A Project Report

on

Face MASK DETECTION

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

C.Ramesh Kumar

Assistant Professor

Submitted By

SUMIT KUMAR YADAV	18SCSE1180078
PRIYANKA CHAUDHARY	18SCSE1180006

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA

DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"FACE MASK DETECTION"** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Science and Engineering** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the July 2021 to Dec 2021, under the supervision of C.Ramesh Kumar Assistant Professor , Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Sumit Kumar Yadav(18SCSE1180078)
Priyanka Chaudhary(18SCSE1180006)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

C.Ramesh Kumar(Assistant Professor)

CERTIFICATE

The Final Project Viva-Voce examination of Sumit Kumar Yadav (18SCSE1180078) & Priyanka Chaudhary(18SCSE1180006) has been held on DEC 21 and his/her work is recommended for the award of **Bachelor of Technology in Computer Science and Engineering.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: DECEMBER, 2021

Place: Greater Noida

Statement of Project Report Preparation

1. Thesis title: Face Mask Detection .
2. Degree for which the report is submitted: Bachelor of Technology
3. Project Supervisor was referred to for preparing the report.
4. Specifications regarding thesis format have been closely followed.
5. The contents of the thesis have been organized based on the guidelines.
6. The report has been prepared without resorting to plagiarism.
7. All sources used have been cited appropriately.
8. The report has not been submitted elsewhere for a degree.

Sumit

Name: Sumit Kumar Yadav(18SCSE1180078)

Priyanka Chaudhary(18SCSE1180006)

TABLE OF CONTENTS

ABSTRACT

COVID-19 pandemic caused by novel corona-virus is continuously spreading until now all over the world. The impact of COVID-19 has been fallen on almost all sectors of development. The healthcare system is going through a crisis. Many precautionary measures have been taken to reduce the spread of this disease where wearing a mask is one of them. In this project, we propose a system that will find out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras. The novel Coronavirus had brought a new normal life in which the social distance and wearing of face masks plays a vital role in controlling the spread of virus. But most of the people are not wearing face masks in public places which increases the spread of viruses. This may result in a serious problem of increased spreading. Hence to avoid such situations we have to scrutinize and make people aware of wearing face masks. Humans cannot be involved for this process, due to the chance of getting affected by corona. Hence here comes the need for artificial intelligence(AI), which is the main theme of our project. Our project involves the identification of persons wearing face masks and not wearing face masks in public places by means of using image processing and AI techniques and sending alert messages to authority persons. The object detection algorithms are used for identification of persons with and without wearing face masks which also gives the count of persons wearing mask and not wearing face mask and Internet of Things (IOT) is utilized for sending alert messages. The alert messages are sent to the authority persons through mobile notification and Email. Based on the count of persons wearing and not wearing face masks the status is obtained. Depending upon the status warning is done by means of using buzzer and LED’s.

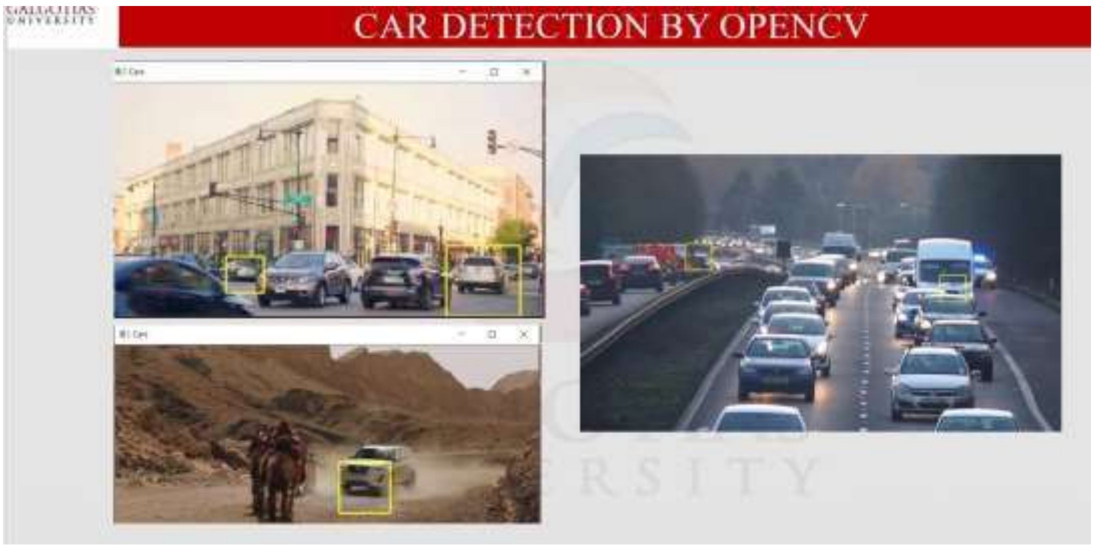
INTRODUCTION

The novel coronavirus covid-19 had brought a new normal life. India is struggling to get out of this virus attack and the government implemented lockdown for the long way. Lockdown placed a pressure on the global economy. So the government gave relaxations in lockdown . Declared by the WHO that a potential speech by maintaining distance and wearing a mask is necessary. The biggest support that the government needs after relaxation is social distancing and wearing of masks by the people.But many people are getting out without a face mask this may increase the spread of covid-19. Economic Times India has stated that " Survey Shows that 90 percent Indians are aware, but only 44 percent wearing a mask ". This survey clearly points that people are aware but they are not wearing the mask due to some discomfort in wearing and carelessness. This may result in the easy spreading of covid-19 in public places. The world health organization has clearly stated that until vaccines are found the wearing of masks and social distancing are key tools to reduce spread of virus. So it is important to make people wear masks in public places. In densely populated regions it is difficult to find the persons not wearing the face mask and warn them. Hence we are using image processing techniques for identification of persons wearing and not wearing face masks. In real time images are collected from the camera and it is processed in Raspberry Pi embedded development kit. The real time images from the camera are compared with the trained dataset and detection of wearing or not wearing a mask is done. The trained dataset is made by using machine learning technique which is the deciding factor of the result. The algorithm created by means of using a trained dataset will find the persons with and without wearing face masks. The Internet of Things (IOTs) can be used for connecting objects like smartphones, Internet TVs, laptops, computers, sensors and actuators to the Internet where the devices are linked together to enable new forms of communication between things and people, and between things themselves. Intimation messages are sent to authority persons by means of using IOT . As the spread of the virus occurs through physical contact, conventional recognition systems (such as fingerprints) or typing a password on a keyboard become insecure. Thus, facial recognition systems are the best option, as they do not require physical interaction as in other cases. However, the use of the face mask within these systems has represented a great challenge for artificial vision , because at the time of facial recognition, half of the face is covered and several essential data are lost. This clearly denotes the need to create algorithms that recognize a person when they are wearing a face mask . This has made it necessary to implement new strategies to achieve robustness in the current systems . In this sense, convolutional neural networks (CNN) belong to a set of techniques grouped under the so-called deep learning . Thus, over the years, this technology has been adapted to the needs of the human being, as established in , developing applications in various fields of knowledge, such as agriculture , military area , and medicine , among others. The contribution of this type of neural network has also been applied to analyze dental images, and this is technically described in the review of . In , a system that analyzes medical images is proposed, through selective data sampling, that detects hemorrhages in color images. On the other hand, in , a technical review of the contributions of the CNN in the mammographic breast cancer diagnosis (MBCD) is shown. Although there are several related investigations, they are still in the initial stages, with the clear objective of providing robust tools in the future. In , a review is described that seeks to identify the chronological advancement of CNN in brain magnetic resonance imaging (MRI) analysis. Although its use in medicine is not recent, it has now been directed particularly to applications related to COVID-19 . Various methods and procedures are used for the diagnosis of this disease, and one of them is the review of computed tomography scans. For this reason, suggests a rapid and valid method based on AI (10 CNN) for the detection of pulmonary affections. The results show a sensitivity, specificity, and precision of over 99%. Similarly, in a system for automatic disease diagnosis using the EfficientNet architecture is described. The results denote an average accuracy of over 96%, validating the contribution in situations of health crisis. Taking X-rays is another way to identify the virus affectations in the patient's chest Given this, in a deep learning method based on nCOVnet networks is presented for detection, the results of which show an accuracy of between 98% and 99%. Something similar is done in , where chest X-ray images are analyzed and the various training techniques of the networks are compared, obtaining an accuracy of 98.33% when using ResNet-34. Although in most cases CNNs are used in the diagnosis of COVID-19, they can also be used in other applications, as part of contagion prevention measures In a system is presented that allows people to be monitored when entering and being inside a certain place, and to evaluate if they are complying with the established biosecurity measures. In the event that this is not complied with, other people can be informed to exercise caution and health personnel to apply the respective measures. They have also been used to develop detection systems for the proper use of face masks. For this reason in a system is proposed that differentiates the people who use a mask or not with the algorithms RCNN, Fast RCNN, and Faster RCNN with an accuracy of 93.4%. In the VGG-16 CNN model is used to implement a detection system with an accuracy rate of 96%. Similarly, in they propose the SSDMNV2 model based on the MobileNetV2 architecture, which has an accuracy of 92.64% when performing the experimental tests. On the other hand, describes a system for the detection of face masks using a support vector machine (SVM) algorithm. The datasets are the Real-World Masked Face Dataset (RMFD), the Simulated Masked Face

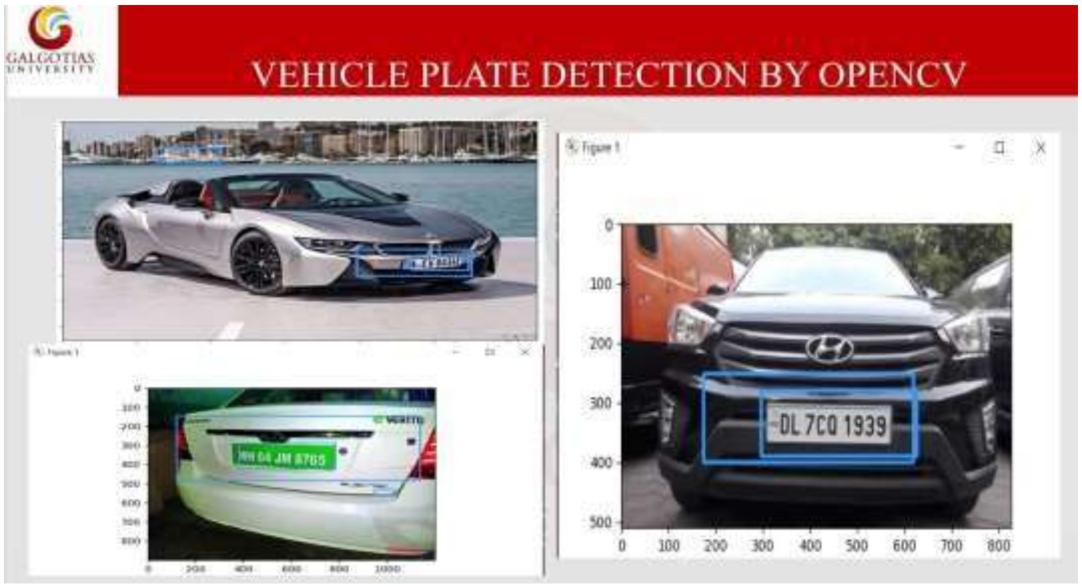
Dataset (SMFD), and the Labeled Faces in the Wild (LFW). The results show an accuracy of 99.64% with SVM in RMFD, 99.49% in SMFD, and 100% in LFW. In InceptionV3 transfer learning is used, obtaining an accuracy of 99.92% during training and 100% during tests with SMFD data. In a method to identify the correct use of masks is defined by combining classification networks and super-resolution of images (SRCNet). An accuracy of 98.70% is achieved, surpassing conventional image classification methods of this type. The problem of facial recognition due to the use of face masks during the COVID-19 pandemic has caused new horizons to be explored in artificial intelligence, representing a challenge for researchers, which has motivated the development of ocular recognition systems, as a parallel response. In a facial recognition system using eye information and CNN trained by ImageNet is presented. The results present an accuracy of between 90–95%. Similarly provides a facial recognition system using SVM with three databases (UBIPr, Color FERET, and Ethnic Ocular). Performance tests show a yield of approximately 92%. In continuity with the works described in the bibliography, this document presents a facial recognition system for people regardless of whether they use a face mask or not. For this purpose, the work has been organized into four sections, Section 2 contains the materials and methods, Section 3 shows the results, and Section 4 presents the discussion.Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision is consists of various aspects such as image recognition, object detection, image generation, image super-resolution , Facial Expression Recognition, Real time object detection, Generate synthetic image, Detect Pedestrians on Videos , Cars Moving Detection on Videos, Car's Plate Detection to detect the no. plate of car and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars.

Applications of Computer Vison:

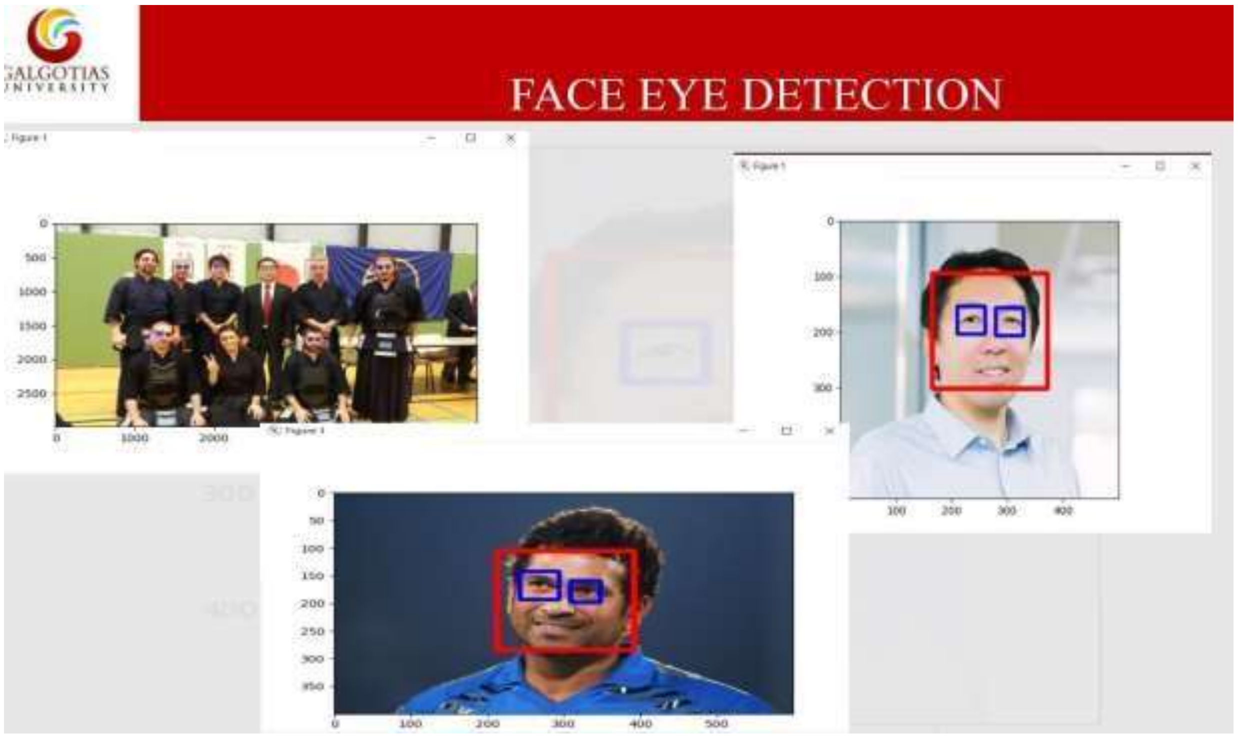
- Car Detection



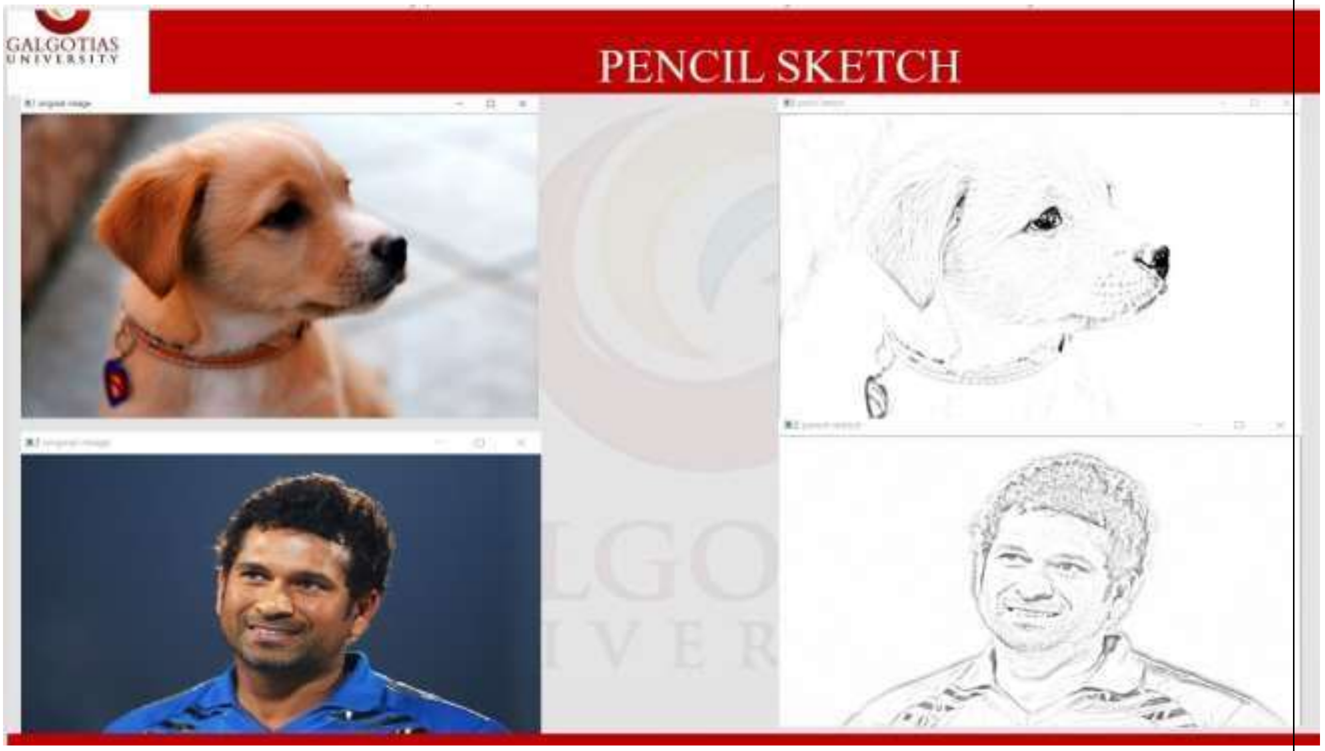
- Vehicle Plate Detection



- Face Eye Detection



- Pencil Sketch



An object detection system finds objects of the real world present either in a digital image or a video, where the object can belong to any class of objects namely humans, cars, etc.

In order to detect an object in an image or a video the system needs to have a few components in order to complete the task of detecting an object, they are a model database, a feature detector, a
hypothesiser and a hypothesiser verifier. This is a review of the various techniques that are used to detect an object, localize an object, categorize an object, extract features, appearance information, and many more, in images and videos.

Now in our project we propose a system that will find out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras. While a person without a mask is detected, necessary action is taken.

Now what it can do is it can capture the image of a person at that instant who is not wearing masks and send it to the authority , it can inform the authorities via any mail service . It can be used in homes, so that if any person is not wearing masks then the doors will not open .

LITERATURE REVIEWS

Machine learning for image classification

In the content based image classification using deep learning , Joseph Redmon et.al proposed You Only Look Once (YOLO) algorithm for real time object detection. Sanzidul Islam et.al 2020, gave a deep learning based assistive System to classify COVID-19 Face Mask which is implemented in rasbperrypi-3. Velantina et.al 2020, made an COVID-19 facemask detection by means of using Caffe model. Senthilkumar et.al 2017, compared the two most frequently used machine learning algorithms K-Nearest Neighbour and Support Vector Machine in his work for face recognition. Senthilkumar et.al 2018, proposed a new and fast approach for face recognition**.**

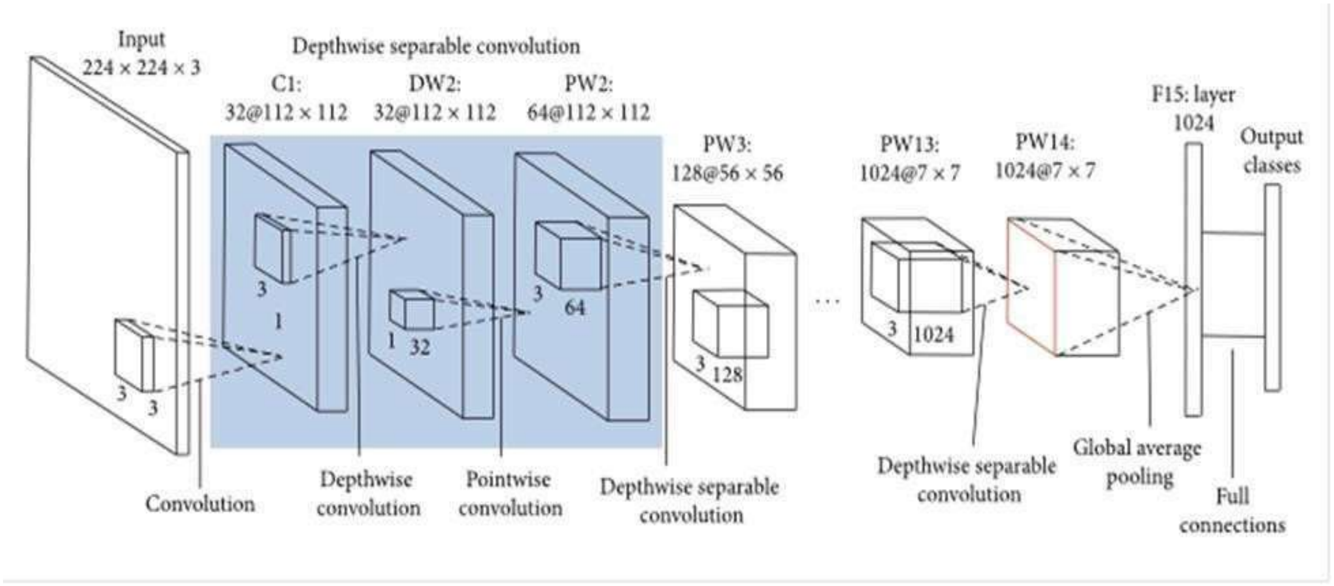
Internet of Things

Luigi Atzori et.al reviewed different versions of the Internet of Things are reported and corresponding enabling technologies. Lu Tan et.al and Neng Wang discussed the Future internet in their work. Feng Xia et.al and others discussed briefly about the Internet of Things, 2012 in their work.

IOT device and Machine Learning

Yair Meidan et.al 2017, has implemented nine IOT devices and treated each IOT device as separate classes. For classification purposes, deep learning techniques were used. Yair Meidan et.al and Michael Bohadana et.al 2017, proposed a security system for detection of unauthorized IoT devices using machine learning techniques. Liang Xiao et.al 2018, has improved the IoT Security techniques based on machine learning using Artificial Intelligence concept. With this, based on the above literature surveys, we have made a new deep learning algorithm for face mask detection. The details are elaborated in forthcoming chapters.

In our project, we are using the MobileNet v1 algorithm for classifying the image. MobileNet Architecture



The MobileNet model is designed to be used in mobile applications, and it is TensorFlow's first mobile computer vision model.

MobileNet uses **depthwise separable convolutions.** It significantly **reduces the number of parameters** when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

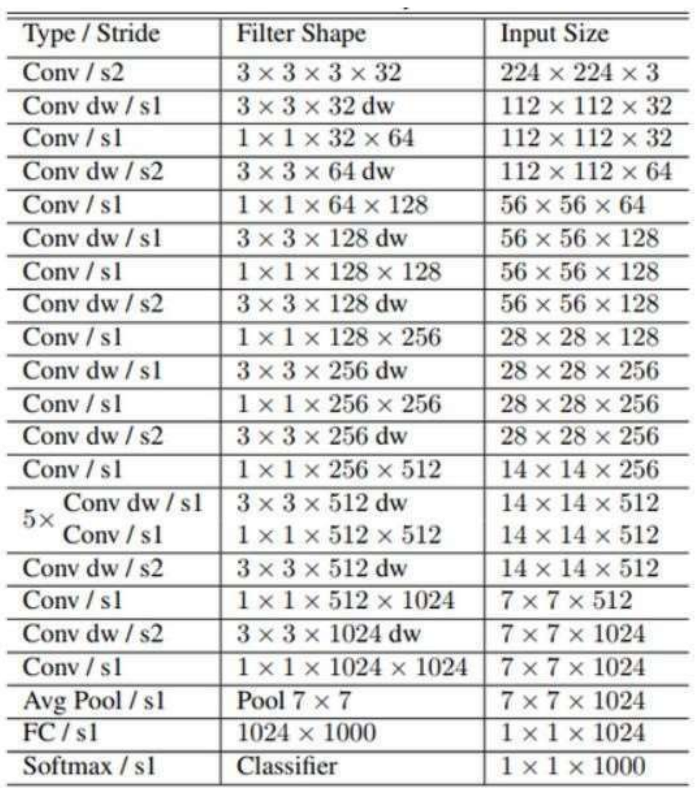
A depthwise separable convolution is made from two operations.

1. **Depthwise convolution.**

2. **Pointwise convolution**.

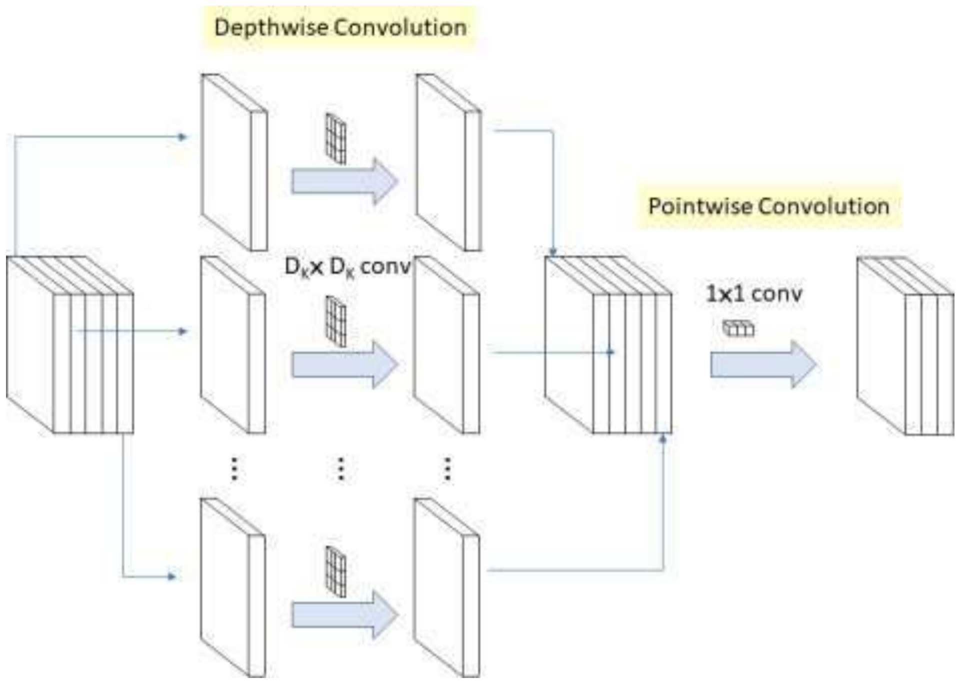
MobileNet gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.

Architecture of MobileNet



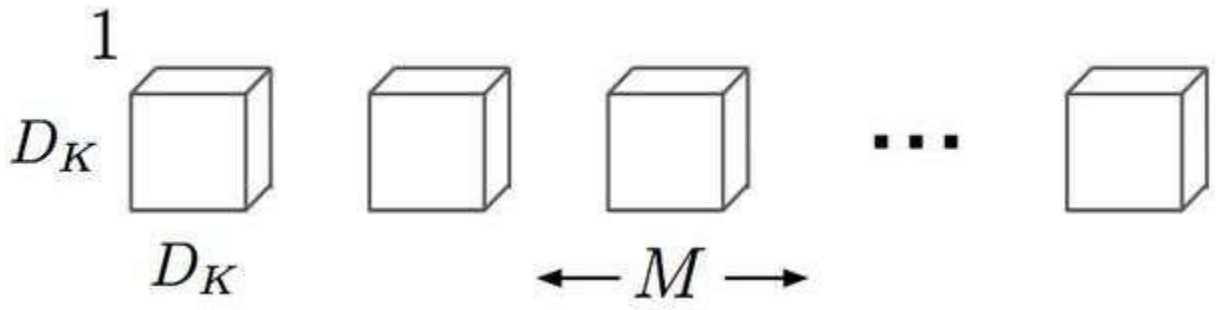
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Depthwise separable convolution is **a depthwise convolution followed by a pointwise convolution** as follows:



Depthwise Separable Convolution

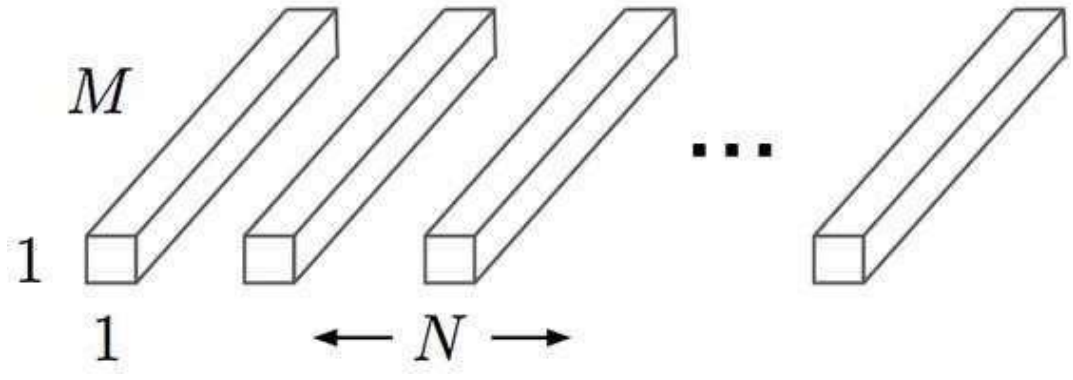
1. **Depthwise convolution** is the **channel-wise DK×DK spatial convolution**. Suppose in the figure above, and we have five channels; then, we will have 5 DK×DK spatial convolutions.
2. **Pointwise convolution** is the **1×1 convolution** to change the dimension.
3. **Depthwise convolution**



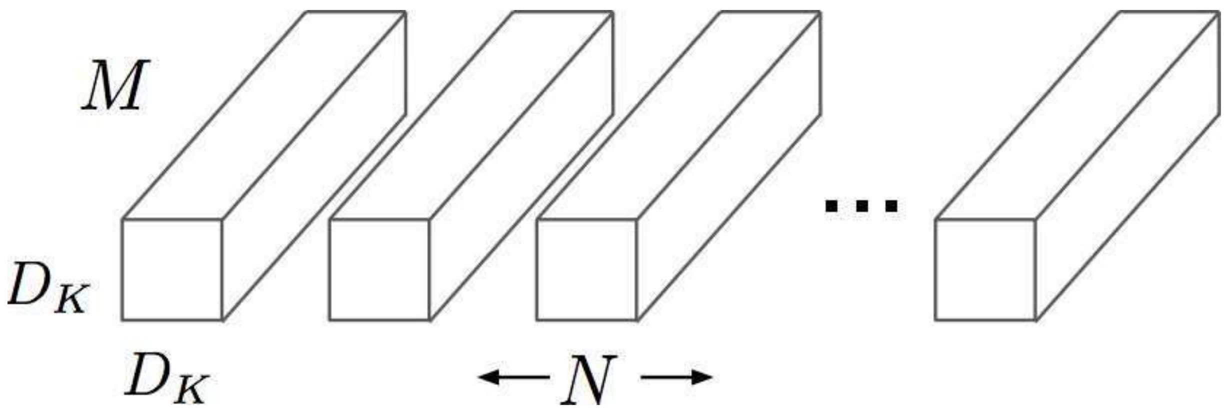
1. **Depthwise convolution.**

It is a map of a single convolution on each input channel separately. Therefore its number of output channels is the same as the number of the input channels. Its computational cost is
Df^2 * M * Dk^2.

2. **Pointwise convolution.**



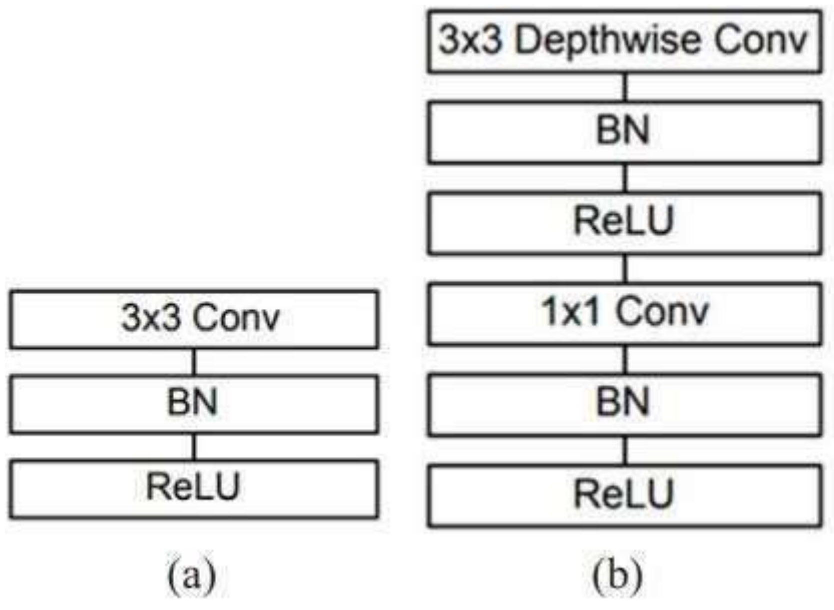
Convolution with a kernel size of 1x1 that simply combines the features created by the depthwise convolution. Its computational cost is
M * N * Df^2.



Difference between Standard Convolution and Depthwise separable convolution.

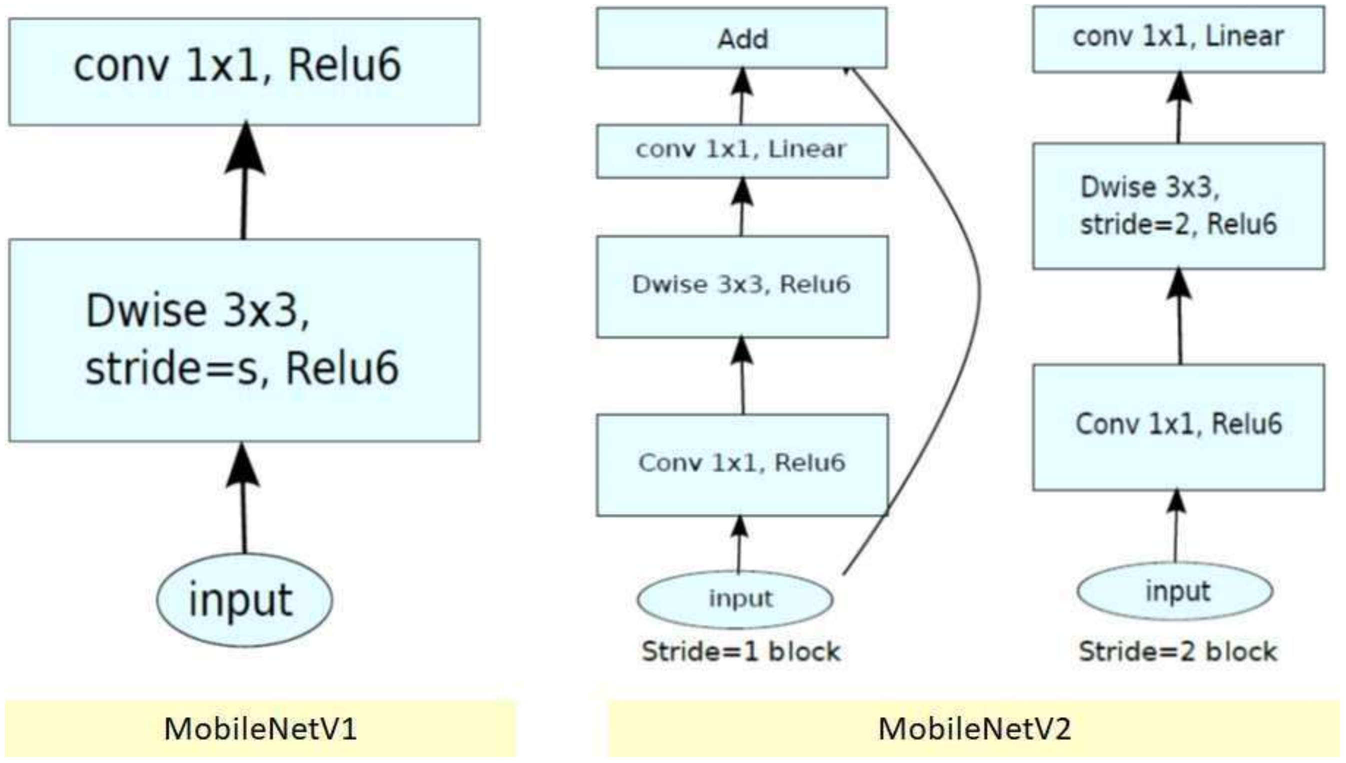
Standard Convolution

The main difference between MobileNet architecture and a traditional CNN instead of a single 3x3 convolution layer followed by the batch norm and ReLU. Mobile Nets split the convolution into a 3x3 depth-wise conv and a 1x1 pointwise conv, as shown in the figure.



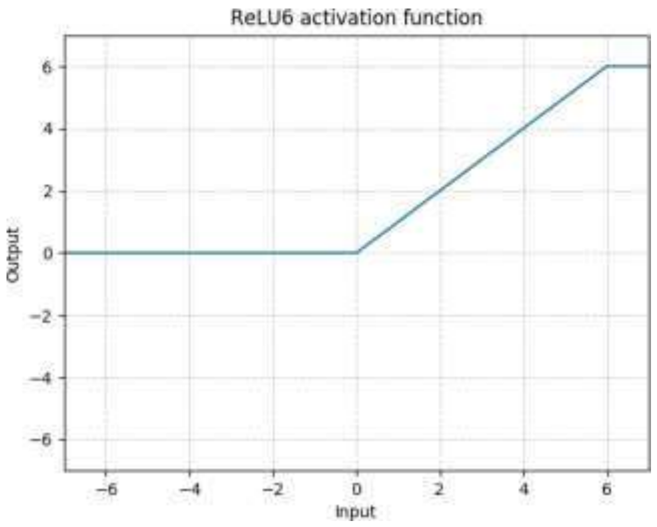
In MobileNetV1, a better module is introduced with inverted residual structure. Non-linearities in narrow layers are removed.

1. MobileNetV1 Convolutional Blocks



In MobileNet V1 there are two layers:

The **first layer** is called a **depthwise convolution**, it performs lightweight filtering by applying a single convolutional filter per input channel. The **second layer** is a **1×1 convolution**, called a **pointwise convolution**, which is responsible for building new features through computing linear combinations of the input channels. **ReLU6** is used here for comparison.

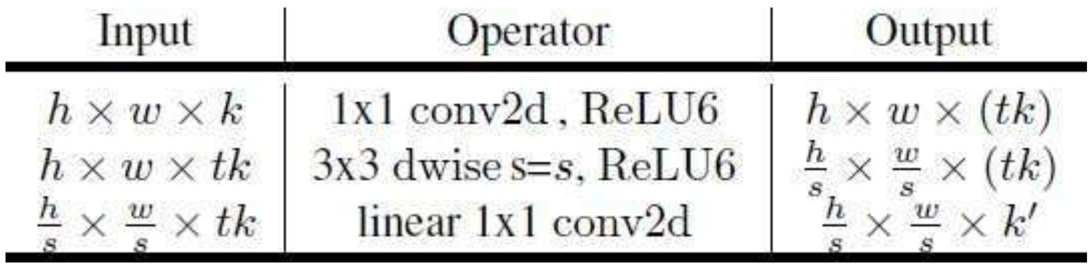


ReLU6

- ★ ReLU6 is used due to its robustness when used with low-precision computation, based on MobileNetV1.

MobileNetV1

- ★ In MobileNetV1, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing.
- ★ There are 3 layers for both types of blocks.
- ★ This time, the **first layer** is **1×1 convolution with ReLU6.**
- ★ The **second layer** is the **depthwise convolution**.
- ★ The **third layer** is another **1×1 convolution but without any non-linearity.** It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non- zero volume part of the output domain.



Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

- ★ And there is an expansion factor *t*. And *t*=6 for all main experiments.
- ★ If the input got 64 channels, the internal output would get 64×*t*=64×6=384 channels.

MODULES AND HARDWARE REQUIREMENT

★ Using opencv library.

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

★ Using keras.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

★ Using numpy.

NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

★ Using matplotlib.

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.

★ Using sklearn.

The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

★ Using tensorflow.

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

★ Using imutils.

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images.

HARDWARE AND SOFTWARE REQUIREMENTS:

- OS = Windows 10
- RAM = 8GB
- Processor= i5 or i7 gen.
- Software : Visual Studio ,

Spyder , Jupyter etc. Need of:

Dataset containing images of people having masks or not having masks (helps in training the Model).

Benefits

YOLO is a popular object detection algorithm because it achieves high accuracy while it is also able to run in real-time. The algorithm "only looks once" at the image means that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression it then gives the recognized objects along with the bounding boxes. In YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO directly optimizes detection performance since it trains on full images. YOLO has a number of benefits over other object detection methods they are-

• YOLO is extremely fast

• YOLO scans the entire image during training and also during testing. So, it implicitly encodes contextual information about classes as well as their appearance.

• YOLO learns generalizable representations of objects so that when it is trained on natural images and tested , the algorithm performs excellently when compared to other top detection methods.

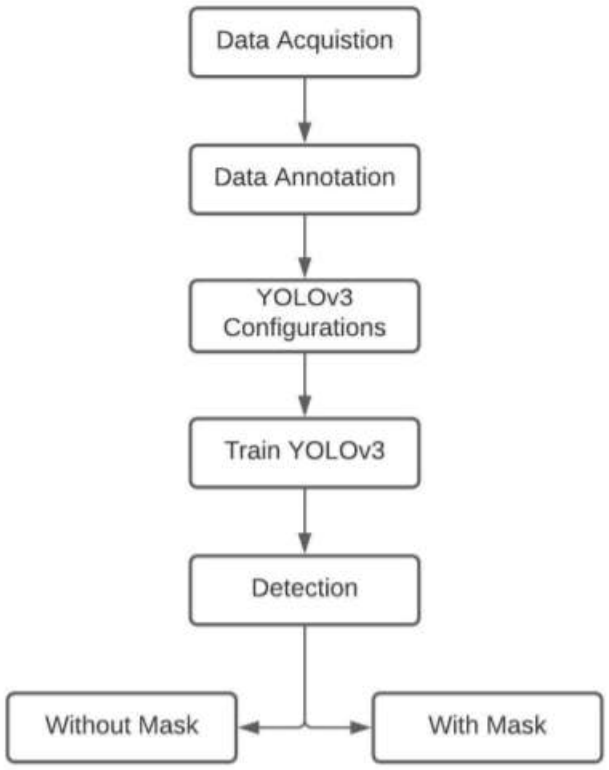


Fig3.1

Workflow

Here the workflow of YOLO object detection algorithm is discussed in detail. Initially a dataset of images is collected which are used for training by means of using YOLO. Dataset consists of images of persons with masks and without masks. figure 3.1 shows the work flow of YOLO.

Data Acquisition

Data is really important for deep learning techniques. If we use more data for training the AI then the result will be better. To train YOLO we need more data and with proper annotation. Using a web-scraping tool we have collected 900 images of both mask and no-mask. These images cannot be used directly so we need to pre-process before feeding into the model. Next step is Data Annotation.

Data Annotation

To train YOLO we need to annotate images for object detection models. Our dataset should be well annotated. There are different types of annotations available. Here a bounding boxes method is used. It creates a rectangle area over images that are present in our dataset. Since Annotation needs more time we are using a tool called LabelIMG to annotate our data.

YOLOv3 Configuration

The YOLOv3 configuration involved the creation of two files and a custom Yolov3 cfg file. YOLOv3 configuration first creates a ”obj.names” file which contains the name of the classes which the model wanted to detect. Then a obj.data file which contains a number of classes in here is 2, train data directory, validation data, ”obj.names” and weights path which is saved on the backup folder. Lastly, a cfg file contains 2 classes. figure 3.2 shows the configuration steps involved. Next is training of our YOLOv3 in which an input image is passed into the YOLOv3 model. This will go through the image and find the coordinates that are present. It divides the image into a grid and from that grid it analyzes the target objects features. Here 80 percent data is used for training , and remaining 20 percent is used for validation. Now weights of YOLOv3 trained on the dataset are created under a file. Using these trained weights now we can classify the persons wearing and not wearing the mask.

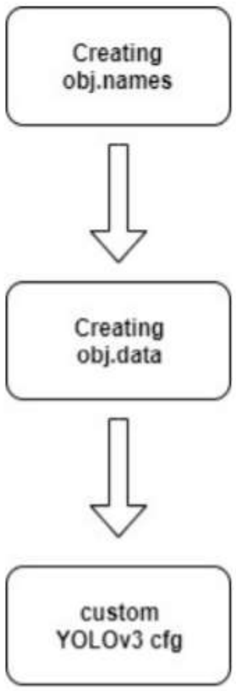


Fig3.2

Face Mask Detection Algorithm

Step 1: Start the program.

Step 2: Input image is feeded.

Step 3: YOLOv3 trained weights are loaded from the disk.

Step 4: Persons with and without face mask are detected by means of object detection algorithm.

Step 5: After detection resultant image is displayed along with count of Persons with and without masks.

Step 6: The ratio of with and without face mask is calculated and based upon ratio status is obtained.

Step 7: Based on status output LED and buzzer connected to Raspberry pi will be activated.

Step 8: Resultant image is saved in Raspberry pi for identification

Material and METHODOLOGY

Description of the Problem

The effects of COVID-19 on the global economy can be seen with the naked eye, as the confinement of people in the homes brings with it less production and slows down the commercial dynamism. However, it should be noted that in situations of health crisis such as the one that continues to be experienced, it is relevant to put people's health before any productive activity. That is why biosecurity measures and social distancing protocols have been implemented to limit the spread of this dangerous virus. As well as the capacity in public institutions, industries and other establishments has been limited, highlighting the so-called telework (in certain cases). Thus, companies have implemented various methodologies, strategies, and techniques to protect the integrity and health, both when entering and staying in face-to-face work sessions. As previously mentioned, CNN have been an important technological tool during this pandemic. Although most approaches have been taken towards the diagnosis of the disease, monitoring and prevention has also been covered. Today, the use of a personal face mask is a mandatory preventive measure. Keeping the mouth, nose, and cheeks covered has now made people only recognizable by their eyes, eyebrows, and hair, which is a problem for the human eye, which tends to find similarities in several faces that have similar features. This problem also affects computer systems, as facial recognition systems are now very common. They are used to unlock the smartphone, access sensitive applications, and to enter certain places. Current systems usually process information from the entire face of the person, which is why technology must adapt to these new conditions. All this is done with the purpose of maintaining the biosecurity of the user, but giving them the opportunity to continue with the activities as naturally as possible. The literature has shown that there are systems that seek to identify whether people use it properly. These works have had very good results. However, facial recognition using biosecurity material has not yet been explored. All of this motivated the present investigation, in which a detection system with two approaches is presented. The first is to develop a face classifier, starting from a database of people with and without a mask. The second describes a facial recognition algorithm in controlled environments, which allows for personnel to be identified automatically, without removing the face mask. This can be implemented as an access system to an institution or a home, but at a low cost. This is ensured by using open-source programming software and simple features that reduce computational expense. For this reason, the possibility of improving the adaptability of current facial recognition systems, in the face of new circumstances, has been established as a starting hypothesis.

Requirements

The programming language used here is Python. For optimal operation, a highprocessing equipment (GPU) is needed. However, we received no external financing, so we chose to work with free Google servers, which are available in Google Colab. Another of the essential requirements is to have the necessary databases in order to carry out the training and obtain the classification and recognition models. Taking into account that building the database of these databases requires a high investment of time when working with artificial intelligence and especially with convolutional neural networks. Additionally, it is necessary to develop a consent form for the people who will allow for taking photographs for the facial recognition algorithm database. This is necessary because there are currently no databases for the recognition of people with face masks. For this, it is necessary to rely on Art 6.1d of the European General Data Protection Regulation (RGPD), in connection with article 46.3 of the LOU. Here, it is mentioned that the data of a person will be treated in accordance with the exercise of public powers conferred on the universities as responsible for the treatment of the data of the students. As well as biometric data ((article 9.2.a) of the RGPD), in which consent will be needed so that it can be part of the exams where facial recognition techniques are applied. The collection, filing, processing, distribution, and dissemination of these information data will require the authorization of the owner and the mandate of the law.

System Development

It is proposed to design a system that is capable of identifying a person's face, even if it is with or without a mask. For the system to work properly, it is necessary to use two databases: the first is for classifier training and consists of a large number of images of people who wear a face mask and others who do not. The second is used for training the facial recognition system, and here there are people with and without the biosafety material (face mask). The input data are obtained either from an image, or a video and the architecture used is MobileNet, with the aim of having a better precision and robustness. This project is divided into three stages, which are described below.

First Stage

This stage focuses on finding the location and dimension of one or more faces, regardless of whether or not they wear a mask, within an image. For this, the OpenCV Deep Learning-based face detection model is used and, as a result, the region of interest (ROI) is obtained, which contains data such as the location, width, and height of the face.

Second Stage

A diagram of the operation of the second stage is shown in Figure 1. This is where the classifier training is performed to detect faces with a mask and without a mask. For this, the "Real-World-Masked-Face-Dataset" database available on Git-Hub is used. Unzipping the files makes available a large number of images of people of Asian origin wearing a mask. From this database, the training of the classifier of the first stage is carried out.

Third Stage

Once the face of the person has been identified, in the third stage, facial recognition is carried out, for which a set of own observations is used that is built based on the faces of various people. For the construction of the set of observations, a balance is sought in terms of gender, namely, five women and five men from whom the images are obtained. Figure shows the set of faces using a mask and Figure shows without a face mask.

Training of Facial Recognition Models

The procedure to obtain the images is as follows: (i) during a week, daily videos of the face of each individual are obtained, seeking to capture different angles and different environmental conditions (lighting changes). (ii) From the videos obtained, images are captured at different moments in order to build a set of observations with images. (iii) The images where the face is not found in the entirety are eliminated. At the end of the procedure, a total of 13,359 images are obtained; 7067 (52.9%) with a face mask and 6292 (47.1%) without a face mask. In practice, acquiring so many images of a face requires a short video recording of a few seconds showing different views of the face. Regarding the identification of the images used for the recognition of people, the images have been labeled from left to right, as follows:

- Women: W1, W2, W3, W4, and W5 (Vicky, Mela, Damaris, Ale, and Yaritza, respectively).
- Men: M1, M2, M3, M4, and M5 (Oscar, Jorge, Pablo, John, and Jonathan respectively).

In this way, when a person is recognized in an image, the name information can be accessed and placed in the image. Therefore, once the necessary data have been obtained, the recognition model is trained. The two facial recognition models follow the architecture of Figure 6, which is briefly explained below. To do this, from this set of observations, the facial recognition model has two approaches: 1. The first model aimed at recognizing people using a face mask. 2. The second model aimed at recognizing people not using a face mask.

Database:

Set of observations of the faces of people using a mask and without using a mask for both approaches of the third stage.

Preprocessing:

For the facial recognition model of people using a face mask, only 3/5 of the upper part of the face is extracted. This in order to discriminate the mask that the person is using (as in the experimental tests, this information is not useful for the recognition model). Whereas, for the facial recognition model without a face mask, the image of the full face is used. In both cases, the resulting image is scaled to a size of 164 × 164 pixels. Characteristics extraction: The FaceNet model is used to extract the most essential characteristics of the face. This model extracts the most essential features from the input image, in this case a face, and returns a vector of 128 features. The input of the network is an image with a human face and, using a deep metric learning technique, it calculates the metric to generate vectors of real characteristics [17]. This network is a model belonging to PyTorch, which can generate neural networks similar to Caffe . The image is inserted into the network, then passes through the neural network and obtains the embedding of each face represented by f(x)   R d . This method attempts to ensure that the image of a specific person (xi a) is closer to all of the images of the same person (xi p) and away from images of other people (xi n). Equation (1) shows the calculation of the loss (L), where α is the margin applied between positive and negative pairs [54]. It receives an image of 164 × 164 pixels as the input data, and a vector of characteristics of 128 elements called “face embedding” is obtained at the output.

L = N ∑ i h ||(x a i) − f(x p i)||2 2 − || f(x a i) − f(x n i)||2 2 + α i (1)

ANN classification:

Once the vector of characteristics has been obtained, any machine learning classification model can be applied; in this case, a feedforward multilayer perceptron is chosen. A feedforward multilayer perceptron (ANN) is chosen, because it does not need a large amount of data, it has been demonstrated that an ANN is a universal approximator with excellent results, and it does not require as much computational cost. After investigating the literature, it is best to use convolutional neural networks, but at this stage it is not necessary, as the best characteristics have been extracted by FaceNet and it only uses a vector of characteristics, instead of a raw image.

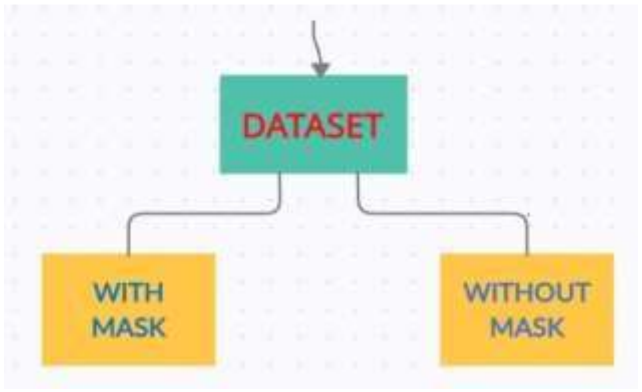
The architecture of our simple neural network applied to the two approaches is as follows:

• 128 neurons in the initial layer (size of the feature vector—face embedding)

- 100 neurons in a hidden layer with built-in ReLu activation
- 50 neurons in a hidden layer with built-in ReLu activation
- 24 neurons in a hidden layer with built-in ReLu activation
- 10 neurons in the output layer with a Softmax activation function
- The following configurations are used for classifier training:
- Epochs: 15
- Batch Size: 32
- Optimizer: Adam
- Loss Function: MSE

First, we will do the Data Preprocessing of our dataset.

Our dataset contains around 700 images of each class i.e people wearing masks and not wearing masks in 2 separate folders with mask and without mask.



We can say that it is a Binary Classification Problem. With masks:

Without Mask

In this process , we will resize each image with 224 X224px and convert it to array and then scale it to range of [-1,1].

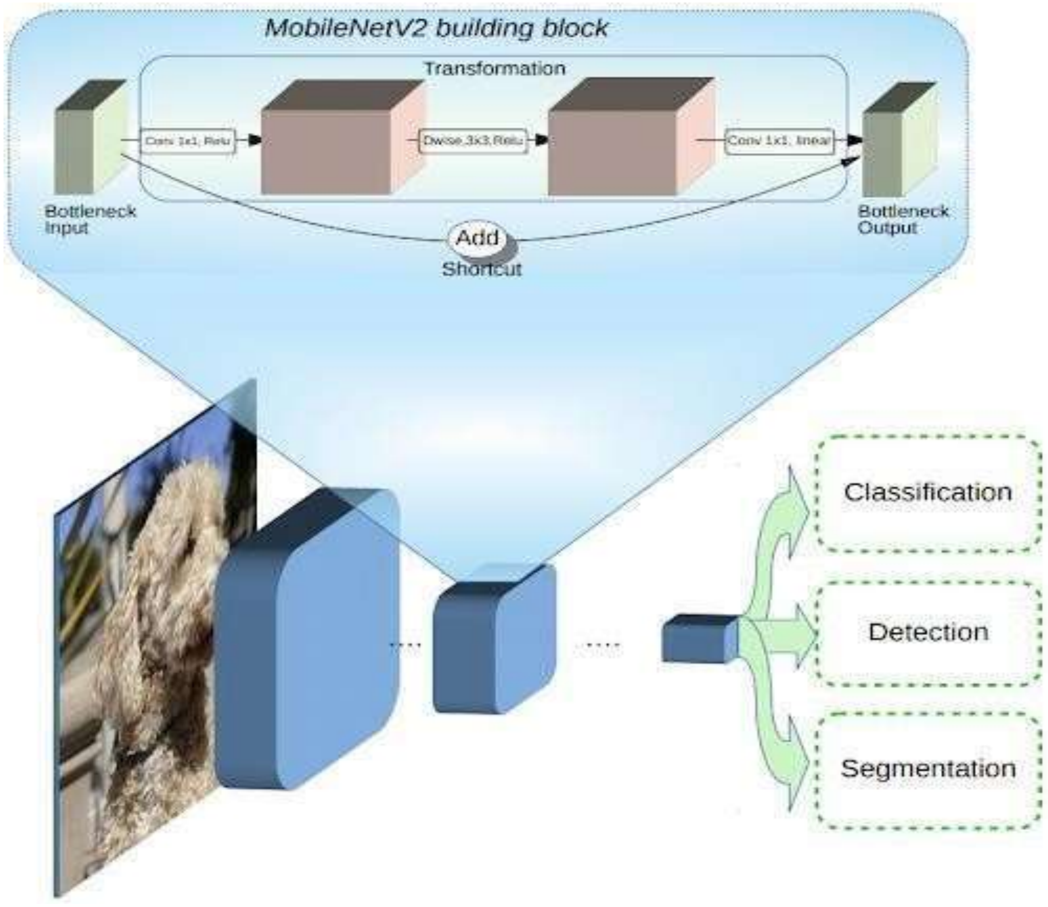
Then perform one-hot encoding to convert our classes to binary format.

Libraries required :

- Tensorflow
- Imutils
- opencv
- pandas
- numpy
- malplotlib.

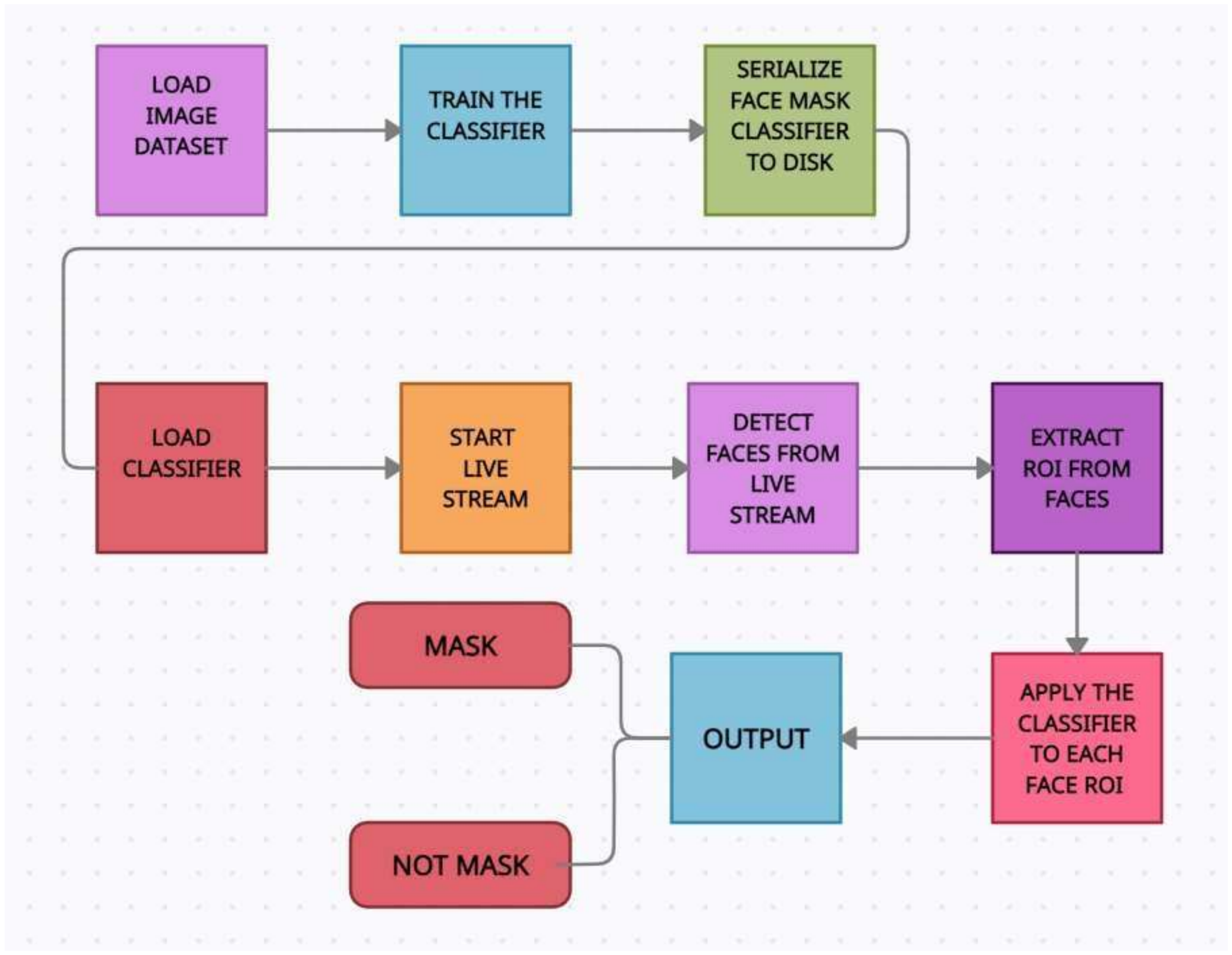
By this we will get our transformed dataset.

Then, we will train our model based on this dataset using MobileNet v1 algorithm.



Then we will detect faces with masks and without masks.

Architecture Diagram



FEASIBILITY STUDY

We can use it in a webcam or any CCTV to detect the person wearing masks or not wearing masks and takes necessary action against them.

We use Convolutional Neural Network and Deep Learning for Real Time Detection and Recognition of Human Faces, which is simple face detection and recognition system is proposed in this paper which has the capability to recognize human faces in single as well as multiple face images in a database in real time with masks on or off the face. Pre-processing of the proposed frame work includes noise removal and hole filling in colour images. After pre-processing, face detection is performed by using CNNs architecture. Architecture layers of CNN are created using Keras Library in Python. Detected faces are augmented to make computation fast. By using Principal Analysis Component (PCA) features are extracted from the augmented image. For feature selection, we use Sobel Edge Detector.

1.The Input Image

Real-time input images are used in this proposed system. Face of person in input images must be fully or partially covered as they have masks on it. The system requires a reasonable number of pixels and an acceptable amount of brightness for processing. Based on experimental evidence, it is supposed to perform well indoors as well as outdoors i.e. passport offices, hospitals, hotels, police stations and schools etc.

2.The Pre-processing Stage

Input image dataset must be loaded as Python data structures for pre-processing to overturn the noise disturbances, enhance some relevant features, and for further analysis of the trained model. Input image needs to be pre-processed before face detection and matching techniques are applied. Thus pre-processing comprises noise removal, eye and mask detection, and hole filling techniques. Noise removal and hole filling help eliminate false detection of face/ faces. After the pre-processing, the face image is cropped and re-localised. Histogram Normalisation is done to improve the quality of the pre- processed image.

3.The Face Detection Stage

We perform face detection usingHAAR Cascade algorithm.This system consists of the value of all black pixels in greyscale images was accumulated. They then deducted from the total number of white boxes. Finally, the outcome is compared to the given threshold, and if the criterion is met, the function considers it a hit.In general, for each computation in Haar-feature, each single pixel in the feature areas can need to be obtained, and this step can be avoided by using integral images in which the value of each pixel is equal to the number of grey values above and left in the image.

Feature =ie{1..N}wi.RecSum(x, y,w,h),

where RecSum (x, y, w,h) is the summation of intensity in any given upright or rotated rectangle enclosed in a detection window and x, y,w,h is for coordinates, dimensions, and rotation of that rectangle, respectively. Haar Wavelets represented as box classifier which is used to extract face features by using integral image

4.The Feature-Extraction Stage

Feature Extraction improves model accuracy by extracting features from pre-processed face images and translating them to a lower dimension without sacrificing image characteristics. This stage allows for the classification of human faces.

5.The Classification Stage

Principal Component Analysis(PCA) is used to classify faces after an image recognition model has been trained to identify face images. Identifying variations in human faces is not always apparent, but PCA comes into the picture and proves to be the ideal procedure for dealing with the problem of face recognition. PCA does not operate classifying face images based on geometrical attributes, but rather checks which all factors would influence the faces in an image. PCA was widely used in the field of pattern recognition for classification problems.PCA demonstrates its strength in terms of data reduction and perception.

6.Training Stage

The method is based on the notion that it learns from pre- processed face images and utilizes CNN model to construct a framework to classify images based on which group it belongs to. This qualified model is saved and used in the prediction section later. In CNN model, the stages of feature extraction are done by PCA and feature selection done by Sobel Edge Detector and thus it improves classification efficiency andaccuracy of the training model.

7.Prediction Stage

In this stage, the saved model automatically detects theofthe face maskimagecaptured by the webcam or camera. The saved model and the pre-processed images are loaded for predicting the person behind the mask. CNN offers high accuracy over face detection, classification and recognition produces precise and exactresults.CNN model follows a sequential model along with Keras Library in Python for prediction of human faces.

Future Work

In this work of face mask detection we have used YOLOv3 to detect the persons with face mask and without face mask with good efficiency and and sent an intimation message to authority persons by means of IOT. It’s performance is really well in images and our detection results were also quite good. This detection can also be used for video stream or camera fed inputs. To get improved performance and speed, Raspberry Pi of higher variant such as 4GB or 8GB RAM can be used to implement the detection algorithm. The Future development of the project is planned to involve the identification of a person and sent the intimation message to the persons mobile who were not wearing face masks. This can be implemented in offices and institutions by means of training the database with employees images or students images and by means of face recognition the person is identified by which the mobile number and other details of the person is obtained from database and hence it will be easy to notify that particular person or useful for taking any actions regarding not wearing face mask.The proposed model can also be enhanced by means of including various parameters like peoples count, social distance and temperature measurement. This project will be very helpful and can be implemented in hospitals, airports, schools, colleges, offices, shops, malls, theaters, temples, apartments etc. and can also be implemented for Covid free event management.

PYTHON PROGRAM

About Python:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python 3.0 was released in 2008. Although this version is supposed to be backward incompatible, later on many of its important features have been back-ported to be compatible with version 2.7.

Features:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Libraries:

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs. The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

Code

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
from cv2 import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
	# grab the dimensions of the frame and then construct a blob
	# from it
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
		(104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the face detections
	faceNet.setInput(blob)
	detections = faceNet.forward()
	print(detections.shape)

	# initialize our list of faces, their corresponding locations,
	# and the list of predictions from our face mask network
	faces = []
	locs = []
```

```
preds = []

# loop over the detections
for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence
        if confidence > 0.5:
                # compute the (x, y)-coordinates of the bounding box for
                # the object
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                # ensure the bounding boxes fall within the dimensions of
                # the frame
                (startX, startY) = (max(0, startX), max(0, startY))
                (endX, endY) = (min(w - 1, endX), min(h - 1, endY))









                # extract the face ROI, convert it from BGR to RGB channel
                # ordering, resize it to 224x224, and preprocess it
                face = frame[startY:endY, startX:endX]
                face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                face = cv2.resize(face, (224, 224))
                face = img_to_array(face)
                face = preprocess_input(face)

                # add the face and bounding boxes to their respective
                # lists
```

```
				faces.append(face)
				locs.append((startX, startY, endX, endY))




	# only make a predictions if at least one face was detected
	if len(faces) > 0:
		# for faster inference we'll make batch predictions on *all*
		# faces at the same time rather than one-by-one predictions
		# in the above `for` loop
		faces = np.array(faces, dtype="float32")
		preds = maskNet.predict(faces, batch_size=32)

	# return a 2-tuple of the face locations and their corresponding
	# locations
	return (locs, preds)

# load our serialized face detector model from disk
prototxtPath = r"C:/Users/sumit/OneDrive/Desktop/Project 2/deploy.prototxt"
weightsPath = r"C:/Users/sumit/OneDrive/Desktop/Project
2/res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
print("Starting video stream...")
vs = VideoStream(src=0).start()


# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 500 pixels
	frame = vs.read()
```

```
        frame = imutils.resize(frame, width=500)

        # detect faces in the frame and determine if they are wearing a
        # face mask or not
        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

        # loop over the detected face locations and their corresponding
        # locations
        for (box, pred) in zip(locs, preds):
                # unpack the bounding box and predictions
                (startX, startY, endX, endY) = box
                (mask, withoutMask) = pred

                # determine the class label and color we'll use to draw
                # the bounding box and text
                label = "Mask" if mask > withoutMask else "No Mask"
                color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

                # include the probability in the label
                label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

                # display the label and bounding box rectangle on the output
                # frame
                cv2.putText(frame, label, (startX, startY - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
                cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

        # show the output frame
        cv2.imshow("Mask Detector", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

# do a bit of cleanup
cv2.destroyAllWindows()
```

```
vs.stop()
 ########################################################################
# various libraries used
import numpy as np
import matplotlib.pyplot as plt
import os
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,img_to_array,load_img
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Input,Dense,Dropout,AveragePooling2D,Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from imutils import paths

# initializing the initial learning rate, no. of epochs and batch size

INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = [r,"C:\Users\sumit\OneDrive\Desktop\dataset"]
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("Loading Images..........")

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
 for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)

        data.append(image)
        labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
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)

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
        rotation_range=20,
        zoom_range=0.15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest")

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

# construct the head of the model that will be placed on top of the
# the base model
```

```
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will not be updated
during the first training process
for layer in baseModel.layers:
        layer.trainable = False




# compile  model
print("Compiling Model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
        metrics=["accuracy"])

# train the head of the network
print("[Training head...")
H = model.fit(
        aug.flow(trainX, trainY, batch_size=BS),
        steps_per_epoch=len(trainX) // BS,
        validation_data=(testX, testY),
        validation_steps=len(testX) // BS,
        epochs=EPOCHS)

# make predictions on the testing set
print("Evaluating Network...")
```

```
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
	target_names=lb.classes_))

# serialize the model to disk
print("Saving mask detector model...")
model.save("mask_detector.model", save_format="h5")


# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

Result

The model is successfully deployed can be seen in fig .3 and fig. 4.
The experiments carried out have the purpose of demonstrating the potential use of the system, so tests are carried out using the recall metrics, Precision, F1-score, and the corresponding macro avg and weighted avg. The objective of using these metrics is to evaluate the system from different perspectives. Recall and precision indicate the ability of the model to correctly detect true positives. Recall also considers the false negatives detected, and the precision of the false positives detected by the model. False positives, in this case of face detection with masks, occur when an object is labeled as a face. For example, the system frames a plant as a face with a face mask, as it is false that a plant is a positive face. The reasons this can occur in our system are numerous, which is why hard work is needed in order to collect a large database so that the model being trained can better distinguish the desired classes (faces). False negatives occur when a face is not detected in the first stage, because the face has covered areas that make classification difficult; in this proposal, this initial classifier is a pre-developed tool. On the other hand, the F1-score provides a global measure of the system's performance, it is a combination of precision and recall (in a single value), with 0 being low performance and 1 being the best possible performance (all cases detected correctly). By considering the macro avg metric, sd can get a general idea of the average of all of the experiments, while the weighted avg establishes an average measure of all of the experiments, but considering the number of observations of each class. Thus, in the event that a class has a higher score, the final weighted avg score will not be affected by it, but will give a value of importance to each score depending on the amount of observation. When considering these metrics, what is sought is to verify the robustness of the method by classifying both classes. In the second stage, the face classifier training with a mask and faces without a mask takes a period of approximately 10 h, and in the third stage the extraction of face embeddings takes approximately 5 h and the ANN-based classifier training takes 10 min.

CONCLUSION

The designed algorithm was effectively able to detect the person wearing masks or not wearing masks and also shows the probability of detection. This prototype system allows for the facial recognition of people with and without a mask, and could be used as a low computational consumption proposal for personnel access control. The two models of this system are tested with images, thus achieving better precision and optimization for each model. The face of someone found in the database is successfully classified to provide the name tag and probability of success. The three stages of the system allowed for the relevant characteristics of a person’s face to be extracted, and thus use a simple neural network for the classification task. In this sense, the use of a “Face Embeddings” as input to the neural network obtained satisfactory results in the experiments carried out. During the training of the third stage, it is possible to notice that there is an overadjustment, this fact is due to the fact that the database is built for this stage with a few participants, although it is composed of several images, and does not exist much variability. However, the system shows potential to be used in differentiated facial recognition applications. It should be noted that if a face is not found in the database, it will be detected, but the tag “mask” or “no mask” will be added, which refers to whether the person is wearing a mask. It should be considered that the level of confidence used in the system to accept if a face belongs to someone is 0.6. When defining whether people are wearing a mask or not, the accuracy is 99.65%. When evaluating the facial recognition model with the test data of people who do not use a mask, an accuracy of 99.96% is obtained, and for those who use a mask, an accuracy of 99.52% is obtained. In this way, the basis for future research that can expand the study in this field is established. In the bibliographic review, the use of MATLAB has been evidenced as an alternative proposal that could provide a lower number of false positives that should be evaluated. It is also proposed to investigate new extraction architectures that can be compared with FaceNet, and to thus choose the best one. One important thing to note is that the system has difficulty detecting certain faces when wearing a mask. This problem is due to the fact that initially, the Open CV Deep Learning based face detector was being used and it is not designed to detect faces with masks. It could also be observed that the face recognition stage is not robust when the detected face presents a certain angle of inclination. However, this is not a problem of great impact, as this application is oriented to access control and at this point, the person must maintain a firm and straight posture in front of the device that acquires the image. Therefore, as a future work, merging the first and second stages in the same model and creating an own algorithm that directly searches for a face with and without a mask should be considered. This avoids first using the Open CV face detector and then classifying faces with and without mask. This will further reduce the processing time and make the model more robust. In addition, it is proposed that in the future, a comparative study of the models used for the transfer of learning can be carried out in order to determine the best model and network trained in

unfavorable evaluation conditions. Once the final models have been trained, they can be compressed and deployed on low-cost embedded devices such as Raspberry Pi or mobile devices.

REFERENCES

1.Adjabi, I.; Ouahabi, A.; Benzaoui, A.; Taleb-Ahmed, A. Past, present, and future of face recognition: A review. Electronics 2020, 9, 1188. [CrossRef]

2. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. Comput. Intell. Neurosci. 2018, 2018. [CrossRef]

3. Chakraborty, B.K.; Sarma, D.; Bhuyan, M.K.; MacDorman, K.F. Review of constraints on vision-based gesture recognition for human-computer interaction. IET Comput. Vis. 2018, 12, 3–15. [CrossRef]

4. Ko, B.C. A brief review of facial emotion recognition based on visual information. Sensors 2018, 18, 401. [CrossRef] [PubMed]

5. Egger, M.; Ley, M.; Hanke, S. Emotion Recognition from Physiological Signal Analysis: A Review. In Proceedings of the Electronic Notes in Theoretical Computer Science, Yangzhou, China, 14–17 June 2019; Elsevier B.V.: Larnaca, Cyprus, 2019; Volume 343, pp. 35–55. [CrossRef]

6. Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. Appl. Sci. 2019, 9, 909. [CrossRef]

7. Dang, K.; Sharma, S. Review and comparison of face detection algorithms. In Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering, Noida, India, 12–13 January 2017; pp. 629–633.

8. Cook, C.M.; Howard, J.J.; Sirotin, Y.B.; Tipton, J.L.; Vemury, A.R. Demographic Effects in Facial Recognition and Their Dependence on Image Acquisition: An Evaluation of Eleven Commercial Systems. IEEE Trans. Biom. Behav. Identity Sci. 2019, 1, 32–41. [CrossRef]

9. Dargan, S.; Kumar, M. A comprehensive survey on the

biometric recognition systems based on physiological and behavioral modalities. Expert Syst. Appl. 2020, 143, 113114. [CrossRef]

10. Jeon, B.; Jeong, B.; Jee, S.; Huang, Y.; Kim, Y.; Park, G.H.; Kim, J.; Wufuer, M.; Jin, X.; Kim, S.W.; et al. A facial recognition mobile app for patient safety and biometric identification: Design, development, and validation. JMIR mHealth uHealth 2019, 7, e11472. [CrossRef]

11. Gonzalez-Sosa, E.; Fierrez, J.; Vera-Rodriguez, R.; Alonso-Fernandez, F. Facial soft biometrics for recognition in the wild: Recent works, annotation, and COTS evaluation. IEEE Trans. Inf. Forensics Secur. 2018, 13, 2001–2014. [CrossRef]

12. Galterio, M.G.; Shavit, S.A.; Hayajneh, T. A review of facial biometrics security for smart devices. Computers 2018, 7, 37. [CrossRef]

13. Karthik, K.; Aravindh Babu, R.P.; Dhama, K.; Chitra, M.A.; Kalaiselvi, G.; Alagesan Senthilkumar, T.M.; Raj, G.D. Biosafety Concerns During the Collection, Transportation, and Processing of COVID-19 Samples for Diagnosis. Arch. Med. Res. 2020, 51, 623–630. [CrossRef]

14. Souza, T.M.L.; Morel, C.M. The COVID-19 pandemics and the relevance of biosafety facilities for metagenomics surveillance, structured disease prevention and control. Biosaf. Heal. 2020, 3, 1–3. [CrossRef]

15..https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV1_In verted_Residuals_CVPR_2018_paper.html.

16..https://www.semanticscholar.org/paper/MobileNets%3A-Efficient-Convolutional-Neural-Networks-Howard-Zhu/3647d6d0f151dc05626449ee09cc7bce55be497e