IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 9, Issue 4, August – Sep 2021
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

MASKED FACE RECOGNITION IN REALTIME USING CNN

Dr.A. Panja Prasad Sinha
Electronics and Communication Engineering
DPR College of Engineering, Ludhiana

P Yathvik Sai Trinadh
Electronics and Communication Engineering
DPR College of Engineering, Ludhiana

M Yaswanth Kumar
Electronics and Communication Engineering
DPR College of Engineering, Ludhiana

ABSTRACT: Coronavirus has intensely impacted the planet and has now infected more than 169 million people worldwide. In this pandemic situation, wearing a mask and following interpersonal distance are two of the safety protocols to be followed to avoid the escalation of the virus. In this scenario wearing masks everywhere was mandatory and becomes challenging to identify a person with a mask. So, we came up with the idea of "Masked Face Recognition" to create a safe environment that contributes to public safety. In this paper, we propose a reliable method to detect masked faces in real-time based on FaceNet, Convolution Neural Network, and deep learning techniques in python.

Keywords: Masked Face Recognition, FaceNet, Convolution Neural Network, Deep learning

I. INTRODUCTION

Nowadays, in many organizations, the presence of a person is collected using biometric attendance machines [1], which consists of fingerprint sensors. A person is made to touch the sensors, which leads to the rapid spread of disease. Face recognition is a widely used technique[2] in our day-to-day life, especially in security surveillance, innovative home automation systems, criminal investigations, unlocking devices, and intelligent attendance systems. Face recognition can be implemented using various algorithms in machine learning and deep learning.

Coronavirus is sweeping the globe now. Coronavirus is an unstoppable illness caused by a severe respiratory condition [3] (SARS-CoV-2). Coughing, sneezing, touching items, and rubbing eyes are all ways for people to become infected. As a result, masking and social distancing became necessary during the pandemic. A person wearing a mask is more difficult to recognize. There are some models developed for masked face recognition such as integration between classical machine learning and deep learning techniques with Keras, TensorFlow, and OpenCV by Velantina[4]. Walid Harari designed a model based on deep learning and occlusion removal-based features[5] and some of them worked on Intelligent Face Mask Detection System Performance Evaluation using Deep Learning classifiers[6].

Masked Face Recognition is a technique for recognizing a person who is wearing a mask. In this model, we have used methods like Convolution Neural Network (CNN) and Facenet.

II. CNN ARCHITECTURE

CNN is a type of artificial neural network which is also named as ConvNet. To process pixel data and recognize an image we use a convolution neural network. CNN falls under deep learning which is a subset of machine learning. Deep learning uses algorithms that are inspired by the function and structure of the brain's neural networks. Deep learning makes CNN more potent in image processing and artificial intelligence to perform both descriptive and generative tasks. Through the application of suitable filters, a ConvNet may successfully capture the Spatial and Temporal dependencies in a picture. CNN is made up of several neurons which are spread over various hidden layers[7]. A neuron is a mathematical function that functions similarly to a biological neuron. Here the input data is multiplied with the weights and added up to the bias. The weights get updated as they learn from each iteration of the training process. These weights are nothing but the parameters in between layers.

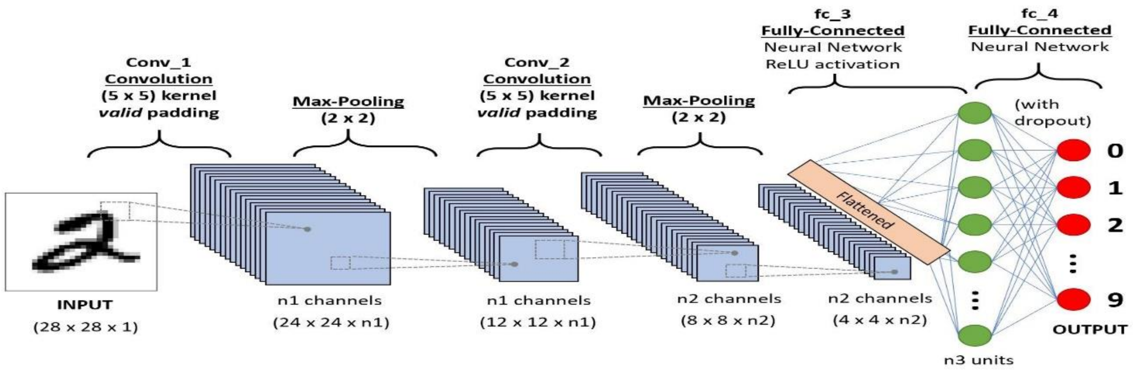

Fig 1: CNN ARCHITECTURE

CNN consists of three layers, namely the input layer, hidden layer, and output layer as shown in Fig 1. There might be any number of hidden layers in between the input layer and the output layer. On average, the number of neurons present in hidden layers should be two-third of the neurons present in the input layer in addition to the output layer. In CNN, the hidden layer consists of the pooling layers, convolution layers, normalization layers, and fully connected layers. There are mainly four types of models present in CNN. They are nn4.v1, nn4.v2, nn4.small1. v1, nn4.small2. v1. Out of these, we use the variant nn4.small2.v1. The main reason behind opting for this type of model is that it considers only the outer eyes and nose part of a face for aligning images, shows better accuracy, less CPU and GPU run time even with a fewer number of parameters (3733968) when compared to others[8].

Table 1: Performance and accuracy benchmarks of CNN models

MODEL	NUMBER OF PARAMETERS	RUNTIME(CPU)	ACCURACY
nn4.small 2	3733968	58.9ms ± 15.36ms	0.9292± 0.0134
nn4.small 1	5579520	69.58ms± 16.17ms	0.9210± 0.0160
nn4.v2	6959088	82.74ms± 19.96ms	0.9157± 0.0152
nn4.v1	6959088	75.67ms± 19.97ms	0.7615± 0.0189

Inception: Here in our model, we have made use of the technique called inception. The main motto of the inception layer is to convert a deeper model into a wider model so that some of the processes can be done parallelly as shown in Fig 2. Inception is a model in which it consists of 1x1, 3x3, 5x5 convolution filters and the 3x3 max-pooling layer in between the previous layer and the next layer. The main motive behind opting for those three convolution filters is that the region of interest varies for every input image; we never know how to decide the correct kernel size. So, we use all three convolution filters and max-pooling to address the above issue. concatenate all the outputs obtained from each filter. We must ensure that we use a 1x1 convolution filter before using 3x3 and 5x5 convolution filters to reduce the computation cost. The computation cost of applying a 5x5 filter directly on large input data is nearly ten times more than using a 5x5 filter after a 1x1 filter. When we use max pooling, the output obtained from this layer is converted to the exact dimensions as input by applying zero paddings, which adds zeroes on all four sides of an input matrix and makes sure that the corner values of that input matrix have a fair bit of participation in the feature extraction. All the outputs obtained from each filter will be concatenated and sent to the next layer.

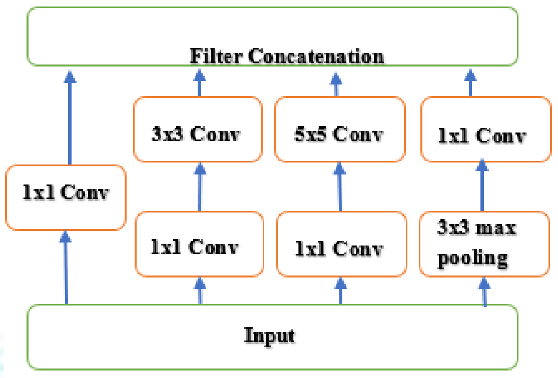

Fig 2: Inception layer

Convolutional Layer: A convolution layer is nothing but a filter that extracts a specific type of feature from an image. Here the feature extraction can be an eye detector, ear detector, or nose detector in case of recognizing a person and changes accordingly depending on the purpose we use the neural network. Each neuron in the layer is automatically trained to filter for a particular kind of feature. The input given to this layer is convoluted with a kernel to get the output which is known as a feature map. The values present in the kernel decides which feature to extract. Here we have used the function conv2D, which considers the submatrix in the input of the same size as the kernel and performs convolution, which is followed by considering all possible submatrices of the input depending on the kernel size and the stride which results in a feature map.

Pooling: Most of the time, the convolution layer is followed by the pooling layer. Position invariant feature detection will be aided by pooling and convolution. As the name says, pooling represents all the values present in the window/submatrix with a single value. Here the window size and the strides are represented in the input function of this layer. This layer is primarily used to minimize the complexity, number of calculations and decrease overfitting due to the fewer parameters, tolerance to fluctuations, and distortions. The two types of pooling procedures that are most employed are maximum pooling and average pooling. The maximum value of all the values present in the filter is considered in max pooling. In contrast, the average value of all the values currently in the filter is considered in average pooling. When opposed to average pooling, max pooling is more commonly utilized since it finds the maximum values by discarding other matters, which leads to the elimination of disturbances.

Normalization: The term "normalization" refers to the process of converting input data to a range between 0 and 1 with a mean of 0 and a standard deviation of 1. This technique is mainly helpful in cases where we have a wide range of input values. Generally, we use this functionality after passing through the activation function. An activation function defines a neuron’s output given a set of inputs after calculating the weighted sum by a nonlinear transformation. The runtime can be decreased by using the normalization function at the end of each layer. So, that it allows faster learning.

Fully Connected Layer: At last, after passing through all the convolution layers, pooling layers, normalization layers present in the model, the output obtained is flattened and is given as input to the fully connected layer. The purpose of flattening is to convert an n-dimensional array into a 1-D array. The purpose of the fully connected layer is to predict the output from all possible outcomes. So that the learning of nonlinear combinations of these features can be done cheaply. Each neuron that is present in the preceding layer is connected to the fully connected layer as inputs. On the other hand, this layer is linked to all of the model's possible outputs.

III.FACE NET

Face Net learns from an embedding function about two images are similar to each other. An embedding in mathematics is one instance of a mathematical structure enclosed within another instance, such as a subgroup of a group. Here x is an input image or an input face image. So, from this face image this FaceNet[9] learns an embedding function *f(x)* with the constant that $||f(x)||^2 = 1$, that is for normalization purposes. In other words, we can write that the Face Net learns a function f, which maps your input face image x; of course, here we assume that this input face image x is of size MxN; so MxN number of pixels. So, it learns a mapping or learns an embedding of an input face as x to R^d, d dimensional feature vector. So, it maps x, which belongs to R^{MxN}, *to* $R^{d,}$ and if $d < MxN$, then what we are gaining is an image of size MxN being represented or embedded, which is known as embedding. It is being embedded into a d dimensional space. This embedding function, which is f that embeds an input image to a d dimensional vector or represents that input image embeds it in this input image to our d dimensional space as a vector, is not handcrafted. But the machine or the deep neural network has to learn this embedding function f; by using the training data made available to the system. When we transform an image into a vector, in a d dimensional vector, it represents that image by a point in a d dimensional space, and the same thing is done over here. If we take two images, say x_i and x_j , both of them are embedded in that d dimensional space and maybe in Euclidean space through this embedding function f. So, $f(x_i)$ is embedding image $x_{i,}$ and $f(x_j)$ is embedding image x_j . So, $||f(x_i)-f(x_j)||^2$ This indicates the squared Euclidean distance between these two embeddings $f(x_i)$ and $f(x_j)$.

Triplet Loss: For training this network, what we need to define is a loss function. So, the loss function described in this Face Net is known as a triplet loss[10]. The triplet loss function makes use of three images that is the negative image(N), positive image(P), anchor image(A), and their respective embeddings are f(N), f(P), f(A). A positive image is an image of the same person present in the anchor image, whereas a negative image is quite opposite of a positive image. The distance between the negative image and the anchor image should exceed the distance between the positive image and the anchor image as shown in Fig 3.

$$||f(x^a)-f(x^p)||_2{}^2<||f(x^a)-f(x^n)||_2{}^2$$

If $f(x^a) - f(x^p)$ equals zero, then the above condition is always actual, which means there will be no further learning as there will be no triplet loss function. So, the model stops learning. The above equation can be modified such that the distance between these two must differ at least by alpha. And that leads to a loss function, which is given by this that ${}^N\sum_{i=1}[||f(x^a)-f(x^p)||_2{}^2-||f(x^a)-f(x^n)||_2{}^2$ Face Net tries to make compact clusters of the images belonging to the same person. The distance between every such pair should be small.

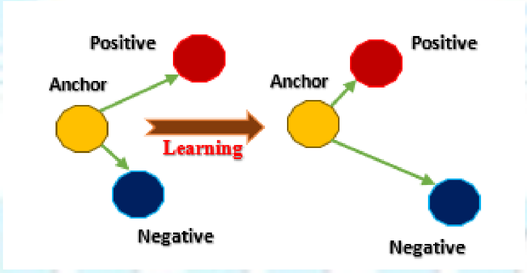

Fig 3: Triplet Loss

Triplet Selection: If we have a database of a vast number of images, selecting all possible triplets becomes exceedingly difficult. So, we form the mini-batches and select the triplets within the mini-batches. We should find out the pair of tricky positive triplets or a positive triplet whose distance is considerable, and a negative triplet whose distance from the anchor is small. These are the pairs that give you faster learning of this network. Instead of finding out the most challenging negative sample, we need to find a semi-hard adverse selection that satisfies the distance between the negative and the anchor sample exceeds the distance between the positive and the anchor sample.

The samples which are semi-hard negative samples may not be complex negative samples because the selection of those may be more straightforward. So, once you have all this and your network is trained. Then, the output becomes a vector embedding your input image to our d dimensional space for any input image.

IV. PROPOSED WORK

Pre-Processing: In pre-processing, we first need to collect the images of all the persons to be detected using our model and place them in a folder with the person folder name. It is better to give more images

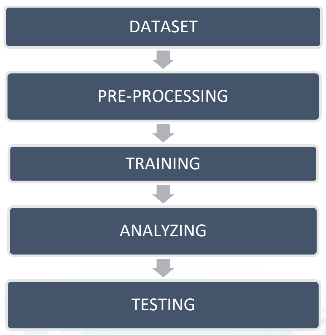

Fig 4: Flow

of each person to get an efficient output. Collect the images of a person with his face masked or unmasked as shown in Fig 6. Unmasked images can be masked using an imaginary mask[11] by taking all the images present in our dataset and make sure that it paste a picture of a mask to every image by considering the nose bridge and chin bottom of a face. We will get a folder where all the images present in our dataset will have masks on their face, and we call this process generating a dataset. Now we need to take all the photos generated above and align them so that the images are cropped into the size 96x96 by selecting only the frontal part of our face using the hog detector[12] as shown in Fig 7. This hog detector is used to detect these features using Dlib algorithms by surrounding each feature with a map of points as shown in Fig 5 which is composed of 68 points also known as landmark points. Every image is aligned so that all the 68 landmarks[13] of every painting are present one above the other. Here the hog detector makes sure that every image in the generated dataset contains the person's face. If there is no face detected, it will discard that image from the generated dataset.

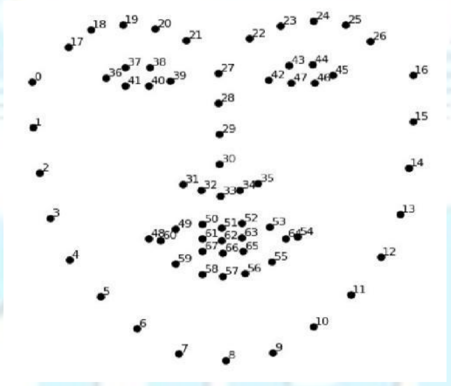

Fig 5: 68 Facial Landmarks

Initialization: We need to initialize our model by calling the function in which we have given all the connections that were made in the model. Now we will initialize all the neurons present in our model with predefined weights which are useful for human facial features extraction. Weights define the importance of all these features in deciding the final output.

Training: Here, a list of all the images present in the dataset is created and assigned labels to every image.

Fig 6: Dataset

The embedding vector[14] of all the images present in the dataset is to be calculated. For this to happen, it will take all the photos from the dataset and align them as mentioned in the pre-processing stage and normalize all the RGB values obtained and send them for prediction. The predicted value is obtained as a 128-dimensional embedding vector when we send the normalized RGB values as input to the model created. Embedding vector is a point in the 128-dimensional plane, and all the vectors of a particular person will be in the form of a cluster. All these embedding vectors generated will be concatenated into an array and will be saved in a file.

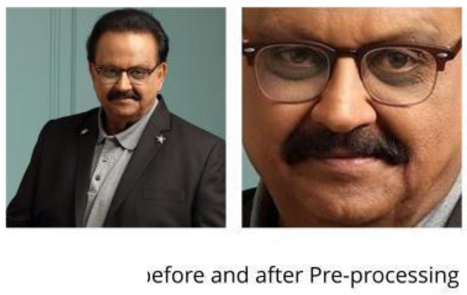

Fig 7: Image before and after pre-processing

Analyzing: The process of analyzing is done to find a preferable threshold value. Here we will discover the euclidean distance between all possible combinations of the two embedding vectors of a particular individual using the equation shown below. This cycle is rehashed for every one of the people present in the dataset. The Euclidean distance of an image obtained from each case is appended into a list named match_distances.

$$D_{ij}^2 = \sum_{v=1}^{n} (X_{vi} - X_{vj})^2$$

In the same way, we will find out the Euclidean distance between the embedding vectors of two different individuals in a random way. Here we can indicate quite a few combinations, and the output distance obtained in each case is appended into a list named unmatched_distances. When a graph is plotted for both matched and unmatched distances, as shown in Fig 8, we can depict the tradeoff value for the threshold so that most of the values present in matched_distances should be present on the left side of the threshold value we have chosen.

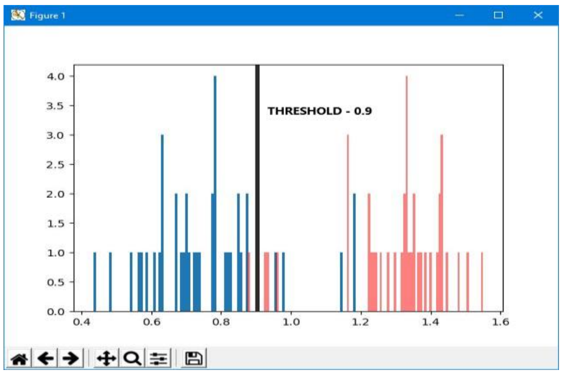

Fig 8: Plot for matched and unmatched distances

Testing: As we all know, when we run a sequence of images or frames, it results in a video. The image captured from each frame detects the number of faces present in it and crops them so that all the faces that are seen at run time should be of size 96x96 by only considering the frontal part of a face. Images are aligned one over the other, so all the 68 landmarks are at the same position, quite similar to the process done in pre-processing. Now, these aligned images are converted into the RGB values of each pixel and normalize those values. These normalized RGB values are sent as input to the model and result in a 128-dimensional embedding vector. This vector obtained at run time will find the euclidean distance with each image present in our dataset and get appended to the list. The minimum value of all the current weights in the list is compared to the threshold value found earlier. If the minimum value obtained is less than the threshold value, it will mention that person with which we got the minimum value as shown in Fig 8. But if the minimum value exceeds the threshold value, then it will show the person captured at run time at that frame as unknown. This process continues for every frame until we stop executing it.

V. EXPERIMENTAL RESULTS & ANALYSIS

The accuracy of the proposed model is mainly calculated using images that are not trained in the training part and given as inputs for the model for calculating the accuracy—the two scenarios needed to be considered for accurate calculations. The first scenario will be viewing the photos of the person without a mask. The second scenario will be considering the images of the person with and without a mask. With the consideration of the above methods, the accuracies obtained are

Table 2 : Accuracy rates

SCENARIO	ACCURACY
Scenario 1	87%
Scenario 2	84%

Here, an SVM classifier is used to calculate the accuracy of the model. In SVM, a hyperplane is a plane in n-dimensional space that tries to separate different classification groups. The rate of similarity(confidence score) for each sample is calculated by considering the proportional relation to the signed distance of the same sample to the hyperplane. Using the confidence score obtained the class labels are predicted. The accuracy required is calculated by dividing the total number of correctly predicted samples by the total number of samples.

$$\text{Accuracy} = \frac{\text{correctly predicted samples}}{\text{Total number of samples}} \times 100$$

The outcomes may be varied when we test it with the webcam feed or any other cam feed. Our experimental results are accurate when the person is nearer the camera. The face should be visible clearly without any shadow or dullness present in the feed (An adequate amount of light should be present in the area to better capture the image).

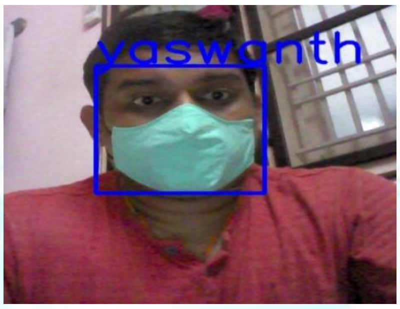

Fig 9: Output of the model

VI. CONCLUSION

Face recognition gets a challenging task in this pandemic. We also know that recognizing a person's face is challenging since current facial recognition technologies will almost likely fail to make an accurate identification while wearing a mask. As a result, we proposed CNN-based "Masked Face Recognition" to detect a masked person's face. Using the above-proposed model of CNN, we can recognize a person wearing a mask with reasonable accuracy.

VII. REFERENCES

[1] M. A. Meor Said *et al*., "Biometric attendance," *2014* International Symposium on Technology Management and Emerging Technologies, 2014, pp. 258-263, DOI: 10.1109/ISTMET.2014.6936516.

[2]. Turk, Matthew A., and Alex P. Pentland. "Face recognition using eigenfaces," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 586-591.

[3] C. I. Paules, H. D. Marston, and A. S. Fauci, “Coronavirus infections—more than just the common cold,” Jama, vol. 323, no. 8, pp. 707–708, 2020

[4]Vinitha.V1, Velantina.V2, “COVID-19 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION”, International Research Journal of Engineering and Technology (IRJET), Volume: 07, pp.1-6, Aug 2020.

[5]Walid Hariri, “Efficient Masked Face Recognition Method during the COVID-19 Pandemic”, pp.1-7, July 2020

[6]C.Jagadeeswari, M. Uday Theja, “Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers”, International Journal of Advanced Science and Technology, Vol. 29, No. 11s, pp.3074-30780, (2020)

[7] S. Albani, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, DOI: 10.1109/ICEngTechnol.2017.8308186.

[8]https://cmusatyalab.github.io/openface/models-and-accuracies/

[9] Florian Schroff, Dmitry Kalenichenko, James Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823.

[10]Xingping Dong, Jianbing Shen; Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 459-474

[11] Aqeel Anwar1, Arijit Raychowdhury2 "Masked Face Recognition for Secure Authentication" arXiv:2008.11104v1 [cs.CV] 25 Aug 2020.

[12] Dalal, Navneet, and Bill Triggs. “Histograms of oriented gradients for human detection.” Computer Vision and Pattern Recognition (CVPR), 2005.

[13] L. Kalapala, H. Yadav, H. Kharwar and S. Susan, "Facial Expression Recognition from 3D Facial Landmarks Reconstructed from Images," 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC), *2020*, pp. 1-5, DOI: 10.1109/iSSSC50941.2020.9358815.

[14] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification.” Journal of Machine Learning Research, vol. 10, no. 2, 2009.