Real Time Face Mask Detection For Preventing the Spread of COVID-19 Using TensorFlow, Keras and OpenCV

Pierjos Francis COLERE MBOUKOU
colerepierjos30@gmail.com
https://pierjos-colere-website.web.app

Mohammed VI Polytechnic University, Al-Khwarizmi, Data Science
Hay Moulay Rachid, Ben Guerir 43150
https://www.um6p.ma

Abstract. Coronavirus disease (COVID-19) spread rapidly in Wuhan-China in December 2019 is a dangerous virus which can be spreading from humans to humans through droplets and airborne. It has rapidly affected our day-to-day life disrupting the world movements insofar as wearing a protective face mask has become a new normal. However, some irresponsible people refuse to wear face mask with so many excuses. Moreover, developing the face mask detector is very crucial in this case to help global society. This paper presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, OpenCV and Scikit-Learn. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. The method attains accuracy up to 98.5% and 97% on our test and validation data sets.

Keywords: Covid-19, Machine Learning, Face Mask Detection, Sequential Convolutional Neural Network, TensorFlow, OpenCV, Keras, Scikit-Learn

1 Introduction

According to the World Health Organization (WHO)'s official Situation Report, COVID-19 has globally infected over 148 million people causing over 3 million deaths [1]. The COVID-19 spreads through air channel when an infected person sneezes or communicate with the other person, the water droplets from their nose or mouth disseminate through the air and affect other peoples in the vicinity [2]. Wearing a mask during this pandemic is a critical preventive measure [3] and is most vital step in times when social distancing is hard to maintain and is essential, particularly for those people who are at a greater risk of severe illness from COVID-19 diseases. Therefore, to protect each other, every person should wear the face mask properly when they are in outdoor. However, most of selfish people won't wear the face mask properly with so many reasons. So, to overcome certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. This paper presents a simplified approach to serve the above purpose using the basic Machine Learning (ML) packages such as TensorFlow, Keras, OpenCV and Scikit-Learn.

The rest of the paper is organized as follows: Section 2 explores related work associated with face mask detection. In Section 3, we define certain terms. Section 4 discusses about materials and method used. Results and analysis are reported in Section 5. Section 6 concludes and draws the line towards future works.

2 Related Work

The goal of face detection is to determine if there are any faces in the image or video. If multiple faces are present, each face is enclosed by a bounding box and thus we know the location of the faces. In face detection method, a face is detected from an image that has several attributes in it. Given an image, the challenge is to identify the face from that image. Human faces are difficult to model as there are many variables that can change for example facial expression, orientation, lighting conditions and partial occlusions such as sunglasses, scarf, mask etc. The result of the detection gives the face location parameters and it could be required in various forms, for instance, a rectangle covering the central part of the face.

Therefor We used Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces [4]. Viola-Jones algorithm is named after two computer vision researchers who proposed the method in 2001, Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features". Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. The algorithm looks at many smaller parts of images and tries to find a face by looking for specific features in each part. It needs to check many different positions and scales because an image can contain many faces of various sizes.

3 Terms explaination

Before explaining our method named convolutional neural network model, we have to explain certain terms for best understanding.

3.1 Convolutional Neural Network

A convolutional neural network (CNN, or ConvNet) is a class of deep neural network, most commonly applied to analyze visual imagery (computer vision tasks) [5] like images detection, facial recognition, digital recognition, etc. Central to the convolutional neural network is the convolutional layer that gives the network its name.

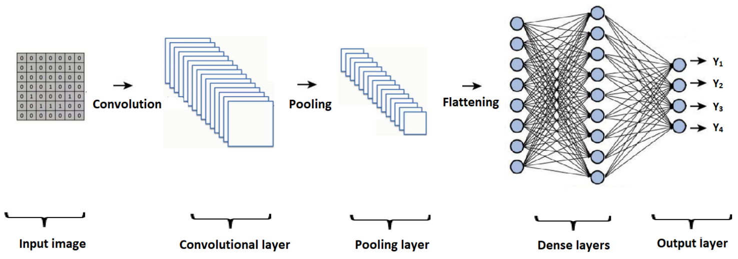


Fig. 1. Architecture of a convolutional neural network (CNN).

3.2 Convolutional Layer

The convolutional layer computes the convolutional operation of the input images using kernel filters to extract fundamental features. This layer is the fundamental block of the Convolutional Neural Network. The term convolution implies a mathematical combination of two functions to get the third function like below.

$$C(T) = (I * K)(x) = \int_{-\infty}^{+\infty} I(T) \times K(T - x)\, \mathrm{d}T$$

It specializes in reducing the size of the input matrix but keeping the important features. This means it detects only the features that contribute the most to the image.

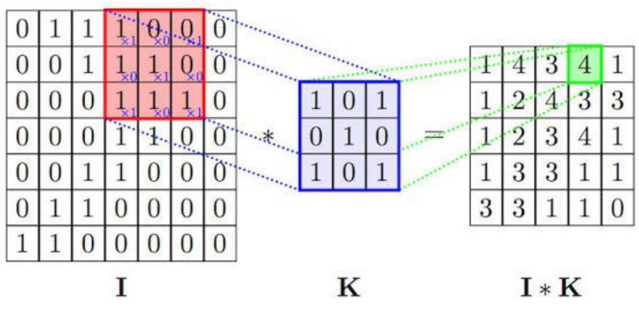


Fig. 2. Convolution Layer Process.

3.3 Pooling Layer

The pooling layer reduces the number of parameters and computation by downsampling the representation. A pooling layer is usually incorporated between two successive convolutional layers. The pooling function can be max or average. We used Maxpooling in our method.

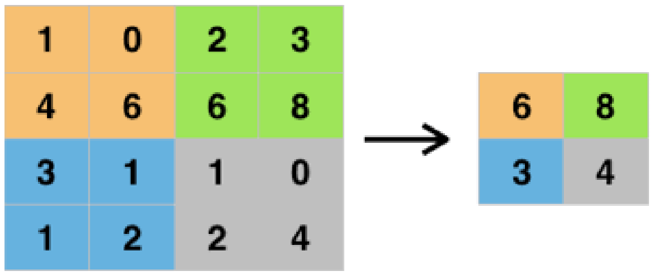


Fig. 3. MaxPooling Layer Process

3.4 Flattering Layer

The transition from bidimensional (or multidimensional) layers to one-dimensional fully connected layers requires a special reshaping operation called "a flatten layer."

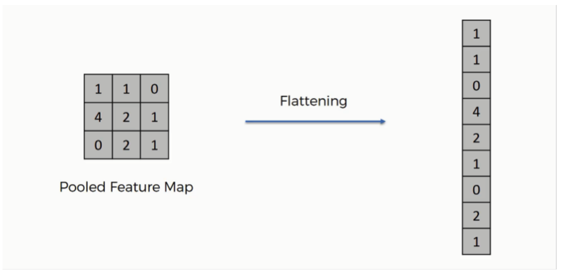


Fig. 4. Flattening Layer Process

3.5 Dropout Layer

This layer reduces the overfitting, which may occur while training by dropping random biased neurons from the model. The likelihood for a neuron to be dropped can be changed by changing the dropout ratio.

4 Materials and Method

4.1 Databases

To build the model that detects if we wear a face mask or not, we combined two databases our datasets which contain images.

The first one was created by Prajna Bhandary [6]. Dataset consists of 1376 images in which 690 images with people wearing face masks and 686 images with people who do not wear masks. She took standard images of faces and applied facial landmarks. Facial landmarks allowed to locate facial features of a person like eyes, eyebrows, nose and mouth. This used an artificial way to create a dataset by including a mask on a non-masked person image. Fig. 1 shows the facial landmarks result with face mask.

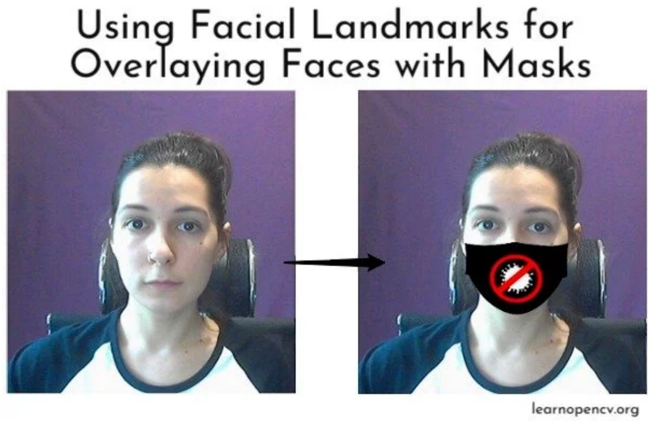


Fig. 5. Facial Landmarks Result, Credits [7]

The second one is from Kaggle [8]. It consists of almost 12000 images. Around 6000 images with the face mask are scrapped from google search and all the images without the face mask are preprocessed from the CelebFace dataset created by Jessica Li [9]. So by cpmbining these two datasets, we have, in total, 13376 images.

4.2 Packages details

To build our face mask classification model, we needed some important packages.

4.2.1 TensorFlow

TensorFlow is an interface for expressing machine learning algorithms developed by Google [10]. It is a comprehensive and flexible ecosystem of tools, libraries and other resources and is utilized for implementing ML systems such as sentiment analysis, voice recognition, computer vision, text summarization, etc. It is also used to reshape the image in the data processing.

4.2.2 Keras

Keras is a high-level neural networks library that is running on the top of TensorFlow. The core data structures of Keras are layers and models [11]. All the layers used in the CNN model are implemented using Keras. This framework written in Python code is easy to debug and allows ease for extensibility.

4.2.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and ML software library. It is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, etc [12]. Our method makes use of these features of OpenCV in resizing and color conversion of data images.

4.2.4 Scikit-Learn

Scikit-learn (also known as sklearn) is a free software machine learning library for the Python programming language [13]. We use it for performance metrics such as confusion matrix, classification report.

4.3 Data Preprocessing

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be tables, images, videos, etc. Our method deals with image and video data using Numpy and OpenCV.

4.3.1. RGB image to Gray image

After labelling our images (0 or 1) where 0 means no mask and 1 with mask, we need to convert them to Gray images. Nowadays, modern image recognition systems regularly work on grayscale images. Introducing nonessential information could increase the size of training data required to achieve good performance. Grayscale is utilized for extracting forms instead of working on color images instantaneously.
CNNs require a fixed-size input image. Therefore we need a fixed common size for all the images in the dataset. So the gray scale image is resized into 100 x 100.

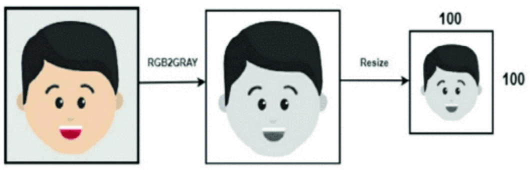


Fig. 6. RGB image to a Gray Scale image of 100x100 size

4.3.2. Reshaping images for the model

Most convolutional neural networks are designed in a way so that they can only accept images of a fixed size. This creates several challenges during data acquisition and model deployment. The common practice to overcome this limitation is to reshape the input images so that they can be fed into the networks. So the images are normalized to converge the pixel range between 0 and 1. Then they are converted to 4 dimensional arrays using data=np.reshape(data,(data.shape[0], 100,100,1)) where 1 indicates the Grayscale image. We can imagine that the final layer of the neural network has 2 outputs – with mask and without mask, the data labels are converted to categorical labels.

4.3.3. Splitting data

After that, to avoid overfitting, we split the data into training and testing sets to evaluate them. 90% data of the data set undergoes training and the rest 10% goes for testing purposes. Then 20% of the training data is used as validation data.

4.4 Proposed Approach or Method

The proposed methodology has been clearly explained using the two algorithms as shown below. In Algorithm 1 shown below, images were taken as an input, resized and normalized. Data was splitted into training and testing batches. Adam optimizer was used to compile the whole model. Categorical crossentropy which is also known as multiclass log loss is used as a loss function (the objective that the model tries to minimize). As the problem is a classification problem, metrics is set to "accuracy". Then the model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting. This process is well explained in Fig. 7

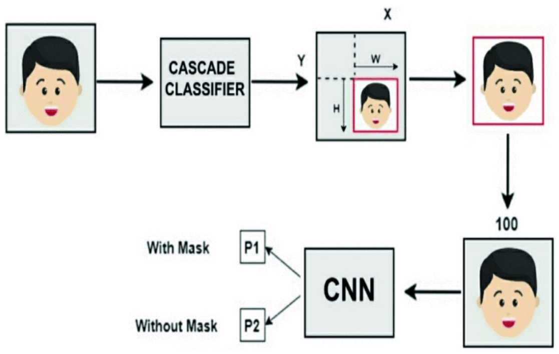


Fig. 7. Overview of the Model

Algorithm 1 : Face Mask Detection Model

```
INPUT: Image dataset including faces with mask and without masks
```

```
OUTPUT: Trained Model
STEP 1: Split dataset into train, test data
STEP 2:
for each image in dataset do
    Label image (0 or 1). Where 0 means no mask and 1 with mask
    Convert the RGB image to Gray-scale image
    Resize the gray-scale image into 100 x 100
    Normalize the image and convert it into 4 dimensional array
end

STEP 3:
for building the CNN model do
    Add a Convolution layer of 200 filters
    Add a Pooling layer
    Add the second Convolution layer of 100 filters
    Add a Pooling layer
    Insert a Flatten layer to the network classifier
    Add Dropout Layer
    Add a Dense layer of 50 neurons
    Add the final Dense layer with 2 outputs for 2 categories
end
STEP 4: Train the model on 80% of train data and validate it on
                                            20 % train data left.

STEP 5: Test the model on test data
STEP 6: Save model which has best accuracy
```

In Algorithm 2 the model trained in previous part was then deployed. If the faces are detected using Voila-Jones algorithm, a bounding box showing the face of the person is shown in the output. Our main challenge is to detect the face from the image correctly and then identify if it has a mask on it or not. The system should also detect a face along with a mask in motion. So this system runs in real-time application when the camera detected the user who wear or does not wear the face mask. If the person does not wear face mask from the camera, it will alert him/her to wear the face mask and will be going on until the person put his/her face mask properly.

Algorithm 2 : Face Mask Detector Deployment

```
INPUT: Camera Information
OUTPUT: Images classified into mask or no mask / Real-time
                                            Classification
STEP 1: Load saved classifier (model) from disk and load face
                                        detector from OpenCV
STEP 2: Load real-time feed from OpenCV
        Read the feed frame by frame (1 sec)
```

```
STEP 3: Apply face detection model to Detect faces in frames read
                                                    in real-time
STEP 4: If faces are detected:
            Crop face to bounding box coordinates from face
                                            detection model

            Get predictions from the face classifier model
            Show output with boxes in real-time feed
            If no mask is detected on face:
                Alert the person to put his/her mask
        Else:
            Show normal feed
```

5 Result And Analysis

The model is trained, validated and tested on a laptop equipped by an Intel i7 processor (16 GB of RAM). The Jupyter Notebook software equipped with Python 3.8 kernel was selected in this project.

5.1 Evaluation of CNN model

The metrics selected for evaluation of our CNN model are explained below.

$$Accuracy = \frac{Tp + Tn}{Tp + Tn + Fp + Fn}$$

$$Precision = \frac{Tp}{Tp + Fp}$$

$$Recall = \frac{Tp}{Tp + Fn}$$

Where :
Tp = True Positive means images which were labelled true and after prediction by model gave true result.
Tn = True Negative refers to images which were labelled true but after prediction resulted in false result.
Fp = False Positive means images which were labelled false and after prediction resulted in false hence false positives.
Fn = False Negative refers to images which were labelled false and after prediction resulted in true hence false negatives. Precision is a metric that quantifies the number of correct positive predictions made. That means minimizing false positives. And recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made; it minimizes false negatives. After 16 epochs the training accuracy came out to be 98% and 96% for validation accuracy as shown below Fig. 8

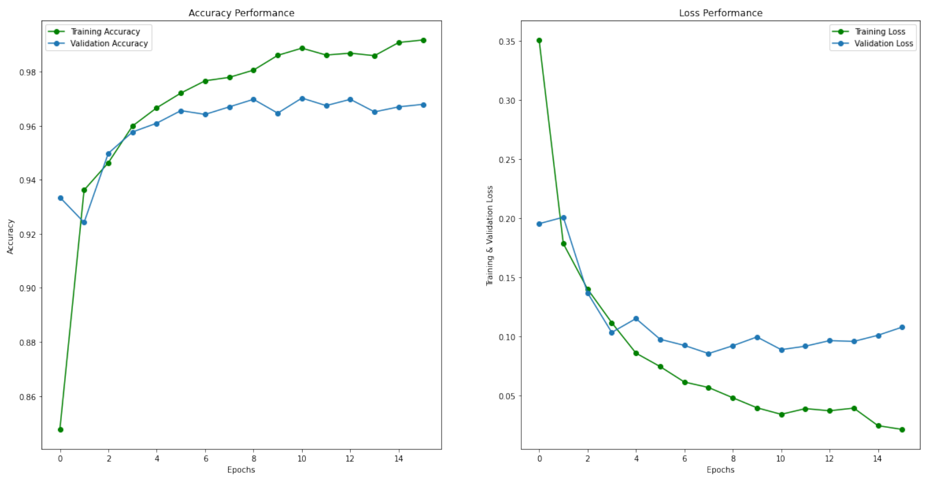


Fig. 8. Model performances (Accuracy and Loss)

The confusion matrix applied on test data and shown in Fig. 9 depicts a form to compare the labels, model prediction, and actual labels it was supposed to predict. It is showing where the model is getting confused. It has successfully identified 1336 true positives, 15 false negatives, 24 false-positive and 1314 true negatives. The classification report shown in Fig. 10 explains the level of f1 score, recall, precision, and accuracy of our CNN model.

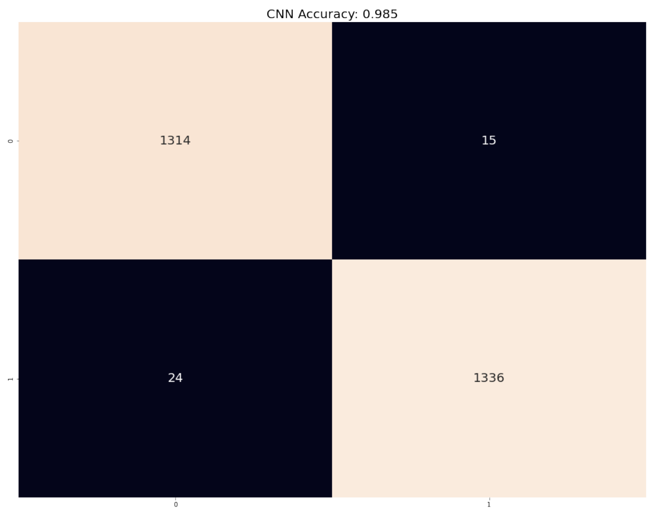
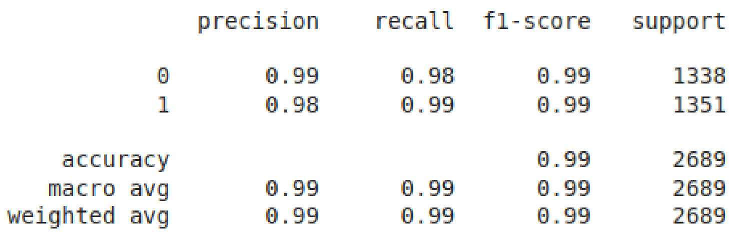


Fig. 9. Model Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1338
           1       0.98      0.99      0.99      1351

    accuracy                           0.99      2689
   macro avg       0.99      0.99      0.99      2689
weighted avg       0.99      0.99      0.99      2689
```

Fig. 10. Classification report

5.2 Visualization of results

Fig. 11 shows us the predictions on some images. These are the predictions made on 2 test images. The rectangular green box depicts the correct way of wearing a mask with an accuracy score on the top left, while the red rectangular box represents the incorrect way of wearing a mask.

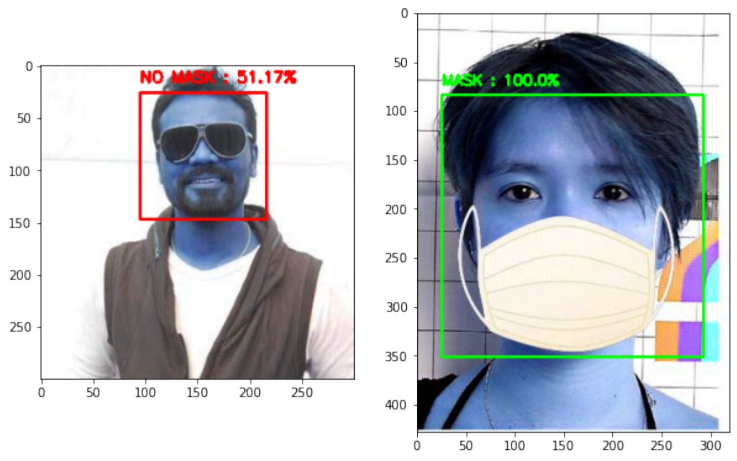


Fig. 11. Predictions on test images

The system can detect occluded faces either with a mask or hand. It considers the occlusion degree of four regions – nose, mouth, chin and eye to differentiate between annotated mask or face covered by hand.

6 Conclusion and Future Work

This work developed the face mask detection by using CNN Algorithm. In this paper, we briefly explained the motivation of the work at first. Then, we illustrated the learning and performance task of the model. Using basic ML tools and simplified techniques the method has achieved reasonably high accuracy. From the analysis and results, the algorithm is able to detect and distinguish a non-wearing and a wearing-mask precisely. This project can be used in numerous applications, such as autonomous driving, education, surveillance, and so on. Many public service providers will ask the customers to wear masks correctly to avail of their services. In future, if COVID-19 persists, we will implement this system in the university cafeteria. We had noticed several people not wearing their masks when purchasing a product. This endangers the UM6P community. And we hope this will be installed in other crowd areas which need face mask detector.

References

1. WHO, `https://covid19.who.int/`
2. P. Kumar, S. Hama, H. Omidvarborna, A. Sharma, J. Sahani, K.V. Abhijith and A. Tiwari, Sustainable Cities and Society, 62 (2020), Article 102382
3. A.M. Rahmani and S.Y.H. Mirmahaleh, Sustainable Cities and Society (2020), Article 102568
4. `https://www.mygreatlearning.com/blog/viola-jones-algorithm`
5. Wikipedia, `https://en.wikipedia.org/wiki/Convolutional_neural_network`
6. `https://www.linkedin.com/feed/update/urn%3Ali%3Aactivity%3A6655711815361761280/`
7. `https://www.reddit.com/r/computervision/comments/ihstgi/using_facial_landmarks_for_overlaying_faces_with/`
8. `https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset`
9. `https://www.kaggle.com/jessicali9530`
10. "TensorFlow White Papers", TensorFlow, `https://www.tensorflow.org/about/bib`
11. "Keras documentation: About Keras", `https://keras.io`
12. "OpenCV", `https://opencv.org`
13. "Sckit-Learn", `https://scikit-learn.org`