ECE 8725

SUPERVISED LEARNING
Fall-2020

FINAL COURSE PROJECT

Face Mask Detection Using Deep Learning and Computer Vision

Rounak Singh
14348681
rsft6@umsystem.edu

Table of Contents

1. INTRODUCTION

Wearing a face mask has become the new normal. With the Covid-19 pandemic around, it has become mandatory to have face coverings at public places like offices, supermarkets, job and study campus areas and other places of public gatherings. People are not allowed to move outside without having a face mask on. They are not offered services and are not welcomed if they violate social distancing norms and not respect the need of the hour. For example, riding an uber, shopping at a departmental store, traveling through public transport, flights, airports, schools and many other places require face masks.

Fig: Face mark warnings at public places

To maintain public safety and ensure social discipline, organizations have now started to monitor people in order make sure they have a face covering. They use face-mask detectors to detect people without face-coverings. This project involves a similar approach to build a face-mask detector. It is implemented using Deep Learning and Computer Vision with their popular libraries such as TensorFlow, Keras, OpenCv and Scikit-Learn along with a data set containing images of people wearing masks and people not wearing masks. The developed model can detect faces of people from images, video files and live video and then correctly detect if a particular person is wearing a face mask or not.

Face-mask detection involves detecting the location of face then determining whether that person has a face-covering or not. This idea has its roots in object- detection i.e detecting certain objects from a number of classes. It is then implemented for this particular application.

2. METHODOLOGY

2.1 Deep Learning: It is a subfield of machine learning concerned with algorithms inspired by structure and function of the brain called 'Artificial Neural Networks'.

2.2 Computer Vision: It is a field of study thar seeks to develop techniques that help computers see and understand the content of digital data like images and videos. It is widely used for Image Processing purposes and Object Detection.

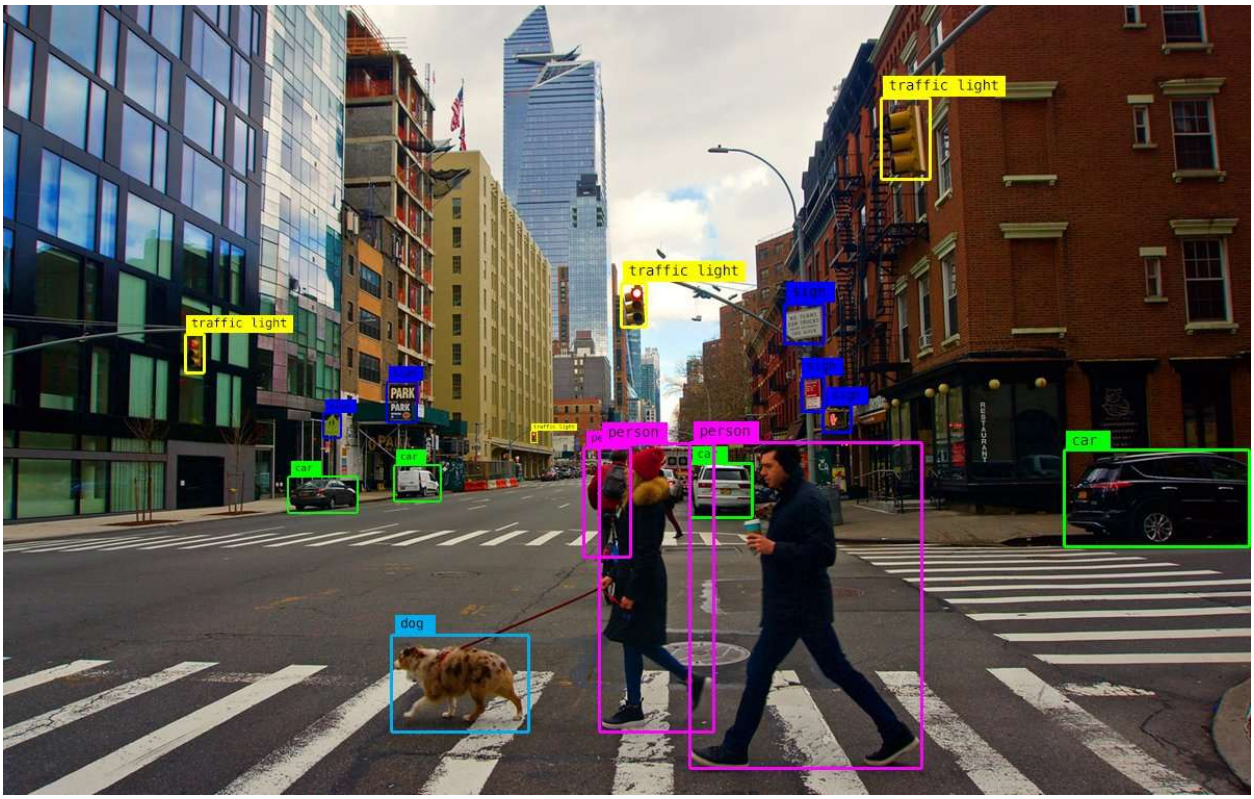

Fig: Object detection

2.3 Formulation of Project: In face detection method, a face is detected from an image using its several attributes. Given an individual image, the challenge is to identify the face from the picture. It is a tedious task to detect face from an image as faces are different in shape, sizes, complexion, feature, etc. Simply training any model over a set of images containing faces is a laborious task and consumes lot of time in training. So, while training, it is required that all the images must contain a single face and have the following- same size, color and orientation. This makes detection easier for the model as it trains on huge data of same kind. To achieve this, there are two stages involved. The following block diagram gives an overview of the two stages involved in the project.

The first stage of this project uses Deep Learning method which employs TensorFlow and Keras to build a Convoluted Neural Network (CNN) that trains on a Data set containing images, the classifier is build using Scikit learn which classifies the data into 2 classes namely, ‘with-mask’ and ‘without-mask’. Output of the Classifier is saved using new variables.

The second stage of the project uses Computer Vision which uses OpenCV library to detect faces using predefined facial attributes from Facial recognition library (haars-frontal face features) along with the classifier output obtained in stage one to detect mask on the face from a given image.

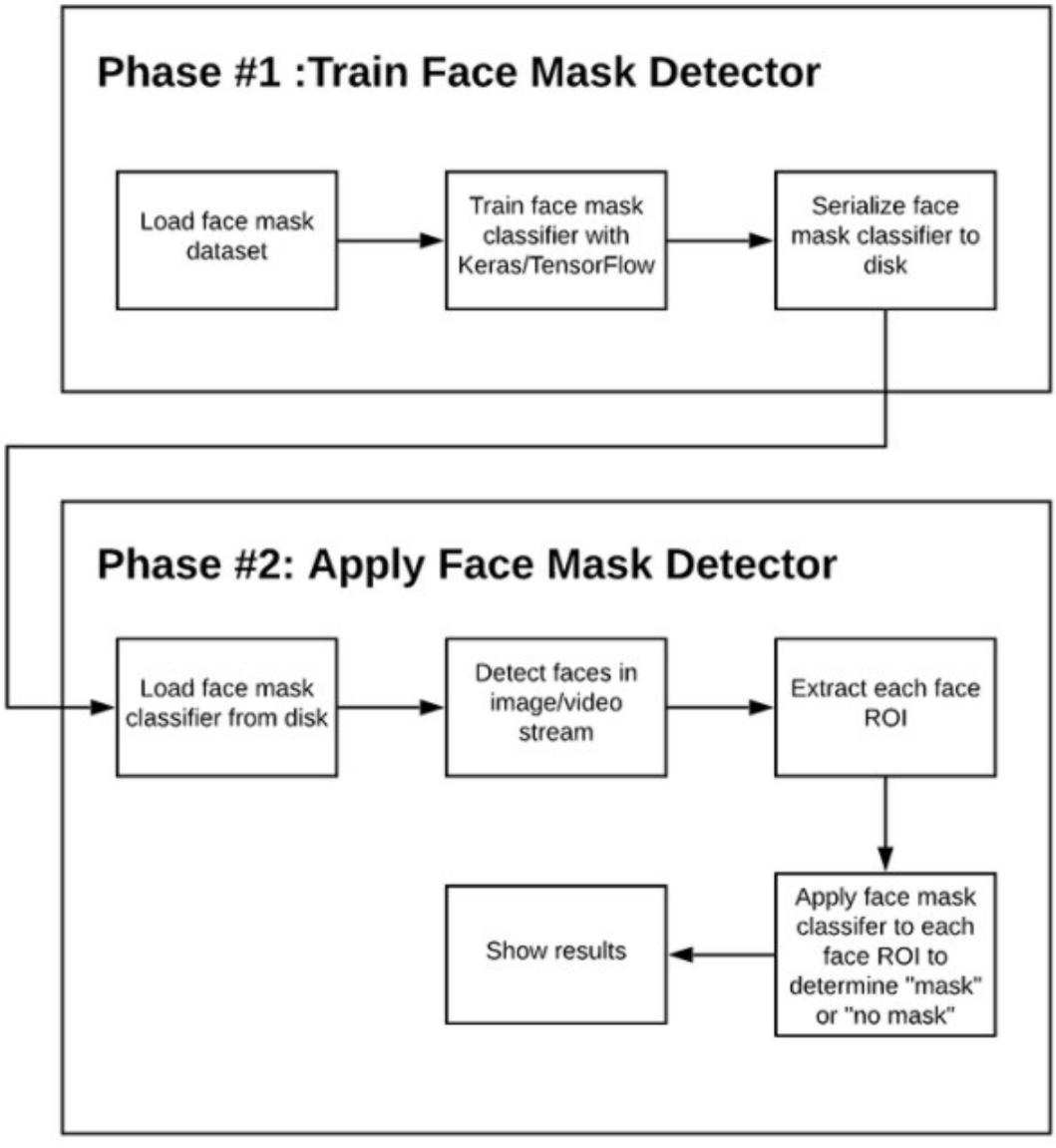

Fig: Block Diagram

3. LIBRARIES AND DATASET

3.1 TensorFlow: TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, etc. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing.

3.2 Keras: Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration speeds. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models. All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

3.3 OpenCV: OpenCV (Opensource Computer Vision Library), an opensource computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth. The proposed method makes use of these features of OpenCV in resizing and color conversion of data images.

3.4 Scikit-Learn: It is a free opensource python library for machine learning used to build algorithms involving regression, classification and clustering.

3.5 OS: It is a general-purpose library used to build miscellaneous Operating System interfaces. It provides a portable way of using operating system dependent functionalities.

3.6 Dataset: The Dataset contains 1376 images of in which 690 images are of people wearing masks and the rest 686 images are of people not wearing masks. It was created by Prajna Bhandari and is available at Github. A Sample of the dataset is shown below

Dataset sample:

Fig2: Dataset Samples

4. IMPLEMENTATION

4.1 Data Preprocessing: Data Preprocessing involves the conversion of the data from a given format to programmer friendly format, i.e cleaning of dataset constituents and making it ready for models.

4.2 Data Visualization: This means transforming random data values to understandable formats and then visualizing it in the form of text, graphs or images.

The total number of imaged are classified into two categories:

1. With Mask
2. Without Mask

The statement '*os.listdir(data_path)'* is used to fetch the directory where the dataset is saved. The labels are defined as follows:

'labels=[i for i in range(len(categories)]' . It sets the labels as [0, 1].

This gives us the number of labels and each category is mapped to its respective label using '*label_dict = dict(zip(categories.labels))'* which returns an iteration of tuples in the form of a zip object where all the objects passed in the iterations are paired together. i.e images and their respective images and label. The mapped label looks like *{with mask:0 and without mask:1}*

4.3 Converting the colored images to grayscale

Most of the detection systems and pattern recognition systems work on grayscale images. Colored images have extra information of the colors involved and a lot of such images create a non-essential data. This could increase the training time and size of the training data required to achieve a given objective. Hence it become necessary to remove this non-essential information.

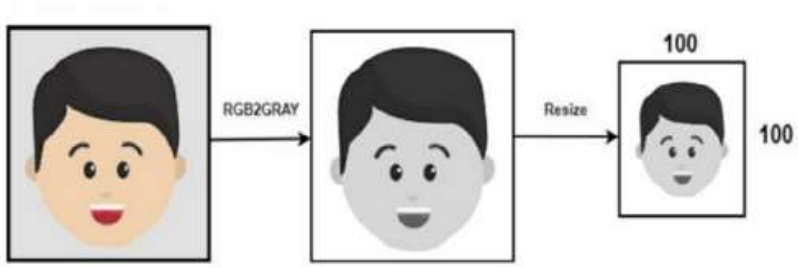

Fig3: Color to Gray scale conversion

Using the OpenCV library, I used the function '*cv2.Color(input_image,flag)'* to convert all the colored images into grayscale images. [specific commad used: gray = cv2.Color(img, COLOR_BGR2GRAY)].

4.4 Size Conversion:

Deep Convoluted Neural Networks require a fixed-size input images. The images from the dataset were resized to 120x120 using cv2.resize().

The images are also required to be a tensor (3D) where each channel has a prominent unique pixel. This is true for the all the images in the dataset. Also, most CNNs only accept fine-tuned images which creates a number of problems from data collection to the final implementation of the model. These issues can be avoided by tuning the images into a tensor of required dimensions. The images are reconfigured as following:

The images are normalized to converge the pixel range between 0 and 1. Later, these are converted to 4D arrays using '*data = np.reshape(data,(data_shape[0], img_size,img_size,1)'* , where 1 at the end represents a gray scale image and 0 in its place would represent a colored image. The Output layer of the Neural Network has 2 outputs:

1. Output representing images of 'With Mask Type'
2. Output representing images of 'Without Mask Type'

Which is nothing but a categorical type of representation having labels.

4.5 Building the CNN:

The following are the specification of the Artificial Neural Network used in the project.

1. Type: Sequential Model
2. The Artificial Neural Network used here: Sequential model.
3. The input layer: First CNN-200
4. Activation function: Rectified Linear unit
5. Dropout layer: 50%
6. Max Pool layers: 200
7. Mask (Kernel) size: 3x3
8. Padding: allowed
9. Second CNN-100
10. Flatten Layer
11. First Dense layer neurons = 64
12. Second Dense layer (Output) Neurons = 2
13. Labels-2

4.6 Compiling:

Optimizer: 'adam'
Loss function: 'categorical_crossentropy'
Metrics: 'accuracy'

The Neural Network is built using *Keras* library. The first convolution layer learns using 200 layers. Max Pooling helps in spatial dimension reduction. The second CNN has size 100 and the input to it is given by flattening the output of the first layer. (The matrix form is converted into a vector). The figure below gives an idea of how the Neural Network model looks.

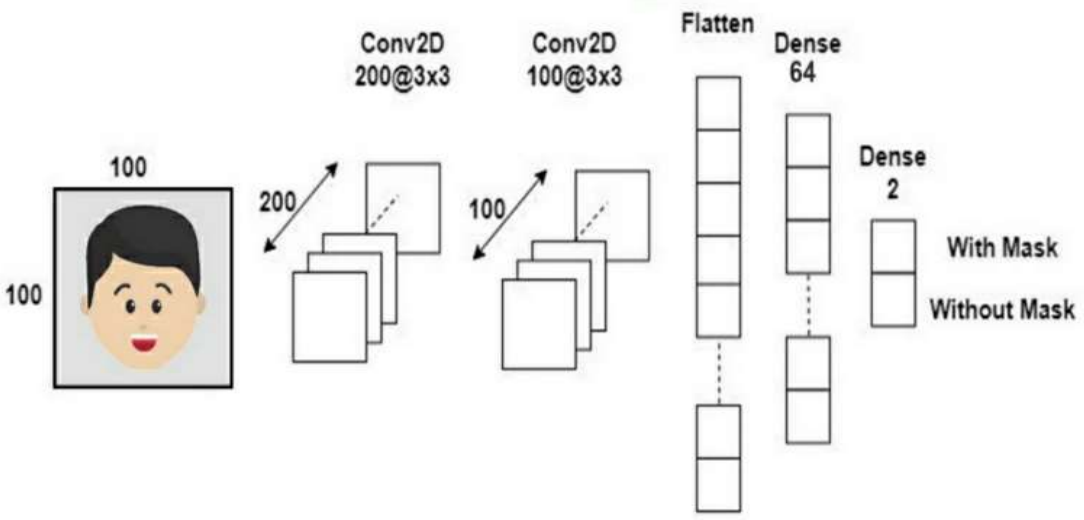

Fig4: Representation of Neural Network

4.7 Training and Testing: From Scikit-learn library, '*train_test_split*' is used to split the data for training and testing. And the model trains on the reconfigured images from the data set. The Testing is done on new images, video files and live video using webcam. The model is trained for 15 iterations.

5. RESULTS

5.1 Accuracy:

```
: print(model.evaluate(test_data,test_target))

138/138 [==============================] - 5s 35ms/step
[0.17557063561094843, 0.9420289993286133]
```

Fig5: Accuracy of the model

5.2 Training Loss vs. Validation Loss:

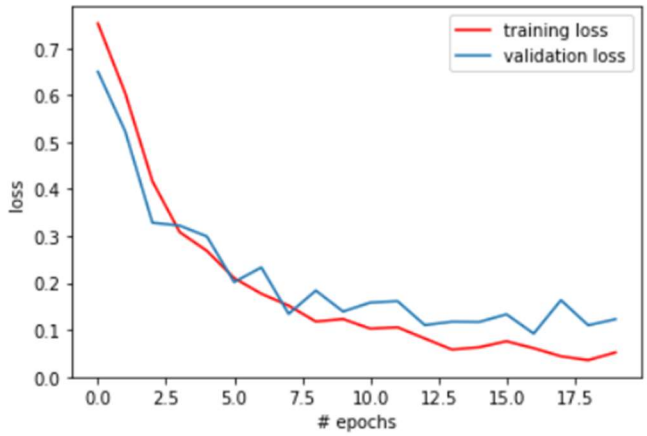

Fig6: Training vs. Validation Loss

5.3 Training accuracy vs Validation accuracy:

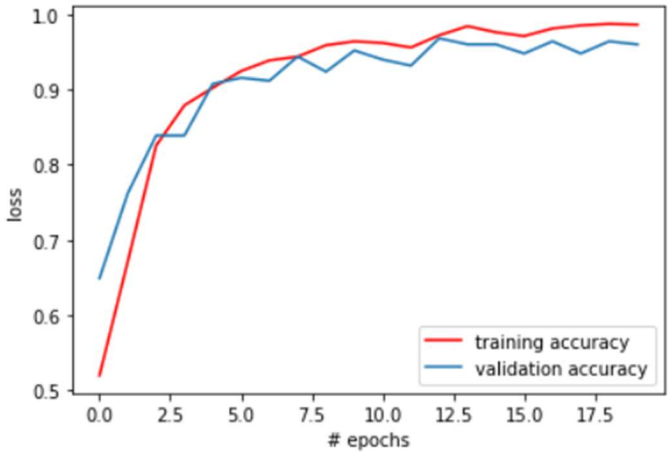

Fig7: Training vs. Validation Accuracy

5.4 Output:

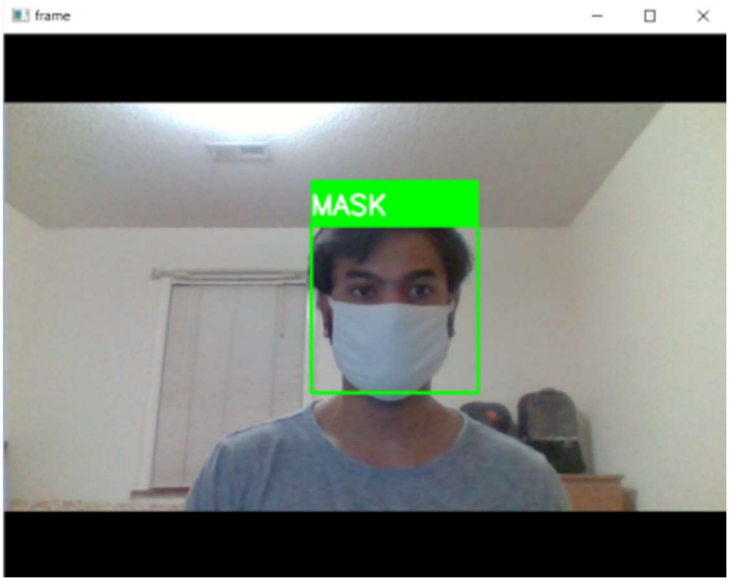

Fig: With Mask

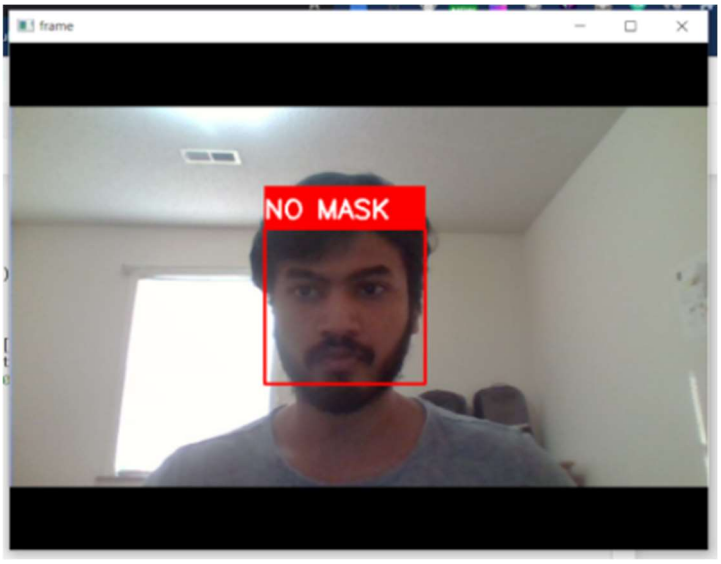

Fig: Without mask

6. REFERENCES

[1] Lippert, Christoph, and Benjamin Bergner. "Face Mask Detector."

[2] Das, Arjya, Mohammad Wasif Ansari, and Rohini Basak. "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV.

Appendix:

Dataset and code: https://github.com/singh-rounak/Face-Mask-Detector