INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 4 - V7I4-1684)
Available online at: https://www.ijariit.com

Web-based system for face mask detection and face recognition

Pranjal Rane
pranjal020700@gmail.com
Vishwakarma Institute of Technology, Pune, Maharashtra

Ritvik Patil
pritvik0@gmail.com
Vishwakarma Institute of Technology, Pune, Maharashtra

Ojas Natu
natuojas@gmail.com
Vishwakarma Institute of Technology, Pune, Maharashtra

Prahlad Pore
prahlad1501@gmail.com
Vishwakarma Institute of Technology, Pune, Maharashtra

Pranay Shridhar
pranay.19990427@gmail.com
Vishwakarma Institute of Technology, Pune, Maharashtra

ABSTRACT

Even though people are now getting vaccinated, it might still take time for things to come back to normal. Until then, wearing masks and social distancing is the most-effective preventive measure against the spread of COVID-19. This leads to the necessity of a system that will detect face masks in real-time video streams. Our solution detects whether a person is wearing a mask or not. If he/she is not wearing a mask then his/her photo would be captured and given as input to a face recognition system which will identify the person. The solution further provides the person's information to the administrator and allows him to take any further action if needed. The above solution is implemented using neural networks to detect masks and the website is created using Google Flutter framework.[9] The 'Web Based System for Face Mask Detection and Face Recognition' can be used to ensure the safety of people by implementing it in large crowd gathering places like college campuses and offices.

Keywords: *Mask Detection, Real-time, Face Recognition, ResNet, Web App, Firebase*

1. INTRODUCTION

In today's time, a mask is considered to be an important measure to suppress the transmission of various diseases and save lives. Masks are used as part of a comprehensive approach which includes physical distancing, avoiding crowded places, good ventilation, cleaning hands, covering sneezes and coughs can significantly lower the rate of spread of any infectious disease. Even though people are now getting vaccinated, it might still be too early to return to normal. Until then, wearing masks is the only solution to stop the spread of COVID-19.

The 'Web Based System for Face Mask Detection and Face Recognition' is used to ensure the safety of people by implementing it in crowd gathering places like college campuses and offices. This system detects face masks in real-time video streams. If a person is not wearing a mask then his/her photograph is captured and given as an input to the face recognition system which identifies the person. The administrator is informed about this and can later take necessary actions against the defaulter, like reprimanding by sending an email or issuing a fine using the web based system. The administrator can also manage the database of the people using the web based system. The web based system also gives a detailed overview about the demography (pertaining to mask wearing) of the campus by giving comprehensive information using visual methods like bar graphs and pie charts.

2. LITERATURE REVIEW

There are several toolkits available on the internet to develop solutions based on machine learning algorithms. We have chosen the Dlib toolkit for the above purpose. Dlib is a C++ library which was created to deal with complex softwares to solve real-life problems. One notable feature of the face recognition model within Dlib is its accuracy of 99.38% [1] against the standard Labeled Faces in the Wild Benchmark. This indicates the accuracy of the models in correctly predicting, almost every time, whether any two given images are of the same person or not. It consists of a ResNet network with 29 convolution layers in it, making it a deep network.[2]

In regular deep learning networks, conventional layers are followed by fully connected layers which are used for classification tasks. These layers are called plain networks.

When a network tends to get deeper, i.e. the number of layers are increased, there is a tendency of arising of the problem of vanishing or exploding gradient. This is mainly due to the chain rule which results in multiplication of 'n' number of derivatives as we progress forward through a network of n hidden layers. If these n derivatives are large enough then their product, the

gradient, increases exponentially and eventually leads to gradient explosion. Similarly, if these n derivatives are small enough, the gradient decreases exponentially and eventually leads to vanishing gradient. This vanishing/exploding gradient problem could be addressed in three ways - Reducing the amount of layers, Gradient Clipping, Weight Initialization.

Although reducing the amount of layers would definitely help in solving the above problem, it will also reduce the ability of the model to solve more complex problems since less number of layers would result in relatively less complex mappings. In gradient clipping, we limit the gradient value while the model is being trained simultaneously. This prevents the gradient from either vanishing or exploding. In the last approach, the weights could be carefully initialized in the beginning instead of random initialization to limit the vanishing/exploding gradient problem. This approach is known for only partially solving the gradient problem.

Among the various convolutional neural networks, residual networks prove to be the most suitable one for achieving the objective of this project. [4][5] This is mainly due to the characteristics of residual networks -

- Residual networks could be relatively easy to optimize
- Residual networks tend to achieve high accuracy relatively quickly in comparison with other models

3. LITERATURE REVIEW

The implementation is divided into 3 sub-parts:-

1. Mask Detection
2. Face Recognition
3. Firebase and Python
4. Web Application

3.1 Mask Detection

In order to develop a custom face mask detector, we divided the task into two distinct phases, where each comprises of its own unique respective sub-steps (as shown in Figure 1 below):

1. Training: This is the first phase and the focus is on loading our unique face mask detection dataset from local storage and training a deep learning model (using Keras/TensorFlow) on the loaded dataset, followed by serializing the face mask detector model to local storage. [7]
2. Deployment: In this phase, the trained face mask detector model is deployed, the mask detector then performs face detection, and then classifies each face provided as input as 'with_mask' or 'without_mask'. [7]

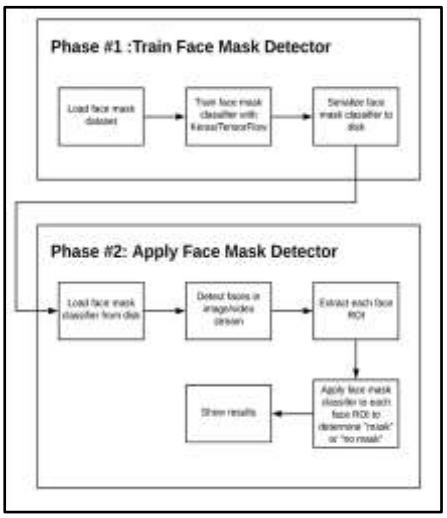

Figure. 1. Phases and individual sub-steps for building a face mask detector

To create a custom face-mask dataset, we employed the following solution :

1. Scrapping normal images of faces off the web.
2. Followed by, creating a custom Python script to add face masks to the above scrapped images of faces using computer vision and thereby creating an artificial dataset of face-mask which is still applicable to the real world.

To create a dataset of faces wearing masks, we had to first extract the faces from the images, this is achieved using facial landmarks. Facial landmarks enable us to automatically detect the coordinates of facial structures like Eyes, Eyebrows, Nose, Mouth and Jawline. From here, we applied these facial landmarks to compute the bounding box location of the face in the image. Once we were able to determine the coordinates of the face in the image, we extracted the face which is the Region of Interest (ROI) for our use case. And from there, applying facial landmarks to the extracted region of interest allowed us to localize the eyes, nose, mouth, etc. Next, we procured an image of a generic face mask with a transparent background such as the one portrayed in Figure 2.

Figure 2. An example of a face mask [10]

The mask in Figure 3 is then automatically appended to the face by using the extracted facial landmarks (namely the points along the chin and nose) to compute the exact location where the mask has to be placed. The above mask is then rotated and resized to place it accurately on the face. Thus, we were able to create an artificial face mask dataset by repeating this process for all our input images.

Now that we have created a dataset, we can continue with the training of the dataset. To get a trained model we'll be fine-tuning the ResNet101 architecture, a highly efficient architecture that can be applied to embedded devices with limited computational capacity.
To train the model we have used Keras/Tensorflow, which allows us to:

- Perform Data augmentation
- Load the ResNet 101 classifier (we will fine-tune this model with pre-trained ImageNet weights)
- Build a new fully-connected (FC) head
- Perform Pre-processing
- Load image data

Further, scikit-learn (sklearn) is used for segmenting our dataset, binarizing class labels (with_mask and without_mask), and printing a classification report. [6] Along with scikit-learn matplotlib is used to plot the training curves.

The first step to create a mask-detection deep learning model is to fine tune the ResNet101 architecture. Fine-tuning setup is a three-step process:

1. Load ResNet101 with pre-trained ImageNet weights, leaving off the head of the network.
2. Followed by constructing a new fully-connected head, and replacing the old head by this newly fully connected head.

3. The final step of fine-tuning is to freeze the base layers of the network. During backpropagation the weights of these base layers will not be changed, but the weights of the head layer will be tuned and updated.

Now that our dataset is prepared and the model architecture is fine-tuned, we can train our face mask detector network (ResNet101) using Keras and TensorFlow. The model once trained gave an accuracy of about 99% as shown in Figure 3.

Figure 3. Training outcomes

Now that our face mask detector is trained, the next steps are as follows :

- Load a live video stream.
- Detect faces in individual frames.
- Provide detected faces to the trained model to classify them as either 'with_mask' or 'without_mask'.

3.2 Face Recognition

Real-time Face Recognition is achieved by a method called deep metric learning.
A typical deep learning network is trained to:

- Accept a single input image
- And output a classification/label for that image.

However, deep metric learning is different. Instead of giving a single label as output or the coordinates of objects in an image, we are generating a real-valued feature vector as output. Considering the dlib facial recognition network as an example, the output real-valued feature vector for this deep learning network is a matrix of 128 real-valued numbers that is used to quantify a human face. Training for such networks is done using triplets.

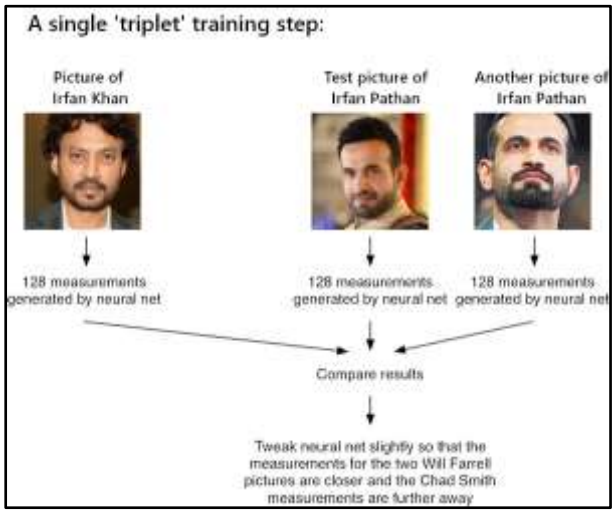

Figure 4. Facial recognition via deep metric learning using a "triplet training step."

In this implementation we provide three images to the network:

- Two of the provided images are of the faces of the same person.
- The third image is a random face which is not the same person as the other two images.

As an example, let's consider Figure 1 above where we provided three images: one of Irfan Khan and two of Irfan Pathan. The deep learning network then converts the faces into a numeric matrix, constructing the 128-d embedding (numeric matrix) for each.

From there, the general idea is that we'll tweak the weights of our neural network so that the 128-d measurements of the two Irfan Pathan will be closer to each other and farther from the measurements for Irfan Khan.

Our implementation of the network architecture for face recognition is based on ResNet-34 from the Deep Residual Learning for Image Recognition paper by He et al., but consists of fewer layers and the total number of filters is reduced to half. The above mentioned deep learning network was trained by Davis King on a dataset of approximately 3 million images, when tested on the Labeled Faces in the Wild (LFW) dataset the deep learning network proved to be comparable to other state-of-the-art methods, reaching an accuracy of 99.38%.

In order to perform face recognition with Python and OpenCV we employed two additional python libraries:

- dlib
- face_recognition

The dlib library which is maintained by Davis King, consists of the above explained implementation of "deep metric learning" which is used to extract face embeddings, these face embeddings are later employed for the actual recognition process. Along with this we employed the face_recognition library, created by Adam Geitgey which wraps around Davis King's dlib's facial recognition functionality, making it easier to work with.[1] 900

3.3 Firebase and Python:

To store the databases of students and staff, we used Firestore Database in Google's Firebase. In this we made four collections:

- users: To store each student's information.
- staff: To store each staff's information.
- defaulters: To store details of each defaulter student.
- defaulter_staff: To store details of each defaulter staff.

We used Firebase Storage to store all the photos of students, staff and the defaulters photos which were taken if a person without mask was detected.

In order to perform database operations on Firebase using Python, we need to install two additional libraries:

- Pyrebase : It is a Python library that provides an interface to Firebase's REST API. [8] In simple words, it allows us to use Python to manipulate Firebase databases.
- Firebase_admin : It is the Python Library consisting of modules for Firebase Admin SDKs which provides developers with programmatic, second-party authorisation functions for access to Firebase services from trusted environments.

3.4 Web Application

Now that we have developed a system which detects face masks, recognises a person if no mask is detected, and is able to

successfully upload the defaulters information on firebase, it is important to deploy this system in a real life scenario. We simulated a situation wherein the system was deployed in a college campus. We also developed a web application so that an administrator can reprimand the defaulters and maintain consistency in the database.

To develop a basic web application we used Google's Flutter framework [9].
This web application consists of the following 3 pages :

1. Dashboard: This page displays the statistics of defaulters. These statistics are represented in the form of graphs which are time based i.e. they represent the data for the past 7 days.
2. Defaulters: This page displays the information of defaulters for both staff and students. The user can easily switch between the two using a drop down menu.
3. Database: This page displays the information of all the students and staff present on Firebase. It also provides functionalities to add, delete and modify the contents of the data.

4. RESULTS

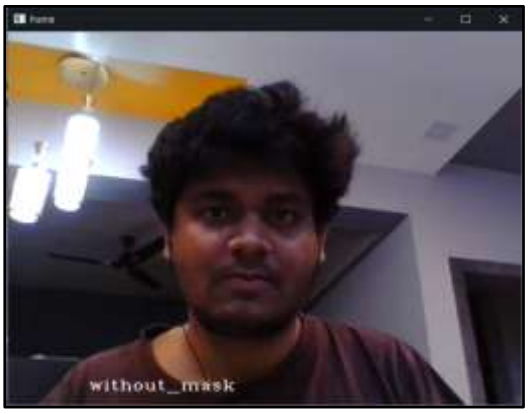

Figure 5. Mask detection(without mask)

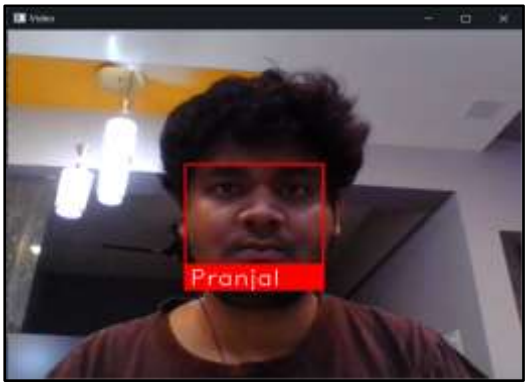

Figure 6. Face recognition

Figure 7. Mask detection(with mask)

Figure 8. Dashboard

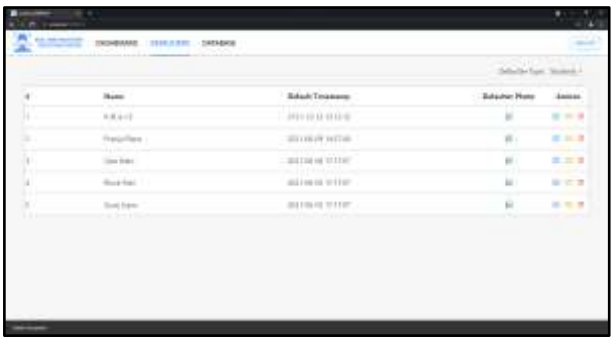

Figure 9. Defaulters

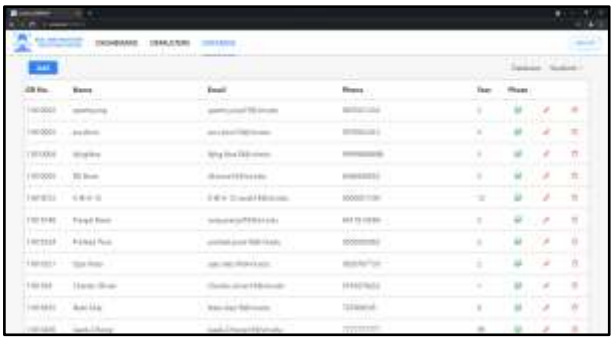

Figure 10. Database

5. CONCLUSION

In this paper, we presented an approach to minimize the spread of COVID 19 in crowded places by creating a system to detect people wearing face masks as well as identify the people not wearing face masks using face recognition in real time. Further, this was extended by keeping records of these defaulters in a database so that necessary action can be taken later. The mask detection was done with accuracy of more than 98% and face recognition of people with accuracy more than 99%. This solution could be implemented with various organizations across the globe since COVID 19 is a global pandemic.

6. REFERENCES

[1] Davis E. King. 2009. Dlib-ml: A Machine Learning Toolkit. J. Mach. Learn. Res. 10 (12/1/2009), 1755–1758.

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[3] Saravanan, Sharma & Shanmugasundaram, Karthikeyan & Ramasamy, Sathees. (2016). FAREC — CNN based efficient face recognition technique using Dlib. 192-195. 10.1109/ICACCCT.2016.7831628.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2015). Deep Residual Learning for Image Recognition.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference

on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[6] Pedregosa, Fabian & Varoquaux, Gael & Gramfort, Alexandre & Michel, Vincent & Thirion, Bertrand & Grisel, Olivier & Blondel, Mathieu & Prettenhofer, Peter & Weiss, Ron & Dubourg, Vincent & Vanderplas, Jake & Passos, Alexandre & Cournapeau, David & Brucher, Matthieu & Perrot, Matthieu & Duchesnay, Edouard & Louppe, Gilles. (2012). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 12.

[7] Train a Face Recognition Model to Recognize Celebrities https://algorithmia.com/blog/train-a-face-recognition-model-to-recognize-celebrities

[8] Pyrebase (Firebase's REST API) Documentation : https://pypi.org/project/Pyrebase/

[9] Building a web application with Flutter : https://flutter.dev/docs/get-started/web

[10] https://naparecycling.com/guide/face-masks/

BIBLIOGRAPHY

Pranjal Rane
Vishwakarma Institute of Technology, Pune, Maharashtra, India

Ritvik Patil
Vishwakarma Institute of Technology, Pune, Maharashtra, India

Ojas Natu
Vishwakarma Institute of Technology, Pune, Maharashtra, India

Prahlad Pore
Vishwakarma Institute of Technology, Pune, Maharashtra, India

Pranay Shridhar
Vishwakarma Institute of Technology, Pune, Maharashtra, India

