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

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

Under The Supervisionof
Mr. P. RAJAKUMAR
Assistant Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1010737 – MD AQUIL ANWAR

18SCSE1010573 – MD FAIZAN

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 period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Mr. P. RAJAKUMAR, Assistant Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

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

18SCSE1010737-MD AQUIL ANWAR
18SCSE1010573 –MD FAIZAN

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

Supervisor

Mr P. RAJAKUMAR(Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1010737 MD AQUIL ANWAR, 18SCSE1010573 – MD FAIZAN** has been held on _____ 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:

Place:

ABSTRACT

In the light of the COVID 19 pandemic, the World Health Organization (WHO) has declared Use of face mask as an essential biosecurity measure. This caused a problem with the current one A face recognition system that motivates the development of this research. This manuscript explains Development of a system that recognizes people even when wearing a face mask Photo. Classification model based on Mobile Net V2 architecture and Open CV ,I am using a face detector. Therefore, you can use these stages to identify where your face is. Determine if they are wearing a face mask. Face Net model is used as a feature Extractor and multi-layer feedforward perceptron for face recognition. For training A face recognition model produces a series of observations from 13,359 images. 52.9% images 47.1% images with and without face mask. Experimental results show that Is 99.65% accurate in determining if a person is wearing a mask. accuracy 99.52% was achieved with face recognition of 10 people using masks, with face recognition Achieves 99.96% accuracy without a mask Effective strategies to contain the COVID-19 pandemic require attention to mitigate the negative impacts on local health and the global economy. In the absence of effective antiviral drugs and limited medical resources, WHO recommends a set of measures to control infection and prevent depletion of limited medical resources. Wearing a mask is one of the non-pharmacological interventions available to block the main cause of the SARSCoV2 droplet excreted by 4,444 infected people. Despite diverse medical resources and reports of masks, it is still necessary in all countries to open their mouths and cover their noses. The purpose of this document is to contribute to the health of the community by developing high-precision real-time technology that can effectively recognize mask less faces and make them wear masks in public places.

Table of Contents

List of Figures

List of Tables

Acronyms

SVM	Support Vector Maching
ML	Machine Learning
DL	Deep Learning
CNN	Convolution Neural Networks

CHAPTER-1

Introduction

In recent decades, face recognition has become the subject of 4,444 research worldwide. Also, with the advancement of technology and the rapid development of artificial intelligence, very important developments have been made in the field of intelligence. For this reason, SOEs use facial recognition systems to identify and control access.

Persons in airports, schools, offices and other places. Meanwhile, as the COVID-19 pandemic (a global pandemic) spread, government agencies have taken a number of biosecurity measures. Infection restriction rules. A must-have mask. Public places have proven effective in protecting users and those around them. The traditional perception that the virus is spread through physical contact For example, fingerprint Or, typing your password on the keyboard makes it insecure. Therefore, facial recognition systems are the best choice as they do not require physical interaction. In other cases. However, these systems use masks. Big problem with artificial vision,

is half in face recognition. Your face has been disguised and some important data has been lost. This clearly indicates the need to start.

Algorithm for recognizing a masked person ,This was necessary to implement a new strategy to achieve the stability of the existing system. In this sense, Convolutional Neural Networks (CNNs) are a collection of methods. Classified as so-called deep learning.

Thus, over the years, this generation has developed and adapted to human desires, as can be seen . agriculture, military, and pharmaceuticals . The contribution of such a neural community It has been further implemented for dental pixel studies, which is technically defined in the review. In a gadget was proposed to analyze scientific pixels using sampled data. Samples that detect bleeding in pix shades. On the other hand, technology Overview of CNN's contribution to mammography analysis of most breast cancers (MBCD) is displayed. Although there are numerous relevant investigations, nonetheless This is a preliminary step with the pure goal of providing reliable hardware in the future. The review decided to try to understand the chronological development of CNNs in the brain Magnetic resonance imaging (MRI) analysis. Although its use in medicine has not appeared recently, it is now particularly Packages related to COVID-19. using a variety of strategies and approaches Analysis of these disorders and each is a computed tomography review. scanning. For this reason, mainly refers to a fast and legitimate approach based entirely on AI. 1

Effect depends on sensitivity, specificity, and More than 99% accuracy. Similarly in a device for automatic analysis of violations, The Efficient Net framework is defined. Effects represent average accuracy. 96% identified a contribution to the fitness crisis. Another x-ray technique. Methods for detecting the effects of viruses inside a patient's chest. Taking this into account deep learning approaches, mainly based on n COV net networks, have been proposed for discovery. Among them, the display accuracy is 98% to 99%. A similar thing is done in. Analyzing chest x-rays and various educational networking strategies Compared to 98.33% accuracy using ResNet34. In most cases, CNNs are used within COVID19 analysis, but additionally It is used in various packaging as part of anti-pollution measures. Gadgets are provided for people to observe as they come in and are inside. Assess compliance with safe locations and connected biosafety systems measurement. Others will know if this is not respected. Warn and train staff to apply appropriate action. they additionally It was used to extend the sensing structure for correct use of the face mask. For this reason, In a device was proposed to distinguish between those who wear masks and those who no longer wear masks. RCNN, Fast RCNN and Faster RCNN algorithms with 93.4% accuracy. In the CNN version is used to enable high-precision beacons. 96%. Similarly. The dataset is the face of a Real World mask. Data Set (RMFD), Simulated Mask Face Data Set (SMFD), and Tagged Faces Wildlife (LFW). The effect shows 99.64%, 99.49% accuracy from SVM to RMFD. SMFD and 100% LFW. Explore the InceptionV3 switch to get: 99.92% accuracy at any point in training and 100% when checking SMFD data. from [51] An approach to recognizing the best use of masks is described using meshing class networks. and Fix Super resolution. Accuracy reached a higher 98.70%. An existing strategy for this type of graphics class. Face popularity problem due to mask use at some point COVID19 The pandemic has opened up new frontiers in synthetic intelligence research. Research projects that contributed to the rise in popularity of eyes Structure like parallel answer. In the gadget uses information about the eyes to determine the popularity of a face. A CNN with the technology to use ImageNet.

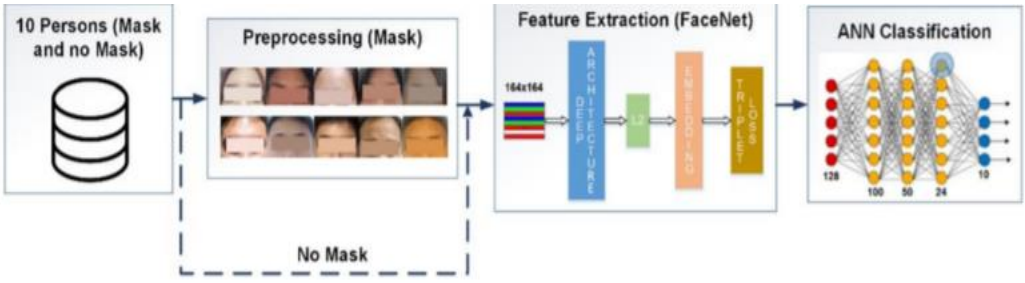
CHAPTER-2
Literature Survey/Project Design

The impact of COVID-19 on the global economy is visible with the naked eye. Locking people at home reduces production and slows it down. commercial dynamism. However, we need to be more careful with health crises. As always, it is important to put people's health first. all production activities. This is where biosecurity measures and social distancing protocols are in place to limit the spread of this dangerous virus. Its capabilities have been limited by government agencies, industry and other entities. The so-called remote work (in some cases). As a result, businesses receive methodologies, strategies and methods to protect their integrity and health. Come and have a multilateral meeting. As previously mentioned, CNNs have been an important technological tool during this pandemic. While the majority are approaching. Diagnosis of the disease, as well as monitoring and prevention was carried out. was covered

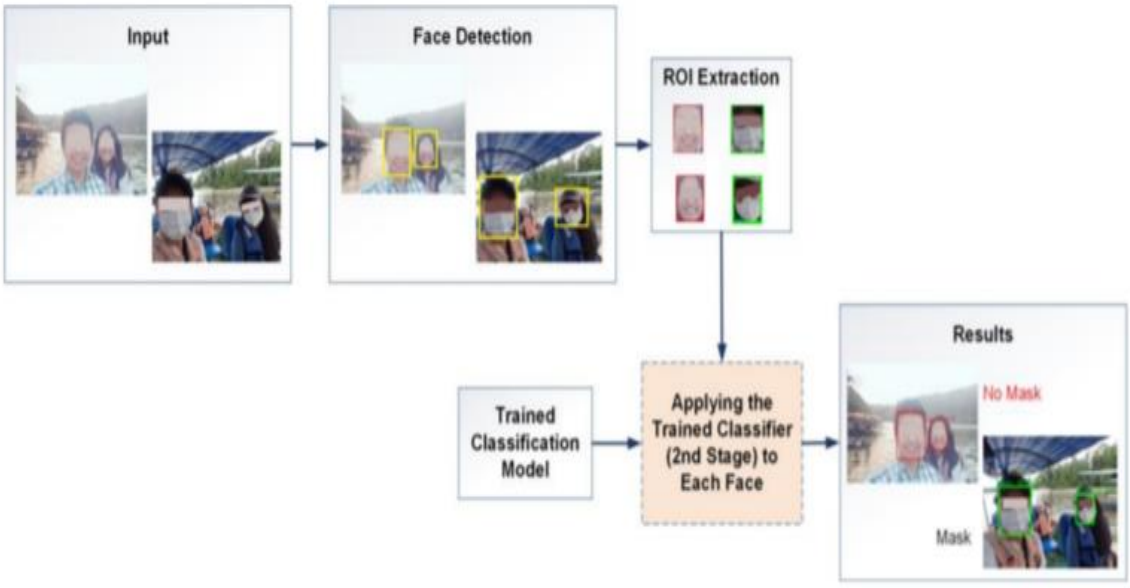
The use of personal masks is an important precaution today. Preserved Shaded Mouth, Nose, and Cheeks

Now people can only recognize them by their eyes, eyebrows and hair.

Problem with the human eye looking for similarities in multiple faces with similar features. This issue also affects computer systems in the following ways: Facial recognition systems are now very common. Used to unlock your smartphone, access confidential applications, and access specific locations. Current systems typically describe these new conditions as they can obtain information from the entire face of a person. All of this is done for the purpose of maintaining biosafety. Allow users to continue their activities in a natural way. Example: maybe. The literature shows that there are systems for determining: Use it correctly. These operations gave very good results. However, the use of facial recognition bioprotective agents has not yet been studied. All of this motivates the present. A study presenting a detection system with two approaches. First, we develop a face classifier based on a database of people with and without masks. Second describes the algorithm for face recognition in a controlled environment. The can automatically identify personnel without removing the mask, making it a system that can be used in institutions or homes, but at a lower cost. This is done using open source software and simple pricing features. For this reason, modern face recognition systems have been created in new situations with the potential to improve adaptability. as an initial hypothesis



The programming language you use is Python. High-performance hardware (GPU) is required for optimal performance. However, because it did not receive external funding,. Other The main requirement - the existence of a database necessary to meet Train and get classification and recognition models. taking into account that Creating a database of these databases is time consuming to work with. It uses artificial intelligence, specifically convolutional neural networks. You should also develop a consent form for who will allow you to: For taking pictures for face recognition algorithm database. necessary This is because there is currently no database that can identify the person wearing the mask. sugar Sustainable Development 2021,To do so, you must comply with Article of the General European Data Protection regulation, The data mentioned here are People will be treated according to the powers granted them. About the university responsible for processing student data. Also Consent is required for use as biometric ,It may be part of an exam where facial recognition technology is applied. collection, In order to store, process, distribute and distribute this informational data, it is necessary to: Owner's Permissions and Legal Rights. system development It is even proposed to develop a system that can recognize human faces. with or without a mask. For the system to work properly, you must use: Two databases: the first is for training the classifier and consists of a large number of images. Those who wear masks and those who do not. The second is used for training. There are facial recognition systems, some with and without biosafety. material (face mask). The input data comes from an image or video and It uses the Mobile Net architecture to improve accuracy and reliability. The project is divided into three phases, which are described below.



This step focuses on finding the position and size of one or more faces in the image, whether or not they are wearing a mask. For this OpenCV Deep Using a training-based face recognition model, the resulting region of interest (ROI) Received containing data such as position, width and height of the face. second step How the second stage works is shown in Figure 1. here it is. The classifier is trained to recognize faces with and without masks. for teeth, Use the "Real World Masked Face Dataset" database available on GitHub. unzip The file contains many images of Asian people wearing them. Mask. A first-stage classifier is trained on this database. The classifier uses the Mobile Net V2 architecture. Low latency and low parameterization ability. Improves mobile performance. Use the model in multiple jobs and tests to increase accuracy. it also saves Sustainable Development 2030,It is simple and does not require special operators to classify multiple images. Various detection tasks for mobile applications. 35 degrees higher MobileNetV2 in the field Combined with Single Shot Detector Lite, it detects compared to the first version. When the bottleneck is encoded it is more accurate and requires 30 parameters. Intermediate input and output as shown.

At the same time, inner layers encapsulate the model's ability to transform low-level concepts (pixels) into high-level descriptors (image categories). It can be used in Via Python's "Tensor flow" library and "Transfer learning" The last layer of a convolutional neural network. This architecture is It was chosen because it is a computationally efficient model. Therefore, it is efficient in terms of processing speed. 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. shows the set of faces using a mask and shows without a face mask.

Set of observations without using a facemask

CHAPTER-3
Functionality/Working of Project

In the new world of coronavirus, multidisciplinary efforts have been organized to slow the spread of the pandemic. The AI community has also been a part of these endeavors. In particular, developments for monitoring social distancing or identifying face masks have made-the-headlines. But all this hype and anxiety to show off results as fast as possible, added up to the usual AI overpromising factor (see AI winter), may be signaling the wrong idea that solving some of these use cases is almost trivial due to the mighty powers of AI. In an effort to paint a more complete picture, we decided to show the creative process behind a solution for a seemingly simple use case in computer vision: Detect people that pass through a security-like camera. Identify face mask usage. Collect reliable statistics (% people wearing masks). Note: the numbers above people are “votes” from a mask classifier: they should be green (positive) when the person has a mask, red (negative) if not, and 0 if undecided. First, we will speak about our unique challenge, as compared to other solutions. Then, we will show what it takes to reliably detect and count people in a camera feed. Expect some really fun stuff like pose estimation After, we will deep dive into how to identify whether each person is wearing a mask or not. We add up on this by showing how the fact that we are using video and not independent frames can help with a better decision. Lastly, we show actual results on a longer video, and brainstorm on next steps and possible improvements. In a nutshell, this system works by performing two main tasks: Object detection with a neural network (SSD), pre-trained for face detection. Output: list of bounding boxes around each detected face. Classification in two classes (with/without mask), using another neural net (MobileNetV2). Output: score from 0 to 1 signifying the probability of a face wearing a mask. The pre-trained face detection model seems to work great for his case, and it detects faces even when they are partially covered by masks. So, no need to re-train anything for the first model (object detector, SSD). As there was no pre-trained classifier to distinguish faces with and without masks, we trained this model with a limited dataset. in a very clever and effective way: we took a dataset with regular faces, and artificially added masks to some of the images. We leveraged other computer vision techniques to automatically place the masks over the faces.

How was our face mask dataset created?

Prajna, like me, has been feeling down and depressed about the state of the world — thousands of people are dying each day, and for many of us, there is very little (if anything) we can do.

To help keep her spirts up, Prajna decided to distract herself by applying computer vision and deep learning to solve a real-world problem:

Best case scenario — she could use her project to help others

Worst case scenario — it gave her a much needed mental escape

Either way, it's win-win!

As programmers, developers, and computer vision/deep learning practitioners, we can all take a page from Prajna's book — let your skills become your distraction and your haven.

To create this dataset, Prajna had the ingenious solution of:

Taking normal images of faces

Then creating a custom computer vision Python script to add face masks to them, thereby creating an artificial (but still real-world applicable) dataset

This method is actually a lot easier than it sounds once you apply facial landmarks to the problem.

Facial landmarks allow us to automatically infer the location of facial structures, including:

Eyes

Eyebrows

Nose

Mouth

Jawline

To use facial landmarks to build a dataset of faces wearing face masks, we need to first start with an image of a person not wearing a face mask:

To build a COVID-19/Coronavirus pandemic face mask dataset, we'll first start with a photograph of someone not wearing a face.

From there, we apply face detection to compute the bounding box location of the face in the image:

The next step is to apply face detection. Here we've used a deep learning method to perform face detection with OpenCV.

Once we know where in the image the face is, we can extract the face Region of Interest (ROI):

The next step is to extract the face ROI with OpenCV and NumPy slicing.

And from there, we apply facial landmarks, allowing us to localize the eyes, nose, mouth, etc.:

Then, we detect facial landmarks using dlib so that we know where to place a mask on the face.

Next, we need an image of a mask (with a transparent background) such as the one below:

An example of a COVID-19/Coronavirus face mask/shield. This face mask will be overlaid on the original face ROI automatically since we know the face landmark locations.

This mask will be automatically applied to the face by using the facial landmarks (namely the points along the chin and nose) to compute where the mask will be placed.

The mask is then resized and rotated, placing it on the face:

In this figure, the face mask is placed on the person's face in the original frame. It is difficult to tell at a glance that the COVID-19 mask has been applied with computer vision by way of OpenCV and dlib face landmarks.

We can then repeat this process for all of our input images, thereby creating our artificial face mask dataset:

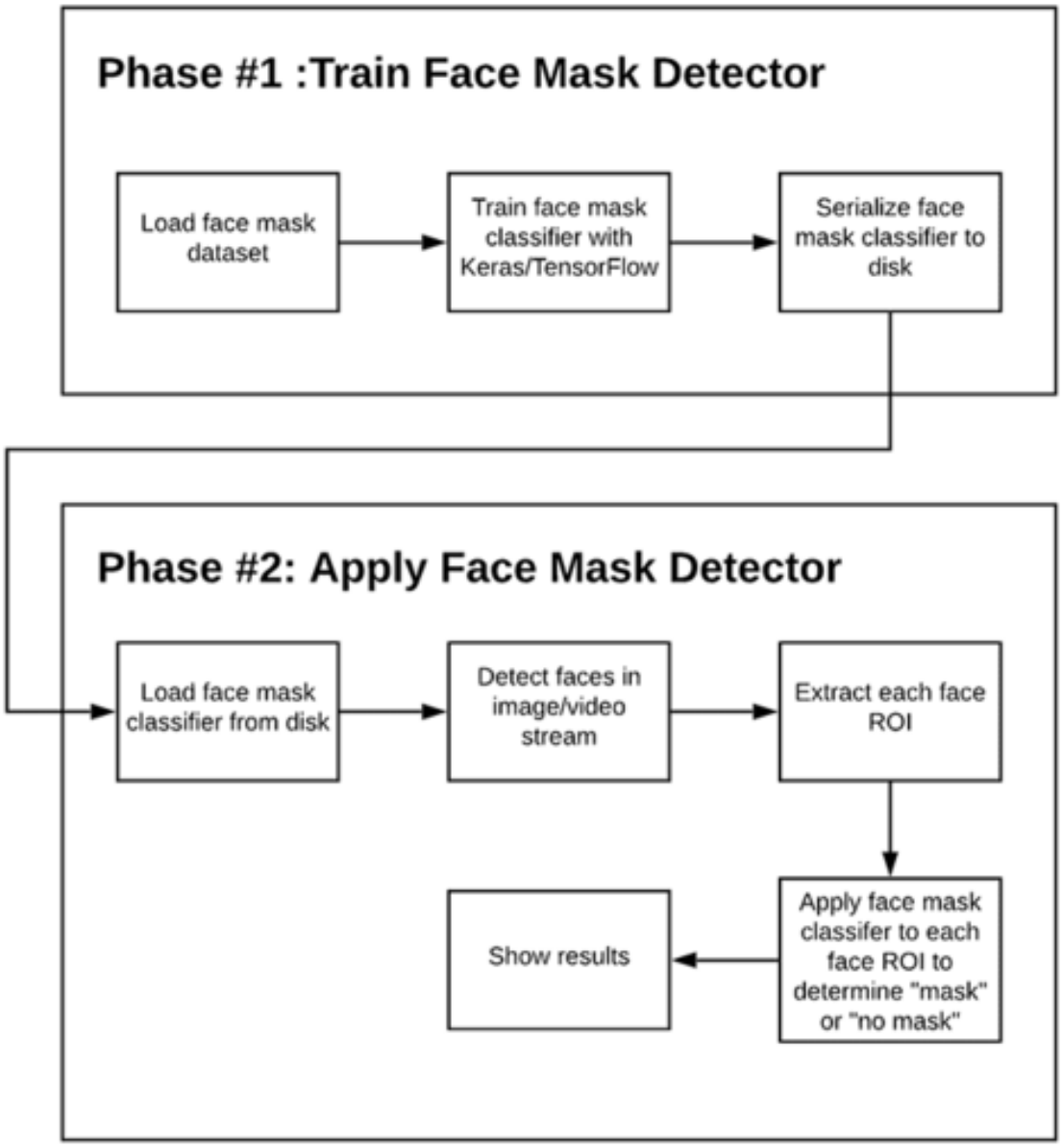
An artificial set of COVID-19 face mask images is shown. This set will be part of our “with mask” / “without mask” dataset for COVID-19 face mask detection with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

However, there is a caveat you should be aware of when using this method to artificially create a dataset!

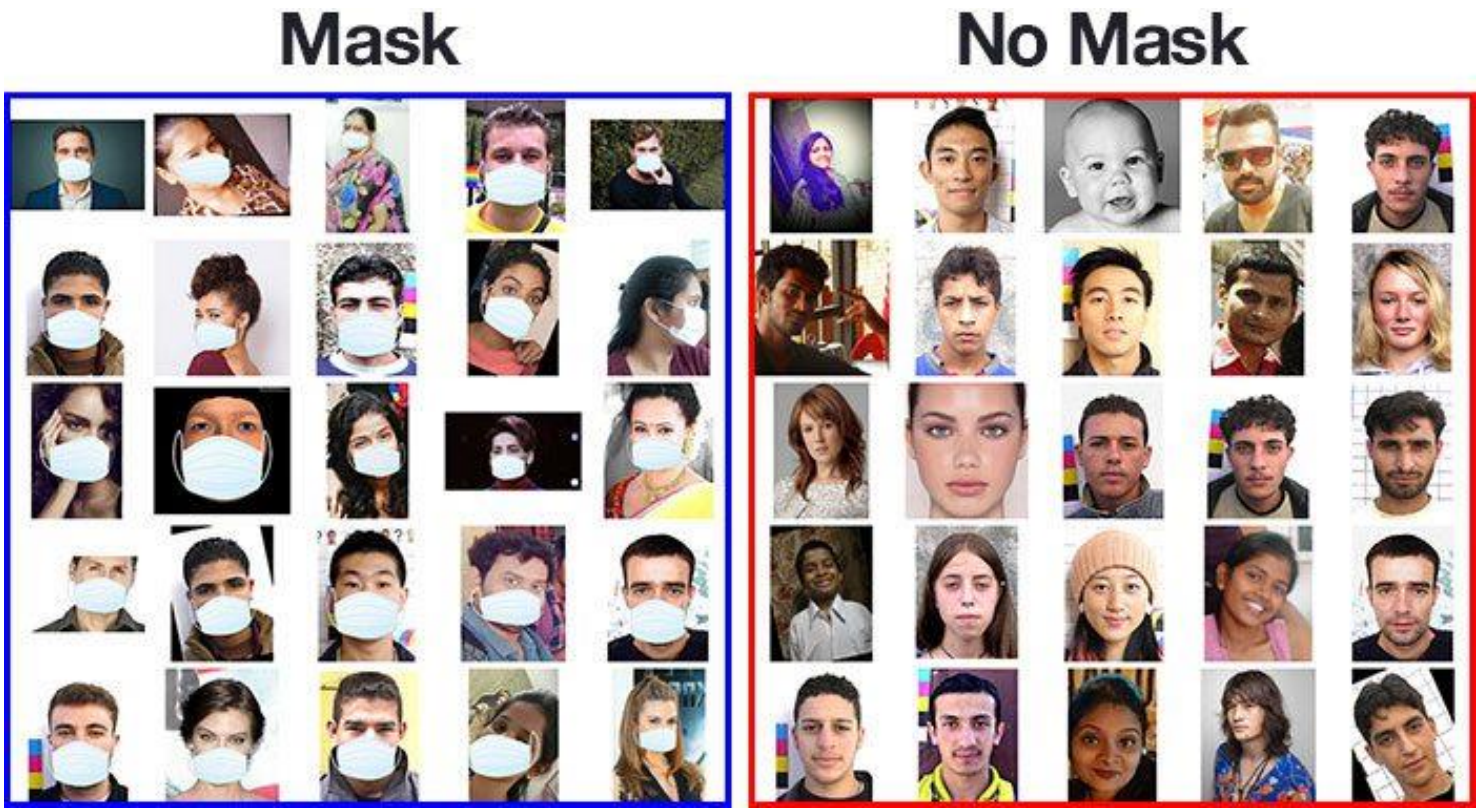
If you use a set of images to create an artificial dataset of people wearing masks, you cannot “re-use” the images without masks in your training set — you still need to gather non-face mask images that were not used in the artificial generation process!

If you include the original images used to generate face mask samples as non-face mask samples, your model will become heavily biased and fail to generalize well. Avoid that at all costs by taking the time to gather new examples of faces without masks.

Two-phase COVID-19 face mask detector

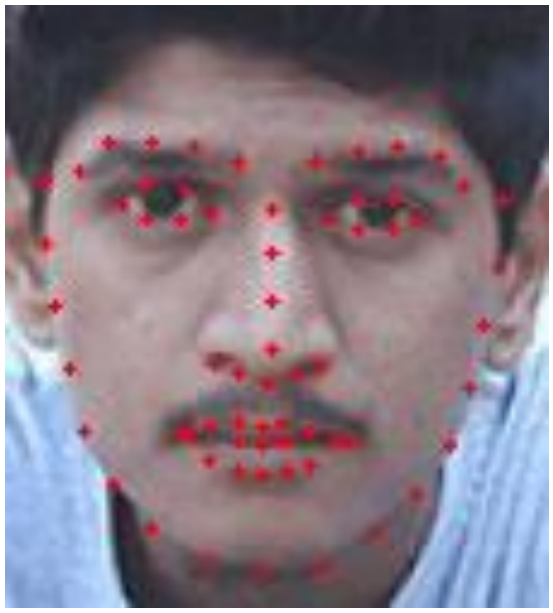


Our COVID-19 face mask detection dataset



To build a COVID-19/Coronavirus pandemic face mask dataset, we'll first start with a photograph of someone not wearing a face.

The next step is to apply face detection. Here we've used a deep learning method to perform face detection with OpenCV.



Then, we detect facial landmarks using dlib so that we know where to place a mask on the face.

Next, we need an image of a mask (with a transparent background) such as the one below:

An example of a COVID-19/Coronavirus face mask/shield. This face mask will be overlaid on the original face ROI automatically since we know the face landmark locations.

In this figure, the face mask is placed on the person's face in the original frame. It is difficult to tell at a glance that the COVID-19 mask has been applied with computer vision by way of OpenCV and dlib face landmarks.

We can then repeat this process for all of our input images, thereby creating our artificial face mask dataset:

However, there is a caveat you should be aware of when using this method to artificially create a dataset!

If you use a set of images to create an artificial dataset of people wearing masks, you cannot “re-use” the images without masks in your training set — you still need to gather non-face mask images that were not used in the artificial generation process!

If you include the original images used to generate face mask samples as non-face mask samples, your model will become heavily biased and fail to generalize well. Avoid that at all costs by taking the time to gather new examples of faces without masks.
Covering how to use facial landmarks to apply a mask to a face is outside the scope of this tutorial, but if you want to learn more about it, I would suggest:

Referring to repository
Reading this tutorial on the Py Image Search blog where I discuss how to use facial landmarks to automatically apply sunglasses to a face
The same principle from my sunglasses post applies to building an artificial face mask dataset — use the facial landmarks to infer the facial structures, rotate and resize the mask, and then apply it to the image.

Implementing our COVID-19 face mask detector training script with Keras and TensorFlow

Now that we've reviewed our face mask dataset, let's learn how we can use Keras and TensorFlow to train a classifier to automatically detect whether a person is wearing a mask or not.

To accomplish this task, we'll be fine-tuning the Mobile Net V2 architecture, a highly efficient architecture that can be applied to embedded devices with limited computational capacity (ex., Raspberry Pi, Google Coral, NVIDIA Jetson Nano, etc.).

Note: If your interest is embedded computer vision, be sure to check out my Raspberry Pi for Computer Vision book which covers working with computationally limited devices for computer vision and deep learning.

Deploying our face mask detector to embedded devices could reduce the cost of manufacturing such face mask detection systems, hence why we choose to use this architecture.

The imports for our training script may look intimidating to you either because there are so many or you are new to deep learning. If you are new, I would recommend reading both my Keras tutorial and fine-tuning tutorial before moving forward.

Our set of tensorflow, keras imports allow for:

Data augmentation
Loading the MobilNetV2 classifier (we will fine-tune this model with pre-trained ImageNet weights)

- Building a new fully-connected (FC) head
- Pre-processing
- Loading image data

We'll use scikit-learn (sklearn) for binarizing class labels, segmenting our dataset, and printing a classification report.

My imutils paths implementation will help us to find and list images in our dataset. And we'll use matplotlib to plot our training curves.

To install the necessary software so that these imports are available to you, be sure to follow either one of my Tensorflow 2.0+ installation guide

Our command line arguments include:

--dataset: The path to the input dataset of faces and and faces with masks
--plot: The path to your output training history plot, which will be generated using matplotlib
--model: The path to the resulting serialized face mask classification model
I like to define my deep learning hyperparameters in one place:
Here, I've specified hyperparameter constants including my initial learning rate, number of training epochs, and batch size. Later, we will be applying a learning rate decay schedule, which is why we've named the learning rate variable INIT_LR.

•Grabbing all of the image Paths in the dataset Initializing data and labels lists
•Looping over the image Paths and loading + pre-processing images. Pre-processing steps include resizing to *224×224* pixels, conversion to array format, and scaling the pixel intensities in the input image to the range *[-1, 1]* (via the pre process input convenience function)
•Appending the pre-processed image and associated label
• To the data and labels lists, respectively
•Ensuring our training data is in NumPy array format
•The above lines of code assume that your entire dataset is small enough to fit into memory. If your dataset is larger than the memory you have available, I suggest using HDF5, a strategy I cover in Deep Learning for Computer Vision with Python
Our data preparation work isn't done yet. Next, we'll encode our labels, partition our dataset, and prepare for data augmentation:
As you can see, each element of our labels array consists of an array in which only one index is "hot".

Using scikit-learn's convenience method, Lines 73 and 74 segment our data into 80% training and the remaining 20% for testing.

During training, we'll be applying on-the-fly mutations to our images in an effort to improve generalization. This is known as data augmentation, where the random rotation, zoom, shear, shift, and flip parameters are established on Lines 77-84. We'll use the aug object at training time.

But first, we need to prepare MobileNetV2 for fine-tuning:
Fine-tuning setup is a three-step process:

Load Mobile Net with pre-trained ImageNet weights, leaving off head of network
Construct a new FC head, and append it to the base in place of the old head. Freeze the base layers of the network. The weights of these base layers will not be updated during the process of backpropagation, whereas the head layer weights will be tuned.
Fine-tuning is a strategy I nearly always recommend to establish a baseline model while saving considerable time. To learn more about the theory, purpose, and strategy, please refer to my fine-tuning blog posts and Deep Learning for Computer Vision with Python
With our data prepared and model architecture in place for fine-tuning, we're now ready to compile and train our face mask detector network:
Compile our model with the Adam optimizer, a learning rate decay schedule, and binary cross-entropy. If you're building from this training script with > 2 classes, be sure to use categorical cross-entropy.

Face mask training is launched. Notice how our data augmentation object (aug) will be providing batches of mutated image data.

Once training is complete, we'll evaluate the resulting model on the test set:

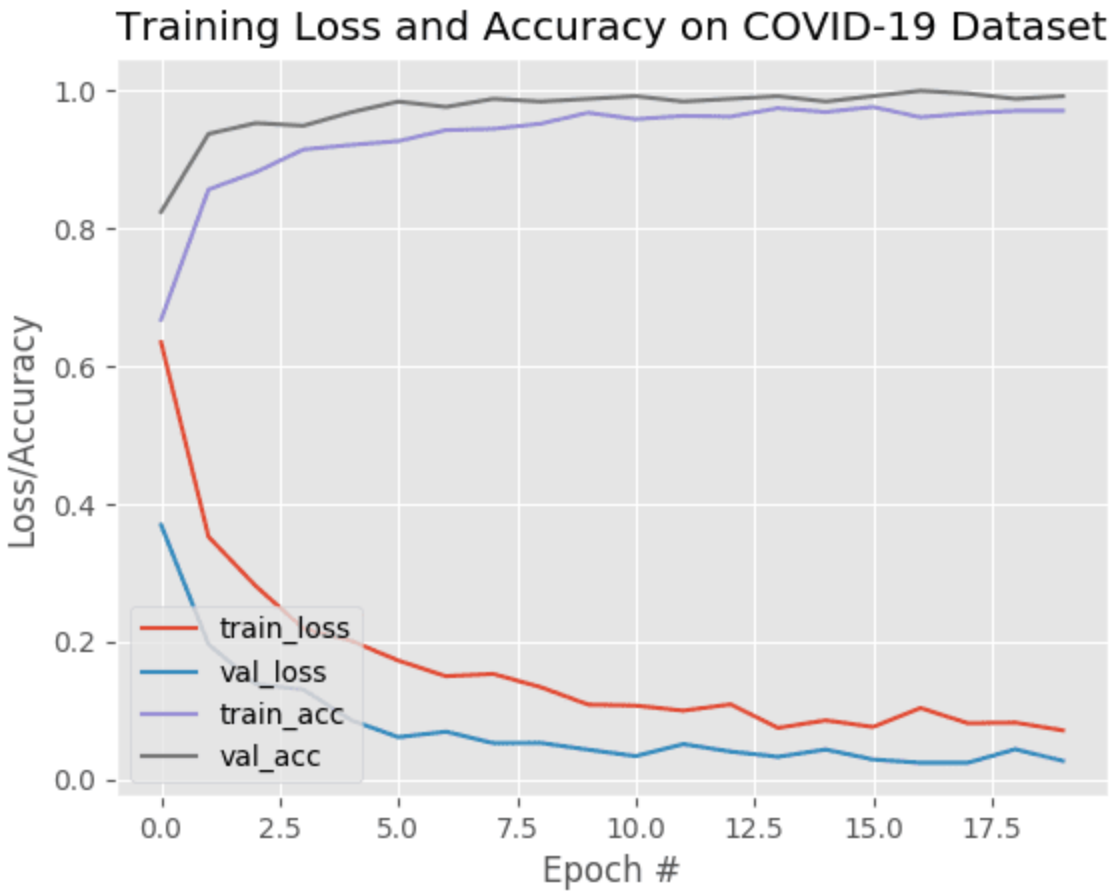
Here, make predictions on the test set, grabbing the highest probability class label indices. Then, we print a classification report in the terminal for inspection.
serializes our face mask classification model to disk.
Our last step is to plot our accuracy and loss curves:

Training the COVID-19 face mask detector with Keras /TensorFlow

We are now ready to train our face mask detector using Keras , TensorFlow, and Deep Learning.

Make sure you have used the “Downloads” section of this tutorial to download the source code and face mask dataset.

From there, open up a terminal, and execute the following command:



As you can see, we are obtaining ~99% accuracy on our test set.

Looking at Figure 10, we can see there are little signs of overfitting, with the validation loss lower than the training loss (a phenomenon I discuss in this blog post).

Given these results, we are hopeful that our model will generalize well to images outside our training and testing set.

Implementing our COVID-19 face mask detector for images with OpenCV

Now that our face mask detector is trained, let's learn how we can:
1.Load an input image from disk
2.Detect faces in the image
3.Apply our face mask detector to classify the face as either with mask
4. or without mask

Open up the
detect_mask_image.py
file in your directory structure, and let's get started:
Our driver script requires three TensorFlow, Keras imports to load our Mask Net model and pre-process the input image.

OpenCV is required for display and image manipulations.

The next step is to parse command line arguments:
With our deep learning models now in memory, our next step is to load and pre-process an input image:
Upon loading our --image from disk ,we make a copy and grab frame dimensions for future scaling and display purposes (Lines 38 and 39).

Pre-processing is handled by OpenCV's blob From Image function .As shown in the parameters, we resize to 300×300 pixels and perform mean subtraction.
Then perform face detection to localize where in the image all faces are.

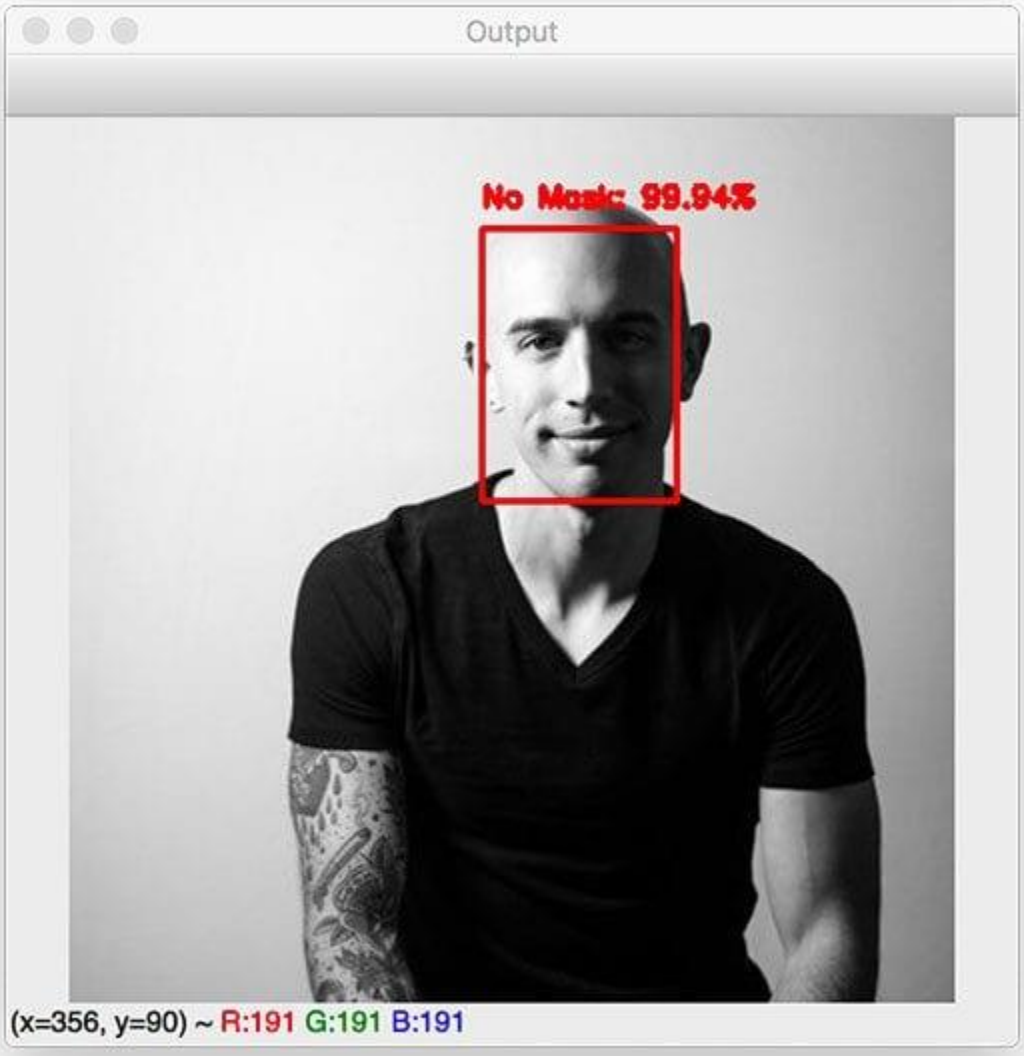
Once we know where each face is predicted to be, we'll ensure they meet the --confidence threshold before we extract the face ROIs:
Here, we loop over our detections and extract the confidence to measure against the --confidence threshold
We then compute bounding box value for a particular face and ensure that the box falls within the boundaries of the image
Next, we'll run the face ROI through our Mask Net model:
•Extract the face
• ROI via NumPy slicing
•Pre-process the ROI the same way we did during training.
•Perform mask detection to predict with mask
• or without mask
From here, we will annotate and display the result!
First, we determine the class label based on probabilities returned by the mask detector model and assign an associated color for the annotation (Line 85). The color will be "green" for with mask and "red" for without mask.

We then draw the label text (including class and probability), as well as a bounding box rectangle for the face, using OpenCV drawing functions
Once all detections have been processed.

COVID-19 face mask detection in images with OpenCV

:

Is this man wearing a COVID-19/Coronavirus face mask in public? Yes, he is and our computer vision and deep learning method using Python, OpenCV, and TensorFlow/Keras has made it possible to detect the presence of the mask automatically. (Image Source)



Uh oh. I’m not wearing a COVID-19 face mask in this picture. Using Python, OpenCV, and TensorFlow/Keras , our system has correctly detected “No Mask” for my face.

What is going on in this result? Why is the lady in the foreground not detected as wearing a COVID-19 face mask? Has our COVID-19 face mask detector built with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras

What happened here?

Why is it that we were able to detect the faces of the two gentlemen in the background and correctly classify mask/no mask for them, but we could not detect the woman in the foreground?
I discuss the reason for this issue in the “Suggestions for further improvement” section later in this tutorial, but the gist is that we’re too reliant on our two-stage process.
Keep in mind that in order to classify whether or not a person is wearing in mask, we first need to perform face detection — if a face is not found (which is what happened in this image), then the mask detector cannot be applied!
The reason we cannot detect the face in the foreground is because:
It’s too obscured by the mask
The dataset used to train the face detector did not contain example images of people wearing face masks
Therefore, if a large portion of the face is occluded, our face detector will likely fail to detect the face.
Again, I discuss this problem in more detail, including how to improve the accuracy of our mask detector, in the “Suggestions for further improvement” section of this tutorial.

Implementing our COVID-19 face mask detector in real-time video streams with OpenCV

At this point, we know we can apply face mask detection to static images — but what about real-time video streams?
Is our COVID-19 face mask detector capable of running in real-time?
Let’s find out.
Open up the detect_mask_video.py file in your directory structure, and insert the following code:
The algorithm for this script is the same, but it is pieced together in such a way to allow for processing every frame of your webcam stream.
Thus, the only difference when it comes to imports is that we need a Video Stream class and time. Both of these will help us to work with the stream. We’ll also take advantage of imutils for its aspect-aware resizing method.
Our face detection/mask prediction logic for this script is in the detect and predict mask function:

From here, we’ll loop over the face detections:
Inside the loop, we filter out weak detections ,and extract bounding boxes while ensuring bounding box coordinates do not fall outside the bounds of the image .
Next, we’ll add face ROIs to two of our corresponding lists:

Convert faces into a 32-bit floating point NumPy array. Additionally, from the previous block has been removed (formerly, it added an unnecessary batch dimension). The combination of these two changes now fixes a bug that was preventing multiple preds to be returned from inference. With the fix, multiple faces in a single image are properly recognized as having a mask or not having a mask.

Secondly, we are performing inference on our entire batch of faces in the frame so that our pipeline is faster. It wouldn't make sense to write another loop to make predictions on each face individually due to the overhead (especially if you are using a GPU that requires a lot of overhead communication on your system bus). It is more efficient to perform predictions in batch.

our face bounding box locations and corresponding mask/not mask predictions to the caller.

Here we have initialized our:

Face detector
COVID-19 face mask detector
Webcam video stream
Let's proceed to loop over frames in the stream:
Inside our loop over the prediction results
Unpack a face bounding box and mask/not mask prediction
Determine the label and color
Annotate the label and face bounding box
Finally, we display the results and perform cleanup:
After the frame is displayed, we capture key presses. If the user presses q (quit), we break out of the loop and perform housekeeping.
Great job implementing your real-time face mask detector with Python, OpenCV, and deep learning with TensorFlow/Keras.

Detecting COVID-19 face masks with OpenCV in real-time

To see our real-time COVID-19 face mask detector in action, make sure you use the "Downloads" section of this tutorial to download the source code and pre-trained face mask detector model.
You can then launch the mask detector in real-time video streams using the following command:

For improvement

As you can see from the results sections above, our face mask detector is working quite well despite:

Having limited training data

The with mask class being artificially generated (see the “How was our face mask dataset created?”

To improve our face mask detection model further, you should gather actual images (rather than artificially generated images) of people wearing masks.

While our artificial dataset worked well in this case, there’s no substitute for the real thing.

Secondly, you should also gather images of faces that may “confuse” our classifier into thinking the person is wearing a mask when in fact they are not — potential examples include shirts wrapped around faces, bandana over the mouth, etc.

All of these are examples of something that could be confused as a face mask by our face mask detector.

Finally, you should consider training a dedicated two-class object detector rather than a simple image classifier.

Our current method of detecting whether a person is wearing a mask or not is a two-step process:

Step 1: Perform face detection

Step 2: Apply our face mask detector to each face

The problem with this approach is that a face mask, by definition, obscures part of the face. If enough of the face is obscured, the face cannot be detected, and therefore, the face mask detector will not be applied.

To circumvent that issue, you should train a two-class object detector that consists of a with mask class and without mask class.

Combining an object detector with a dedicated with mask class will allow improvement of the model in two respects.

First, the object detector will be able to naturally detect people wearing masks that otherwise would have been impossible for the face detector to detect due to too much of the face being obscured.

Secondly, this approach reduces our computer vision pipeline to a single step — rather than applying face detection and then our face mask detector model, all we need to do is apply the object detector to give us bounding boxes for people both with mask and without mask in a single forward pass of the network.

Not only is such a method more computationally efficient, it’s also more “elegant” and end-to-end.

Summary

We, create a COVID-19 face mask detector using OpenCV, Keras/TensorFlow, and Deep Learning.

To create our face mask detector, we trained a two-class model of people wearing masks and people not wearing masks.

We fine-tuned MobileNetV2 on our mask/no mask dataset and obtained a classifier that is ~99% accurate.

We then took this face mask classifier and applied it to both images and real-time video streams by:

Detecting faces in images/video
Extracting each individual face
Applying our face mask classifier
Our face mask detector is accurate, and since we used the MobileNetV2 architecture, it’s also computationally efficient, making it easier to deploy the model to embedded systems (Raspberry Pi, Google Coral, Jetosn, Nano, etc.)

RESULT AND CONCLUSION

By preserving a reasonable proportion of different classes, the dataset is partitioned into training and testing set. The dataset comprises of 1315 samples in total where 80% is used in training phase and 20% is used in testing phase. The developed architecture is trained for 10 epochs since further training results cause overfitting on the training data. Overfitting generally occurs when a model learns the unwanted patterns of the training samples. Hence, training accuracy increases but test accuracy decreases.

The graphical view of accuracy and loss respectively. The trained model showed 95% accuracy.

The developed system can detect the live video streams but does not keep a record. Unlike the CCTV camera footage the admin can not rewind, play or pause it. As whenever a strict system is imposed people always try to break it. Hence when a person is detected with no mask, the head of the organization can be notified via mail that so and so person entered without mask. The proposed system can be integrated with databases of respective organizations to keep a record of the person who entered without mask. With more complex functions a screenshot of the person's face can also be attached to keep it as a proof. As the technology is booming with emerging trends therefore the novel face mask detector which

can possibly contribute to public healthcare. The model is trained on an authentic dataset. We used OpenCV, tensor flow, keras and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video. The accuracy of the model is achieved and, the optimization of the model is a continuous process and we are building an accurate solution by tuning the hyper parameters. This specific model could be used as a use

case for edge analytics. By the developing this system, we can detect if the person is wearing a face mask and allow their entry would be of great help to the society.

As time passes worldwide, the COVID-19 pandemic poses greater challenges for humans. Over the past two years, leaders in biometric systems, such as FLIR Systems , among others, have launched proposals for updated access control and temperature detection. They also highlight the products previously launched by the companies Sense Time, Telpo , and Herta. All of this is focused mainly on temperature control in airports and high-traffic places, with which the development of easy recognition systems has been relegated. As previously mentioned, the regular use of face masks has directly affected facial recognition systems. This has altered the usual development of activities and delayed innovation processes. For example, Apple's Face ID is designed to identify users based on the mouth, nose, and eyes, which must be fully visible. In the same way, other important companies worldwide, such as Go from Amazon and Walmart's "Store of the Future", are affected, as they use this technology to interact with their customers. There are independent investigations as in [55], where Google resources have been used to develop systems for detecting the use of masks, but with still low yields

As can be seen in the bibliography presented, during the advancement of this pandemic, CNNs have been used as a priority for the diagnosis of the disease. Various standard training methods have been used, such as Efficient Net and y Res Net, as well as other proprietary ones such as n CO Vnet. Similarly, for face mask use detection systems, several methods have been tested, such as RCNN, VGG, MobileNetV2, SVM, InceptionV3, and y S RC Net. All of them have been analyzed ,determined that using Mobile-NetV2 would have efficient results, with a low consumption of computational resources. In addition, when comparing the efficiency of the various systems, it is found that this proposal is competitive, as it has 99.65%, only surpassed. From the point of view of facial recognition, a similar proposal has not been found, which clearly shows the added value offered. By having a precision greater than 99%, the starting hypothesis is corroborated, establishing variability in this type of system and also adding robustness, presents a comparison of this proposal with the other works.

Other proposals in face detection have also reviewed, but with the ocular recognition variant, as seen. However, it should be clarified that focusing on the human eye represents a different approach than the one analyzed in this document. In addition, this would require the acquisition of other hardware and software components that would increase the requirements presented. This document seeks to capture the upper part of the face, i.e., eyes, eyebrows, forehead, and hair. Thus, it has been shown that this approach can be successful. On the subject of convolutional neural networks, the most important thing is to have a good database (thousands of observations). This is not a rule, but the larger the database, the better the resulting model, as, in a certain way, more characteristics are extracted and a model that represents them can be generalized. It does not matter if there are many or few classes (people); for each class (person), a considerable amount of observations is needed (images, database, etc.). This type of architecture is ideal when it comes to artificial vision, as it is demonstrated in the literature that convolutional neural networks are leading the advances in artificial vision, which is why this architecture is chosen, in order to have the Sustainability 2021, 13, 6900 16 of 19 best possible results, as the topic of facial recognition is an area where the system should fail as little as possible. The small number of people is not a problem, because the proposal is aimed at access control, which is a structured environment, in order for a restricted area to be limited to 10 or fewer people with entry authorization. In addition, the literature reviewed presents cases with the same performance for differentiating people with and without a face mask. The architecture has a heavy component in Transfer Learning (MobileNet) and is used to train the first classifier (second stage); this model is the one that classifies whether a face is using a mask or not.

Other proposals in face detection have also reviewed, but with the ocular recognition variant, as seen in [52,53]. However, it should be clarified that focusing on the human eye represents a different approach than the one analyzed in this document. In addition, this would require the acquisition of other hardware and software components that would increase the requirements presented. This document seeks to capture the upper part of the face, i.e., eyes, eyebrows, forehead, and hair. Thus, it has been shown that this approach can be successful. On the subject of convolutional neural networks, the most important thing is to have a good database (thousands of observations). This is not a rule, but the larger the database, the better the resulting model, as, in a certain way, more characteristics are extracted and a model that represents them can be generalized. It does not matter if there are many or few classes (people); for each class (person), a considerable amount of observations is needed (images, database, etc.). This type of architecture is ideal when it comes to artificial vision, as it is demonstrated in the literature that convolutional neural networks are leading the advances in artificial vision, which is why this architecture is chosen, in order to have the Sustainability 2021, 13, 6900 16 of 19 best possible results, as the topic of facial recognition is an area where the system should fail as little as possible. The small number of people is not a problem, because the proposal is aimed at access control, which is a structured environment, in order for a restricted area to be limited to 10 or fewer people with entry authorization. In addition, the literature reviewed presents cases with the same performance for differentiating people with and without a face mask. The architecture has a heavy component in Transfer Learning (MobileNet) and is used to train the first classifier (second stage); this model is the one that classifies whether a face is using a mask or not. This has nothing to do with the classes for facial recognition, as facial recognition uses a fairly simple and lightweight architecture such as a feedforward multilayer perceptron. Although the tools used in this work use existing models, the authors did not find a proposal that uses them in combination with a similar application. Another contribution is the use of cloud computing for this architecture, offering a low-cost system.

CONCLUSION AND FUTURE WORKS

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 Sustainability 2021, 13, 6900 17 of 19 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.

REFRENCES

Serign ModouBah, FangMing ,"An improved face recognition algorithm and its application in attendance management system", volume 5, March 2020, 100014, https://doi.org/10.1016/j.array.2019.100014

- Image-based Face Detection and Recognition: "State of the Art"
- https://www.youtube.com/watch?v=gbfj4RH6RlM&feature=youtu.be
- https://www.youtube.com/watch?v=a3HBsN5RvQs
- https://www.youtube.com/watch?v=hT28HIL_q18