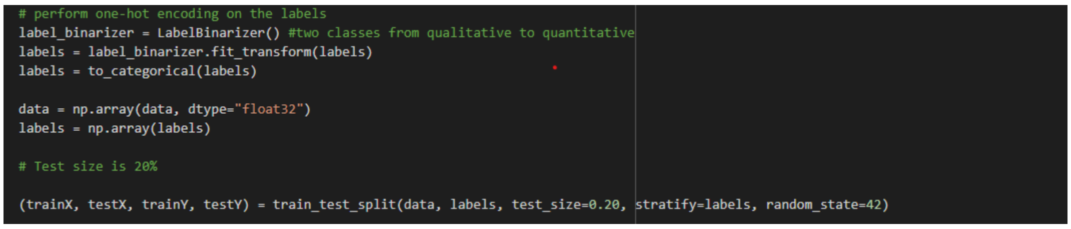
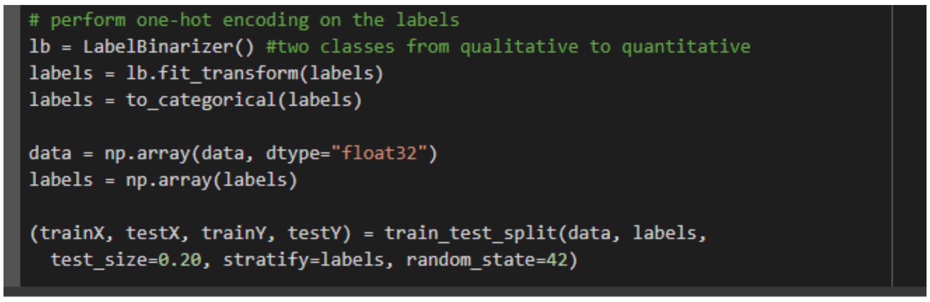

```
# perform one-hot encoding on the labels
label_binarizer = LabelBinarizer() #two classes from qualitative to quantitative
labels = label_binarizer.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

# Test size is 20%

(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels, random_state=42)
```

Figure IV.4: Looping over our dataset

When the image gets converted to an array, we will use the preprocessing input, however this is only a requirement if we will be using MobileNetV2 in our model. Once all the above parameters are in place, all that will be left to do is to append our images which is basically an array, to our preexisting list and when that is done, we will link the corresponding categories into our labels.

After all the images have been appended within their respective lists, we will use the LabelBinarizer in order to convert our data into ones and zeros because it is crucial to convert this data into numpy arrays hence the np.array method. Once that is done we will use the Test Train Split to divide our training and testing sets. The Train Test Split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, We don't need to divide the dataset manually. By default, Sklearn Train Test Split will make random partitions for the two subsets. We went ahead and used a percentage of twenty percent (0.2) which means that out of a thousand image, we will test twenty percent, and the remaining eighty percent will be for the training with an additional random state. The random state parameter is used for initializing the internal random number generator, which will decide the splitting of data into, train and test indices in your case. If random state is None or np. random, then a randomly-initialized RandomState object is returned.



```
# perform one-hot encoding on the labels
lb = LabelBinarizer() #two classes from qualitative to quantitative
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,
  test_size=0.20, stratify=labels, random_state=42)
```

Figure IV.5: Looping over our dataset

IV.4.5 Image Data Augmentation

Thanks to the proposed class imageDataGenerator from the Keras library, it is possible to increase our images in real time while our model continues to train. Each training image can be randomly modified as it is moved to the model. This is particularly helpful in our case, since the size of our Dataset is not that important. The imageDataGenerator class will accept several parameters so we can customize its behaviour to our liking. We start to give it a rotation range, which randomly rotates the generated images from 0 degrees to the given parameter, 20 in our case. The same for zoom range, which zooms in the images randomly from 0 to 15 percent.

The class also gives the possibility to shift width and height range, it does so by subtracting the image width or height to one of the values chosen randomly from the given range. For example, we gave the height and width shift range parameters the values of 0.2 each. If, let's say, the image's width is 300px. It will each time take a random value from 0 to 0.2, multiply the value by 300, then the original value would be subtracted by it. So, if the algorithm has chosen 0.1, it'd make (300*0.1)-300=270px. The same process would be randomly done to the height.

The shear range parameter exploits the concept of the human's eye. Since a human can see something from different angles, the computer should be able to do so. The shear range parameter then distorts the image along an axis (x or y). We gave it 0.15 so it shifts 15 degrees on the x and y axis. Finally, all that horizontal flip does, is flip the images horizontally on a random basis. //

With this done, our dataset will be clearly increased and randomized, thus not biased, in order to maximize results in the learning process.

IV.5 Creating our model

From what we assessed in chapter two and three, the model of face mask detection can optimally be done with the MobileNetV2 classifier and/or the two Conv2D layers (along with the Flattening and Dense to classify). Machine Learning consists of a lot of experimenting, trying out techniques and shifting architectures. So, we went ahead and experimented on the methods written above. We will mostly use the loss percentage on validation and test sets as these losses represent well how our model will perform.

IV.5.1 Using two Convolutional Layers

At first, we tried the bare dataset without Image Data Augmentation along with the two Conv2D Layers, which gave us a really under-performing model, expressed by the validation loss and validation accuracy on the graph of Figure IV.6.

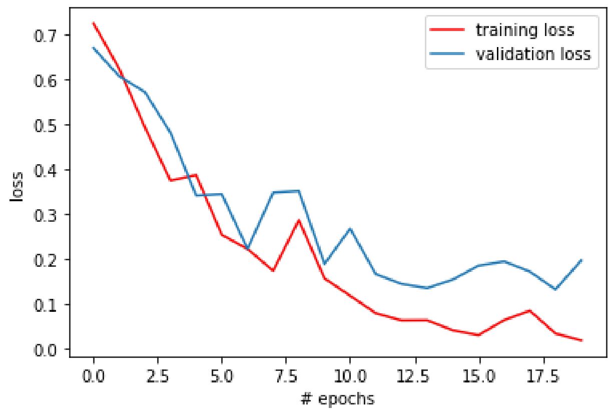


Figure IV.6: Validation loss and training loss with two Conv2D Layers and with no Image Data Augmentation

For the loss on the validation loss which is the metric that should be the lowest since validation is basically real-time conditions, we can see that it performs really poorly with increasing values, staying mostly between 0.2 and 0.3. This shows that our model is far from ready, surely because of the datasets low number of images.

IV.5.2 Using two Convolutional Layers with Image Data Augmentation

Our next attempt consisted on using the Image Data Augmentation class to get more images during the training phase which gave us the plot in Figure IV.7.

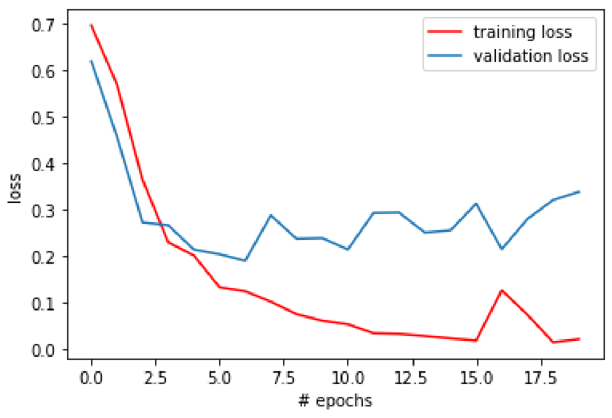


Figure IV.7: Validation loss and training loss with two Conv2D Layers and with Image Data Augmentation

As we can see, the loss on the training set begins to decay as epochs go, going near the zero mark, then it peaked again on the 16th epoch. While the training loss shows better results than the previous plot, even though fluctuating a lot.

IV.5.3 Using two Convolutional Layers with Image Data Augmentation and MobileNetV2 classifier

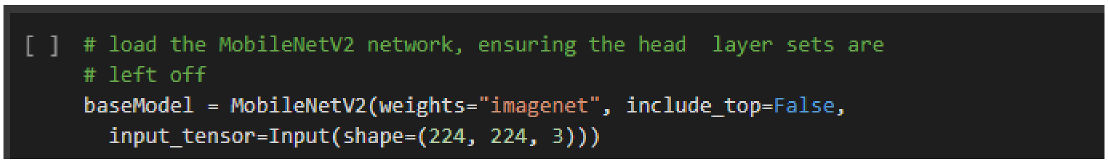
In the next attempt, we are going to use the convolutional neural network architecture as we have mentioned in the parts above with a small modification that will include the usage of MobileNetV2.

After the image is processed we will send that into MobileNetV2 then we will do the MaxPooling, flatten it then create a fully connected layer then get the output.

The usage of MobileNetV2 was used because it's comparbily very fast and uses lesser parameters although it contains a lot of advantages MobileNetV2 does come with a perceived advantage and we have used the word perceived because this disadvantage depends on the size of the dataset, such as it can be less accurate whenever the dataset is really big, this is not a problem for our use case because our model does not require to process a huge amount of data in order to learn.

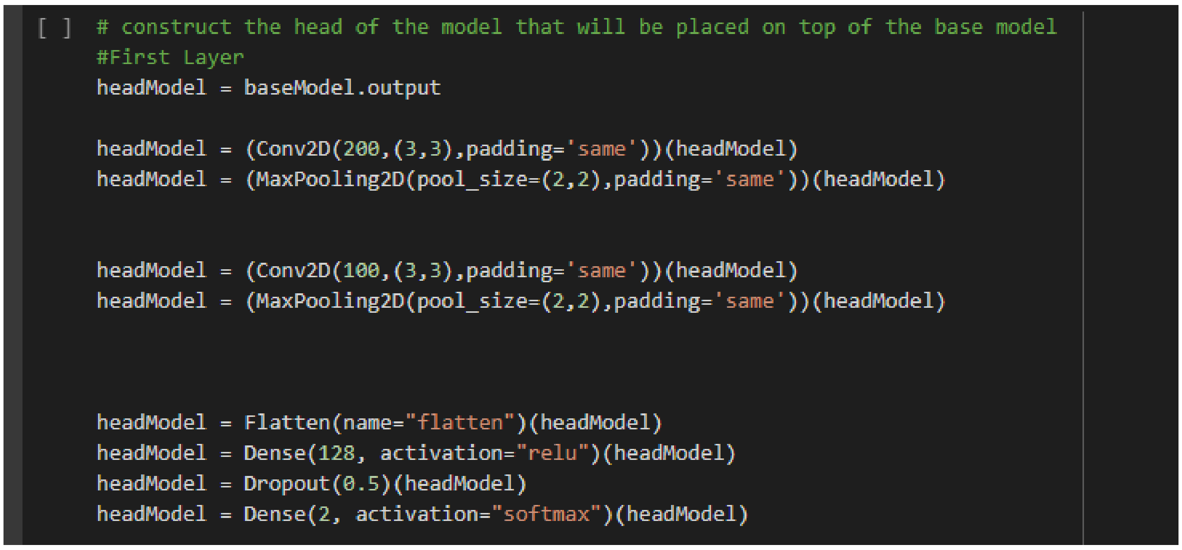
Firstly, we will give the initial learning rate that is a very small and significant value, it basically is a parameter for tuning in an optimizing algorithm which sets the step size of each iteration with a minimal loss function moving, when our learning rate is small the loss value will be calculated properly which means a better accuracy.

We then determine the number of epochs and BS (Batch Size), the batch size is not necessarily a static value but rather one that can be generated through trial and error.



```
[ ] # load the MobileNetV2 network, ensuring the head  layer sets are
    # left off
    baseModel = MobileNetV2(weights="imagenet", include_top=False,
      input_tensor=Input(shape=(224, 224, 3)))
```

Figure IV.8: Using MobileNetV2



```
[ ] # construct the head of the model that will be placed on top of the base model
    #First Layer
    headModel = baseModel.output

    headModel = (Conv2D(200,(3,3),padding='same'))(headModel)
    headModel = (MaxPooling2D(pool_size=(2,2),padding='same'))(headModel)


    headModel = (Conv2D(100,(3,3),padding='same'))(headModel)
    headModel = (MaxPooling2D(pool_size=(2,2),padding='same'))(headModel)



    headModel = Flatten(name="flatten")(headModel)
    headModel = Dense(128, activation="relu")(headModel)
    headModel = Dropout(0.5)(headModel)
    headModel = Dense(2, activation="softmax")(headModel)
```

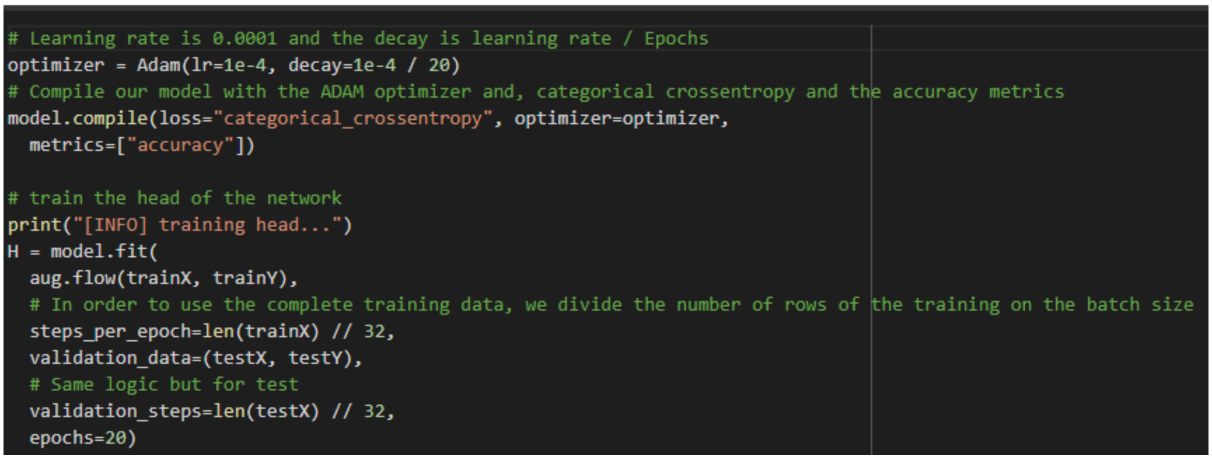
Figure IV.9: Creating our fully connected layers

By using MobileNetV2 we are actually generating two types of models, the first one will be MobileNetV2's model which results will be parsed in the normal model we will further develop, respectively we can generate the following models:

1. Base Model (The one generated by MobileNetV2)
2. Head Model (The head of the model that will be placed on top of the base model)

So, with the help of MobileNetV2 we are creating a base model with as a crucial parameter the weights, which are set to imagenet, this simply indicates that we want to utilize the pre-trained ImageNet weights for the particular model, whilst also mentioning the input tensor which basically is the shape of the image that is going through like we've mentioned in the image processing part.

The next step is to construct our fully connected layer, to do so we are first creating our head model object in which we will parse the output provided by MobileNetV2 (Which is the base model), we then will proceed by using Conv2D,Max Pooling,Flatten,Dense,Dropout all of which have been thoroughly explained in chapter three.



```
# Learning rate is 0.0001 and the decay is learning rate / Epochs
optimizer = Adam(lr=1e-4, decay=1e-4 / 20)
# Compile our model with the ADAM optimizer and, categorical crossentropy and the accuracy metrics
model.compile(loss="categorical_crossentropy", optimizer=optimizer,
  metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
  aug.flow(trainX, trainY),
  # In order to use the complete training data, we divide the number of rows of the training on the batch size
  steps_per_epoch=len(trainX) // 32,
  validation_data=(testX, testY),
  # Same logic but for test
  validation_steps=len(testX) // 32,
  epochs=20)
```

Figure IV.10: Creating our fully connected layers

Once our models are in place we have used the model function which takes two parameters which are input and output whereas, the input is our base model and the output is our head model, we then must sort of freeze the layers of the base model so they are not updated in the training process and once that is done we will give an initial learning rate that we have mentioned above and a decay with a binary cross entropy, with Adam as an optimizer which is similar to ReLu, that latter is good for any image detection method with a tracked accuracy as the only metric we will be looking out for.

Once all of the steps above have been implemented, we fit our model with a flow of the augmented image data so we can get more training data to help our model learn.

At last, after combining the two methods that we got on chapter two and three, we got ourselves the best results yet.

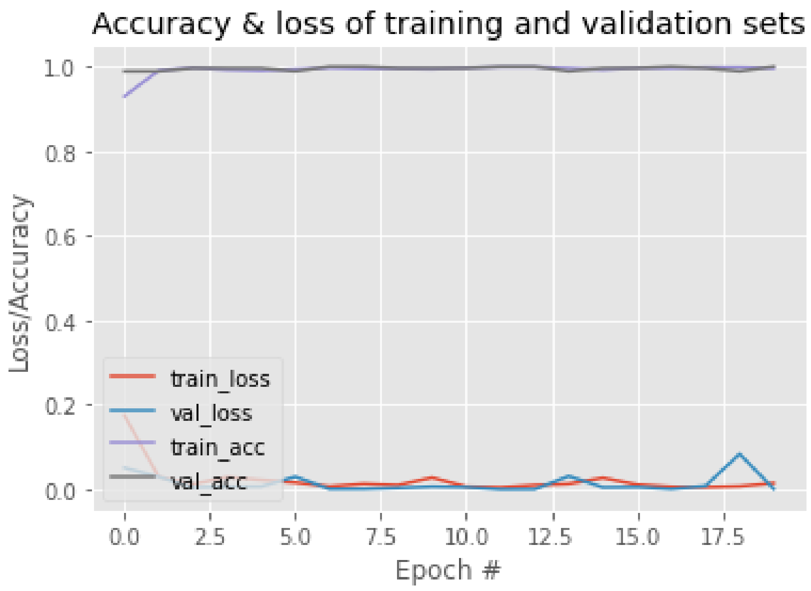


Figure IV.11: Our final accuracy, loss of training and validation sets

As we can see in figure IV.11, the plot is consistent, the accuracy is near the 100 per cent and attains it in both cases, while most importantly, the training loss and validation loss are near zero and are mostly very stable throughout the epochs.

IV.6 Discussing our results

We can then proceed to a comparison of the methods used throughout our trial and error process to see that it is increasing over the time and that using a classifier, in this case, MobileNetV2 and augmenting the image helps tremendously in the performance of a machine learning model.

Table IV.1: A comparison of the different approaches we tried

Method	Loss Mean	Validation Loss Mean	Classifier	Image Data Augmentation
Conv2D	0.22	0.29	No classifier	No
Conv2D	0.19	0.24	No classifier	Yes
Conv2D	0.051	0.014	MobilenetV2	Yes

When comparing to other works such as [19], we can compare the accuracy of the several methods used to ours (taking the peak accuracy on a similar dataset by size), we see that our accuracy surpasses all the other classifier usages :

Table IV.2: A comparison of another similar project when using other classifiers

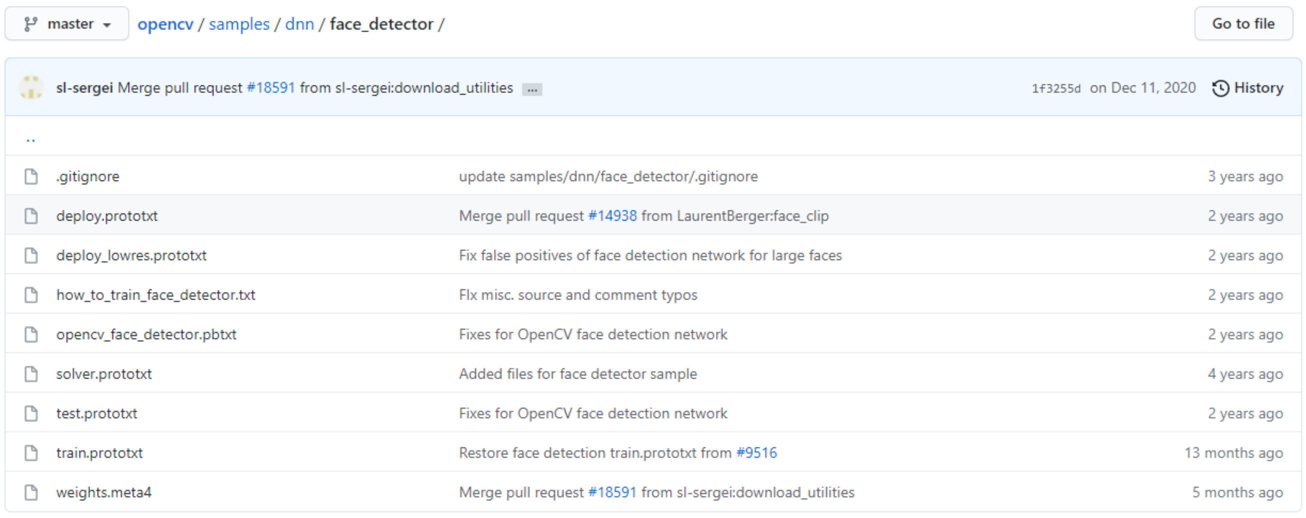
Approach	Accuracy	Precision	Recall
ResNet50	98.38	98.38	98.38
GoogleNet	98.18	98.19	98.18
Jignesh et al.	99.80	99.80	99.80
MobileNetV2 (our approach)	99.91	99.91	99.91

IV.7 Using our model

IV.7.1 Video Detection

Once our face mask model has been generated, and the analysis of the plot makes perfect sense we can start moving on to the live detection part. We must note that we currently have developed on the model for the mask detection, in order to make the webcam detection possible, it is necessary to have a model in regard of face detection only.

The idea is that with the help of the face detector model which can be an out-of-the-box pre trained Haar Cascades, or a more accurate deep learning-based face detector that has been included within the latest releases of OpenCV, this latter comes in the form of what we call a Caffe-based face detector and can be found within OpenCV's official github repository within the dnn(Deep Neural Network) samples.


master opencv / samples / dnn / face_detector / Go to file

sl-sergei Merge pull request #18591 from sl-sergei:download_utilities ... 1f3255d on Dec 11, 2020 History

..		
.gitignore	update samples/dnn/face_detector/.gitignore	3 years ago
deploy.prototxt	Merge pull request #14938 from LaurentBerger:face_clip	2 years ago
deploy_lowres.prototxt	Fix false positives of face detection network for large faces	2 years ago
how_to_train_face_detector.txt	FIx misc. source and comment typos	2 years ago
opencv_face_detector.pbtxt	Fixes for OpenCV face detection network	2 years ago
solver.prototxt	Added files for face detector sample	4 years ago
test.prototxt	Fixes for OpenCV face detection network	2 years ago
train.prototxt	Restore face detection train.prototxt from #9516	13 months ago
weights.meta4	Merge pull request #18591 from sl-sergei:download_utilities	5 months ago

Figure IV.12: DNN Face Detector

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. Choosing this framework makes perfect sense because of its following features :

- Expressive architecture
- Extensible code
- Speed (The ability to process over 60M images per day)

A prototxt file is a text file that holds information about the structure of the neural network: A list of layers in the neural network. The parameters of each layer, such as its name, type, input dimensions, and output dimensions.

These two sets of files are mandatory when deep learning models developed using Caffe are used, so we will first detect the faces using the already available caffe model provided by OpenCV and with the help of our generated model we will be able to detect whether a face in particular is using a mask or not.

Firstly, we will start by loading FaceNet, which is a face recognition pipeline that learns how to map faces to a location in a multidimensional space where the distance between points directly corresponds to a measure of face similarity, this later will have as parametres the caffe model we have mentioned above, and in order to use them we have used a method called readnet which helps reading a deep learning network. Once that is in place we will load our mask model which means that we now have loaded both the face detecting model and the one that would help us detect the masks on the individuals.

The next logical thing to do, is to load our camera, in order to do that we are using a VideoStream, inside this function we precire the “src” which is short for source of our camera (i.e primary camera=0 and so on). We then create a loop to read the input frame by frame. Once that is in place we can call back our function which will have our present three arguments which are the following :

1. Our frame

2. The face detector model
3. The mask detector model

This latter will return both the location (Which is the X and Y axis coordinates of the particular rectangle of our frame) and predictions (The accuracy of the person wearing the mask or not).

With that method in place, all we will do is unpacking and create a sort of overlay on our live stream with BGR parameters (Green= Wearing mask, Red= not wearing mask).

IV.7.2 Demo

Once the program launched, we will be prompted with a frame with a red or green label along the precentage of prediction next to our pre existing labels that we have mentioned above. As seen in the Figure IV.13, if the label is red, then that person is highly probable that they are not wearing a mask.

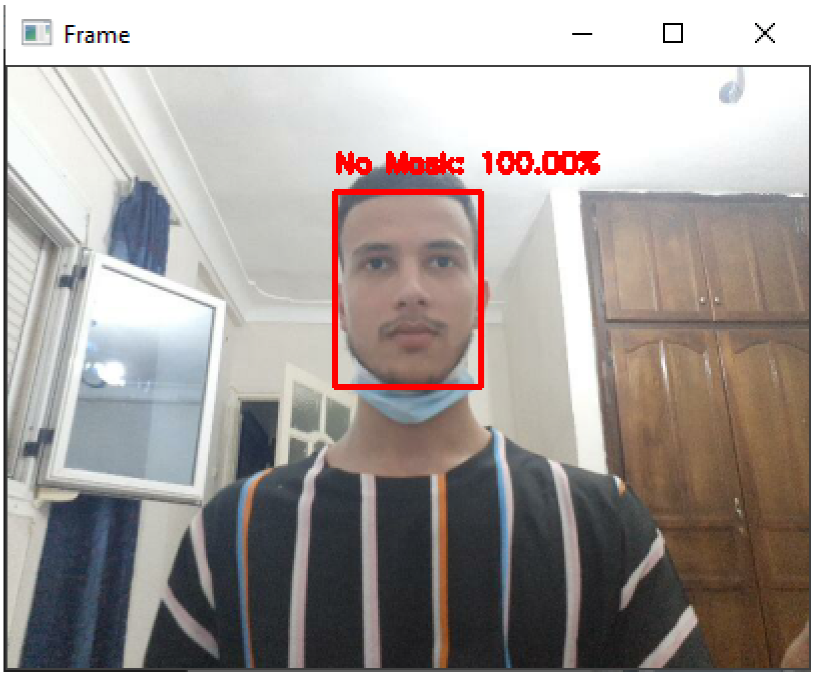


Figure IV.13: Live feed frame when an individual is not wearing a mask

If the mask is lowered, the percentage can be lowered so it matches the probability of a mask covering the face's lower part. For example if the mask is at the level of the mouth it will detect an average accuracy.

Otherwise, if the invividuals are wearing a mask a green label will pop with the probability of it's prediction next to the label. Indicating the percentage when the whole face is covered with a hundred percent like in the Figure IV.14.

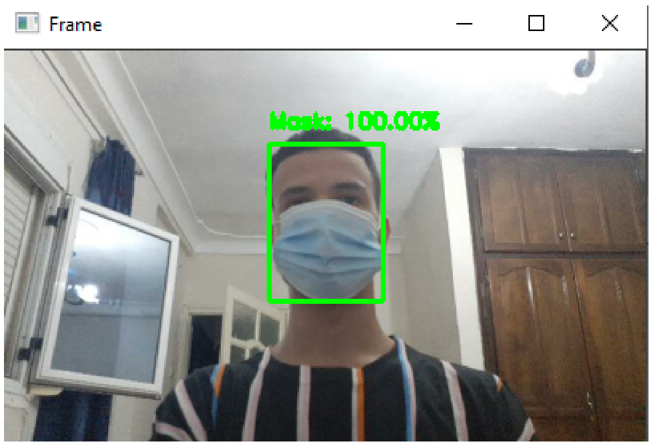


Figure IV.14: Live feed frame when an individual is wearing a mask

However, a common thing that happens is that individuals might be wearing a mask but, this covering fabric does not fully extend, which our model succesfully detects but with a lower precision confidence.

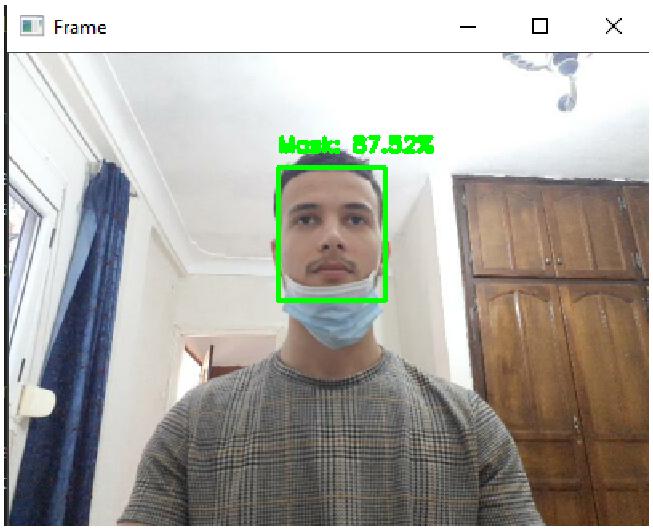


Figure IV.15: Live feed frame when an individual is partially wearing a mask

IV.8 Conclusion

After a trial and error process done with several methods and libraries, we could safely assure that generating a model with the MobileNetV2 classifier and two Conv2D layers is one of the best approaches possible to a face-mask detection system. We also concluded that Image Data Augmentation (and a dataset's size in general), can greatly increase the learning process of a ML model. Video detection then can be applied by looping over frames in real-time which can be used to detect masks in a real life environment.

General conclusion

In this project, we were interested in creating a system that allows to identify people wearing masks. This particularity will allow corporate companies and enterprises to have a solution to the current problem Covid-19 has presented us with.

Firstly, we started by studying the current history of facial detection and what it revolves around, we've made sure to mention the different types of facial detection method an shed some light on the advantages of each one. Then a deep dive was made in the theoretic notions used in the facial detection of our individuals , where an introduction to neural networks and machine learning has been made.

Secondly, a reduction of our research radius has been made, where a focus on mask-wearing individuals was pointed out. This latter included notions such as AI frameworks where we've compared each of Custom vision as well as Tensorflow. Talking about these frameworks allowed us to mention the graphic libraries that are availble in this day and age such as OpenCV, where a study was carried out to further research the prediction confidence of each library this step allowed us to have a better understanding on the need of filtrating our dataset. Once our dataset established we've concluded from the various detection classifiers (MobileNetV2 being a solid candidate) and optimizers that are available to us that using CNN is the way to go for this system.

Which brings us to the third chapter of this research, where we've managed to divide our model into two phases, the first one includes loading our dataset into the system whilst the second one, is about loading the face mask classifier model which will allow us to apply our chosen classifier. Our system's procedure is then layed-out in four different distinct steps: acquiring our dataset, learning and modeling it, developing and training our model then finally deploying our model for live feeds such as webcams. It seems that using the CNN architecture along functions from the Keras machine learning library, is a an important step since it allows for faster results when tuned using MobileNetV2.

Then, on the fourth chapter we went on the implementation phase where we applied the theoric notions and methods explored in past chapters, along with attempts and trials made with a variety of ML techniques. This has led us to conclude on a model generated with an addition of the MobileNetV2 classifier and two Conv2D layers that gave us the better results.
The accuracy of the neural network on image datasets is almost always over 97 percent, which can be improved in the future if we decide to deprecate MobileNetV2 or not. The model we generated can be used by various government agencies in crowded areas such as markets, airports, railway stations, and other crowded areas to ensure that people follow safety precautions. This model could be embedded with an alarm or buzzer in the future, and IoT could be implemented using deep learning CNN. When a person is detected not wearing a mask, the device will sound an alarm. If this model is implemented correctly, it will assist in ensuring human safety in this global pandemic.

Bibliography

[1] Tikoo, Smriti Malik, Nitin. (2016). Detection, Segmentation and Recognition of Face and its Features Using Neural Network. Journal of Biosensors Bioelectronics. 7. 1-5. 10.4172/2155-6210.1000210.

[2] A.J.Goldstein, L.D. Harmon, and A.B Lesk, "Identification ofHuman Faces," Proc. IEEE, May 1971, Vol. 59, No. 5, 748-760.

[3] Paul, Liton Suman, Abdulla. (2012). Face recognition using principal component analysis method. International Journal of Advanced Research in Computer Engineering Technology (IJARCET). 1. 135-139.

[4] M.Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, March 1991.

[5] Eleyan, Alaa. (2009). Face Recognition from Still Images and Video Sequences.

[6] Blackburn, Duane Bone, Mike Phillips, P. Jonathon. (2001). Face Recognition Vendor Test 2000: Evaluation Report. Natl Inst Stand Technol. 3. 71.

[7] Yang, Ming-Hsuan Kriegman, David Ahuja, Narendra. (2002). Detecting Faces in Images: A Survey. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 24. 34 - 58. 10.1109/34.982883.

[8] Kang, Min-Joo Kang, Jewon. (2016). Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. PLOS ONE. 11. e0155781. 10.1371/journal.pone.0155781.

[9] https://docs.microsoft.com/ What is Custom Vision ? (Consulted the 07/03/2021)

[10] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (2017) - Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam.

[11] Adam: A Method for Stochastic Optimization (2014) - Diederik P. Kingma, Jimmy Ba.

[12] Adaptive Subgradient Methods for Online Learning and Stochastic Optimization (2011) - John Duchi, Elad Hazan, Yoram Singer.

[13] C.Jagadeeswari, M.Uday Theja, Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers, International Journal of Advanced Science and Technology Vol. 29, No. 11s, (2020), pp. 3074-3082.

[14] S. Ge, J. Li, Q. Ye and Z. Luo, "Detecting Masked Faces in the Wild with LLE-CNNs," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 426-434, doi: 10.1109/CVPR.2017.53.

[15] https://machinelearningmastery.com/softmax-activation-function-with-python/ (Consulted the 13/03/2021)

[16] https://www.dsxhub.org/ (Consulted the 14/03/2021)

[17] https://www.python.org/doc/essays/blurb/ (Consulted on 18/4/2021)

[18] https://numpy.org/doc/stable/user/whatisnumpy.html (Consulted on 18/4/2021)

[19] I. B. Venkateswarlu, J. Kakarla and S. Prakash, "Face mask detection using MobileNet and Global Pooling Block," 2020 IEEE 4th Conference on Information Communication Technology (CICT), 2020, pp. 1-5, doi: 10.1109/CICT51604.2020.9312083.