FACE MASK RECOGNITION

Members:

Himadri Pandya (20ME10047)

Samarth Singh (20MF10031)

Kondapalli Vivek Abhiram (20IM30008)

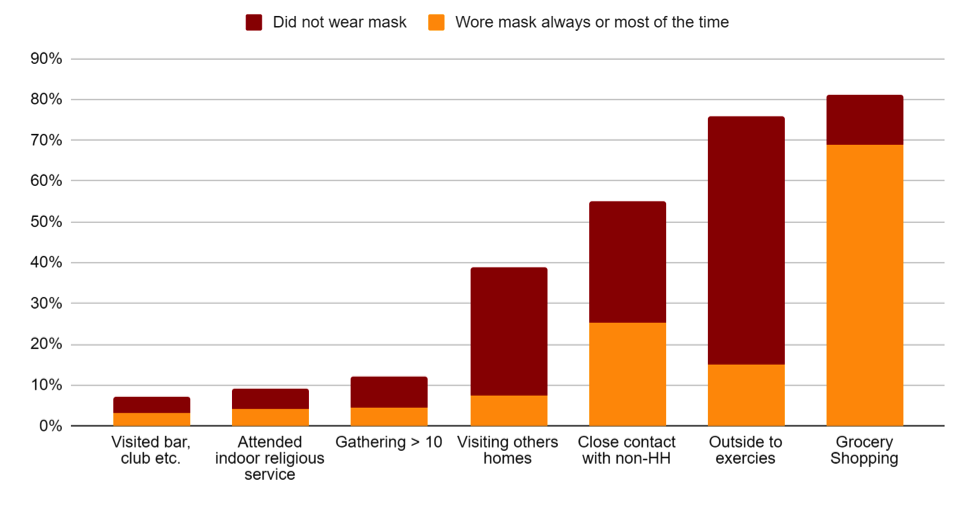
Navya Jattan (20MA20037)

Motivation

In the current situation, masks have become a norm and there is a pressing need to ensure that the majority of the population wears masks. We thought automating the process of mask recognition could be a huge leap in this regard. This project aims to present the following vision:

* Using Face Mask Detection System, Hospitals can monitor if their staff is wearing masks during their shift or not. If any health worker is found without a mask, it will be visible in the feed.
* The Face Mask Detection System can also be used in office premises to detect if employees are maintaining necessary precautions at work. It efficiently monitors employees with/without masks.
* Other implementable areas are railway stations, schools and public places.

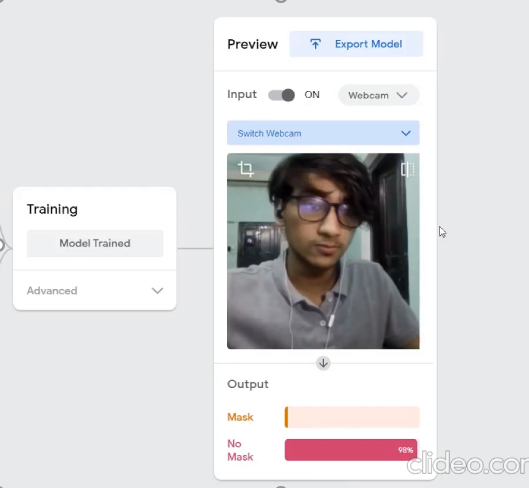
With this aim in mind, the project implementation was decided to be done in Python because its extensive libraries provided a clear outlet to the aim we had in mind.

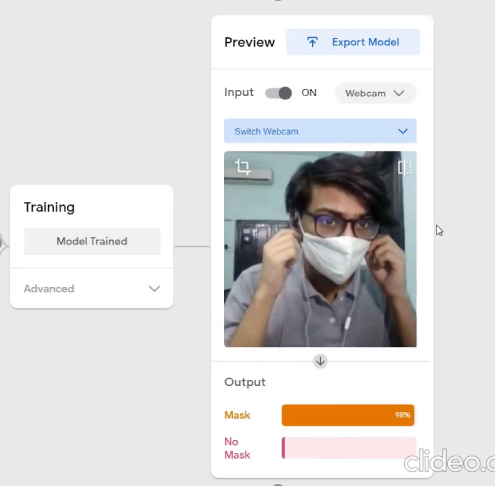


Mask-wearing Statistics

Inspiration

Once we had finalised our topic, we decided for inspirations we could build upon. During our search, we came across a teachable model that allowed us to build classes and obtain a result online. This is what we found:





We decided to keep this result in mind and set our aim to accomplish a similar result.

Libraries used:

* OpenCV
* Caffe-based face detection
* Keras
* TensorFlow
* MobileNetV2

Other resources:

* Kaggle
* GitHub
* Google Colab
* Visual Studio Code

Details of Work Done

* Face Detection using OpenCV

Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces. It is basically a machine learning object detection algorithm which is used to identify objects in an image or video. In OpenCV, we have several trained Haar Cascade models which are saved as XML files. Instead of creating and training the model from scratch, we use this file. We are going to use “haarcascade\_frontalface\_alt2.xml” file in this project.

path to the XML file of goes as an argument to CascadeClassifier() method of OpenCV.

The CascadeClassifier() object we created has a method detectMultiScale(), which receives a frame(image) as an argument and runs the classifier cascade over the image. The term MultiScale indicates that the algorithm looks at subregions of the image in multiple scales, to detect faces of varying sizes.

* Face Recognition in Images

Creating this classifier, we need data in the form of Images containing images with mask on and off Since these images are very less in number, we cannot train a neural network from scratch. Instead, we finetune a pre-trained network called MobileNetV2 which is trained on the Imagenet dataset.

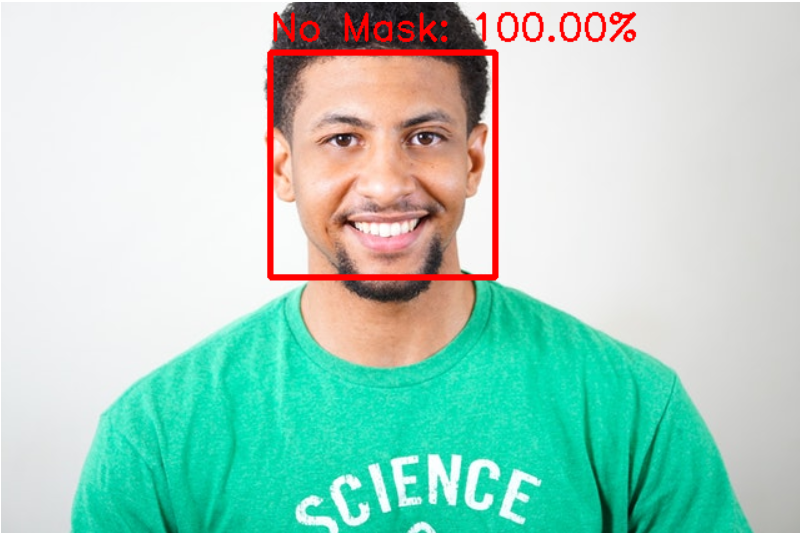
dataset is contained in two folders viz- with\_masks and without\_masks. So, we can easily get the labels by extracting the folder name from the path. Also, the images were preprocessed image and resize it to 224x 224 dimensions.

loaded the pre-trained model and customize it according to our problem. So, we just remove the top layers of this pre-trained model and add few layers of our own. As you can see the last layer has two nodes as we have only two outputs. This is called transfer learning.

Now we need to convert the labels into one-hot encoding. After that, we split the data into training and testing sets to evaluate them. Also, the next step is data augmentation which significantly increases the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, rotation, shearing and horizontal flipping are commonly used to train large neural networks.

* Results of the Image Model

The model gave staggering results with ~98% accuracy ans successfully recognized faces and classified them as masked or not masked. Here are the snaps of a few test cases:





* Scopes of improvisation in the Model

1. Detection of a single face: When we fed images with more than one face, we realised that the model was not able to recognize more faces and only gave results for the leftmost face.
2. Need of a real-time model: In a real-life scenario a more immediate approach was needed. This prompted us to look for a better model. We decided to make a model that would capture real-time video feeds from cameras and readily gave the desired output with minimal lag.
3. Data-base: We had used a relatively small data base to train this model (with approximately 1500 images for each case). To make the model more robust and fast we decided to use a larger database for the video feed model.

* Model Applied in Video Feed

This model builds upon our previous work and gives a more real-time approach.

In order to optimize the coding, we called the image recognition and detection program from our previous code and called it as a method to analyze each frame that was being stored in the array of frames. It combined all the image frames captured from the camera and stored in in an array; it integrated the frame capture and displayed whether the face recognized in the feed is wearing a mask or not. We trained this model using TensorFlow Library of Python and ensured optimization during the process. The program finally displayed this output with a frame around the recognized face tagged “mask” or “no mask”.

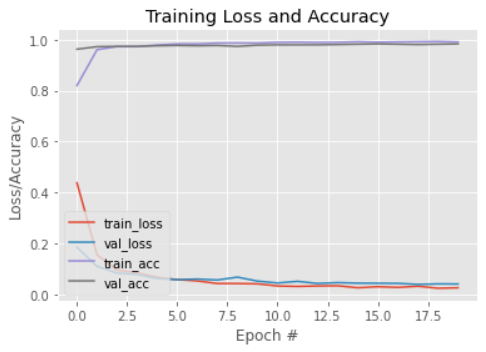
* Cases considered in the final model

1. Real-time Identification: The program captures real-time data from the video feeds of a camera and gives the result with minimal lag.
2. Multiple-face Detection: The model can recognize more than one faces in the feed. This was implemented by varying the database and ensuring that the model identified all possible landmarks instead of stopping after finding one.
3. Data-base: In order to ensure that this model would be more robust and efficient we trained the model using a relatively large database (with approximately 3000 images for each case).

Results

* Training

While training our model we recorded the accuracy of every epoch we trained and the losses that were incurred. The results were really promising with minimal loss and maximum retention. This is the plot that we obtained:

****

* Efficiency

The final model presented a commendable accuracy of ~98.7% with a feed to output lag of 1.8 seconds (worst case) and 0.27 seconds (best case).

This was a great number as the model was able to recognize pretty much every field and was also capable of distinguishing between placing palms on the face and wearing a mask.

Contribution of Members

Himadri Pandya: Coding of video feed and presentation.

Kondapalli Vivek Abhiram: Video Editing and Resource collection.

Samarth Singh: Coding of image feed and presentation.

Navya Jattan: Research and documentation.

Link to the YouTube Video

<https://youtu.be/sNAoUhWixjs>

References

1. <https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/>
2. <https://www.youtube.com/watch?v=atw41gV5LJo&ab_channel=MisbahMohammed>
3. <https://www.hackster.io/karem_benchikha/facemask-detection-88fa1a>
4. <https://www.hackster.io/Atia/nose-mask-checker-bd14ed>
5. <https://tryolabs.com/blog/2020/07/09/face-mask-detection-in-street-camera-video-streams-using-ai-behind-the-curtain/>
6. <https://docs.opencv.org/4.5.2/d6/d00/tutorial_py_root.html>
7. <https://www.tensorflow.org/tutorials>
8. <https://numpy.org/doc/stable/user/quickstart.html>
9. <https://www.kaggle.com/andrewmvd/face-mask-detection>