PROJECT REPORT

on

Real Time Face-mask Detection

(CSE V Semester Mini project PCS-604)

2021-2022

Submitted to:

Mr. B.P Dubey

(CC-CSE-D-VI-Sem)

Submitted by:

Mr. Shubham

Roll. No.: 1018604

CSE-A-VI-Sem

Session: 2021-2022

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNVERSITY, DEHRADUN

CERTIFICATE

Certified that Mr. Shubham Rana (Roll No.- 1018604) has developed mini project on "Real time Face Mask Detection" for the CS V Semester Mini Project Lab (PCS-604) in Graphic Era Hill University, Dehradun. The project carried out by Students is their own work as best of my knowledge.

Date:14/05/2021

(Mr. B.P Dubey)

Project Co-ordinator

CC-CSE-B-V-Sem

(CSE Department)

GEHU Dehradun

(Dr Mahesh manchanda)

Project Guide

Resource Person

(CSE Department)

ACKNOWLEDGMENT

We would like to express our gratitude to The Almighty Shiva Baba, the most Beneficent and the most Merciful, for completion of project.

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our project Co-ordinator Mr. B.P Yadav and our Project Guide Dr. Mahesh Manchanda for his patience, support, and encouragement throughout the completion of this project and having faith in us.

At last but not the least We greatly indebted to all other persons who directly or indirectly helped us during this work.

Mr. Shubham rana

Roll No.- 1018604

CSE-A-VI-Sem

Session: 2021-2022

GEHU, Dehradun

TABLE OF CONTENTS

LIST OF TABLES

TABLE **Page No.**

LIST OF FIGURES

FIGURE **Page No.**

CHAPTER 1

INTRODUCTION

1.1 ABOUT PROJECT

The year 2020 has shown mankind some mind-boggling series of events amongst which the COVID-19 pandemic is the most life-changing event which has startled the world since the year began. Affecting the health and lives of masses, COVID-19 has called for strict measures to be followed in order to prevent the spread of disease. From the very basic hygiene standards to the treatments in the hospitals, people are doing all they can for their own and the society's safety; face masks are one of the personal protective equipment. People wear face masks once they step out of their homes and authorities strictly ensure that people are wearing face masks while they are in groups and public places. To monitor that people are following this basic safety principle, a strategy should be developed. A face mask detector system can be implemented to check this. Face mask detection means to identify whether a person is wearing a mask or not. The first step to recognize the presence of a mask on the face is to detect the face, which makes the strategy divided into two parts: to detect faces and to detect masks on those faces. Face detection is one of the applications of object detection and can be used in many areas like security, biometrics, law enforcement and more. There are many detec tor systems developed around the world and being implemented. However, all this science needs optimization; a better, more precise detector, because the world cannot afford any more increase in corona cases. 1In this project, we will be developing a face mask detector that is able to distinguish between faces with masks and faces with no masks. In this report, we have proposed a detector which employs SSD for face detection and a neural network to detect presence of a face mask. The implementation of the algorithm is on images,

videos and live video streams. The rest of the report is organized as follows. In Section 2, we will go through the literature review and related work.

1.2 ANN

Artificial neural networks (**ANNs**) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve their ability) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express with a traditional computer algorithm using rule-based programming.

An ANN is based on a collection of connected units called artificial neurons, (analogous to biological neurons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream.

Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as backpropagation, or passing information in the reverse direction and adjusting the network to reflect that information.

1.2.1 DNN

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions. These components functioning similar to the human brains and can be trained like any other ML algorithm.

For example, a DNN that is trained to recognize dog breeds will go over the given image and calculate the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network.

Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets.

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network did not accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

Recurrent neural networks (RNNs), in which data can flow in any direction, are used for applications such as language modeling. Long short-term memory is particularly effective for this use.

Convolutional deep neural networks (CNNs) are used in computer vision. CNNs also have been applied to acoustic modeling for automatic speech recognition (ASR).

CHAPTER 2

PROJECT

2.1 Overview

Face mask detection had seen significant progress in the domains of Image processing and Computer vision, since the rise of the Covid-19 pandemic. Many face detection models have been created using several algorithms and techniques. The proposed approach in this paper uses deep learning, TensorFlow, Keras, and OpenCV to detect face masks. This model can be used for safety purposes since it is very resource efficient to deploy. The SSDMNV2 approach uses Single Shot Multibox Detector as a face detector and MobilenetV2 architecture as a framework for the classifier, which is very lightweight and can even be used in embedded devices (like NVIDIA Jetson Nano, Raspberry pi) to perform real-time mask detection. The technique deployed in this paper gives us an accuracy score of 0.9264 and an F1 score of 0.93. The dataset provided in this paper, was collected from various sources, can be used by other researchers for further advanced models such as those of face recognition, facial landmarks, and facial part detection process.

2.2 Methodology

2.2.1 Dataset

The dataset which we have used consists of 3835 total images out of which 1916 are of masked faces and 1919 are of unmasked faces. All the images are actual images extracted from Bing Search API, Kaggle datasets and RMFD dataset. From all the three sources, the proportion of the images is equal. The images cover diverse races i.e Asian, Caucasian etc. The proportion of masked to unmasked faces determine that the dataset is balanced. We need to split our dataset into three parts: training dataset, test dataset and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy. We need a model that performs well on a dataset that it has never seen (test data), which is called generalization. The training set is the actual subset of the dataset that we use to train the model. The model observes and learns from this data and then optimizes its parameters. The validation dataset is used to select hyperparameters (learning rate, regularization parameters). When the model is performing well enough on our validation dataset, we can stop learning using a training dataset. The test set is the remaining subset of data used to provide an

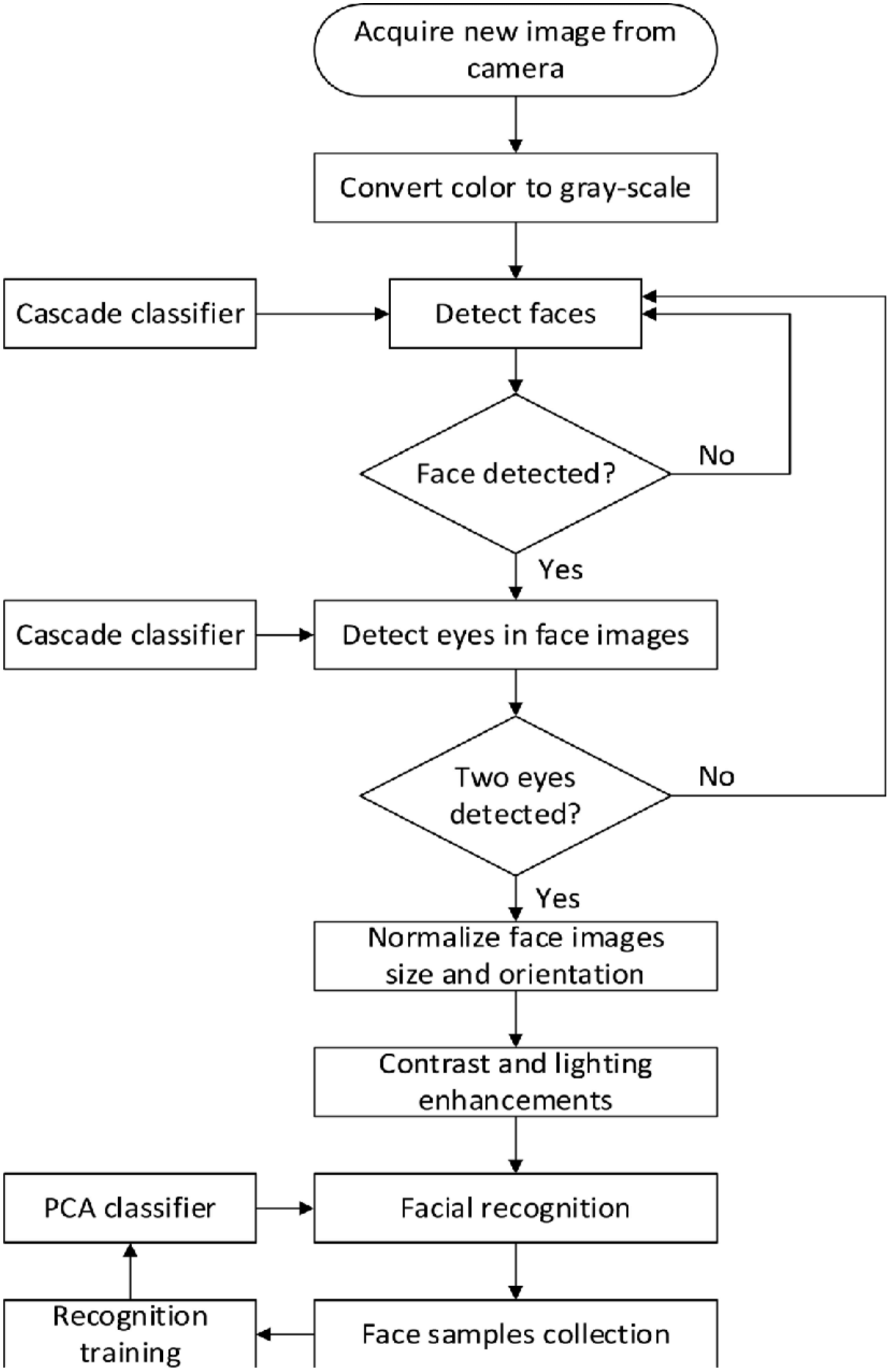
unbiased evaluation of a final model fit on the training dataset. Data is split as per a split ratio which is highly dependent on the type of model we are building and the dataset itself. If our dataset and model are such that a lot of training is required, then we use a larger chunk of the data just for training which is our case. If the model has a lot of hyperparameters that can be tuned, then we need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so we can take a smaller validation dataset. In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20% as the testing data, which makes the split ratio as 0.8:0.2 of train to test set. Out of the training data, we have used 20% as a validation data set. Overall, 64% of the dataset is used for training, 16% for validation and 20% for testing.

2.2 Architecture

The working of the Single Shot Detector algorithm relies on an input image with a specified bound ing box against the objects. The methodology of predicting an object in an image depends upon very renowned convolution fashion. For each pixel of a given image, a set of default bounding boxes (usually 4) with different sizes and aspect ratios are evaluated. Moreover, for all the pixels, a confidence score for all possible objects are calculated with an additional label of 'No Object'. This calculation is repeated for many different feature maps. In order to extract feature maps, we usually use the predefined trained techniques which are used for high quality classification problems. We call this part of the model a base model. For the SSD, we have VGG-16 network as our base model. At the training time, the bounding boxes evaluated are compared with the ground truth boxes and in the back propagation, the trainable parameters are altered as per requirement. We truncate the VGG-16 model just before the classification layer and add feature layers which keep on decreasing in size. At each feature space, we use a kernel to produce outcomes which depicts corresponding scores for each pixel whether there exists any object or not and the corresponding dimensions of the resulting bounding box.

VGG-16 is a very dense network having 16 layers of convolution which are useful in extracting features to classify and detect objects. The reason for the selection is because the architecture 4consists of stacks of convolutions with 3x3 kernel size which thoroughly extract numerous feature information along with max-pooling and ReLU to pass the information flow in the model and adding non linearity respectively from the given image. For additional nonlinearity, it uses 1x1 convolution blocks which does not change the spatial dimension of the input. Due to the small size filters striding over the image, there are many weight parameters which end up giving an improved performance

2.3 FlowChart


Acquire new image from camera
Convert color to gray-scale
Cascade classifier
Detect faces
Face detected?
No
Yes
Cascade classifier
Detect eyes in face images
Two eyes detected?
No
Yes
Normalize face images size and orientation
Contrast and lighting enhancements
PCA classifier
Facial recognition
Recognition training
Face samples collection

2.4 Software requirements:

Operating System	Any OS with clients to access the internet
Network	Wi-Fi Internet or cellular Network
IDE	JupyterLab
Programming language	Python

2.5 Hardware Requirements:

For application development, the following Software Requirements are:

Processor: Intel or high

RAM: 1024 MB

Space on disk: minimum 100mb

For running the application:

Device: Any device that can access the internet

Minimum space to execute: 20 MB

CHAPTER 3

SNAPSHOT OF PROJECT

3.1 Snaps

Importing libraries

```
[1]: import matplotlib.pyplot as plt
     import cv2
     # Technically not necessary in newest versions of jupyter
     %matplotlib inline
```

```
[2]: from keras.optimizers import RMSprop
     from keras.preprocessing.image import ImageDataGenerator
     import cv2
     from keras.models import Sequential
     from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
     from keras.models import Model, load_model
     from keras.callbacks import TensorBoard, ModelCheckpoint
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import f1_score
     from sklearn.utils import shuffle
     import numpy as np
```

```
Using TensorFlow backend.
```

Visulazing the Data

```
[3]: img1 = cv2.imread('./train/with_mask/-110603108-gettyimages-533567012.jpg')
```

```
[4]: plt.imshow(img1)
```

```
[4]: <matplotlib.image.AxesImage at 0x2527737ef98>
```

Creating the Model ¶

[8]:
```
model = Sequential([
    Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2,2),

    Conv2D(100, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dropout(0.5),
    Dense(50, activation='relu'),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

Training the Model

:
```
checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
```

:
```
history = model.fit_generator(train_generator,
                              epochs=10,
                              validation_data=validation_generator,
                              callbacks=[checkpoint])
```

:
```
model.save('wieghts.h5')
```

:
```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("./wieghts.h5")

results={0:' mask detected',1:' without mask'}
GR_dict={1:(0,0,255),0:(0,255,0)}

rect_size = 4
cap = cv2.VideoCapture(0)
height =int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

CHAPTER 4

CONCLUSION

4.1 CONCLUSION

To mitigate the spread of COVID-19 pandemic, measures must be taken. We have modeled a face mask detector using SSD architecture and transfer learning methods in neural networks. To train, validate and test the model, we used the dataset that consisted of 1916 masked faces images and 1919 unmasked faces images. These images were taken from various resources like Kaggle and RMFD datasets. The model was inferred on images and live video streams. To select a base model, we evaluated the metrics like accuracy, precision and recall and selected MobileNetV2 architecture with the best performance having 100% precision and 99% recall. It is also computationally efficient using MobileNetV2 which makes it easier to install the model to embedded systems. This face mask detector can be deployed in many areas like shopping malls, airports and other heavy traffic places to monitor the public and to avoid the spread of the disease by checking who is following basic rules and who is not.

REFERENCE

1. A, A. S., & Naik, C. (2016). Different Data Mining Approaches for Predicting Heart Disease, 277– 281. https://doi.org/10.15680/IJIRSET.2016.0505545
2. Kirmani, M. (2017). Cardiovascular Disease Prediction using Data Mining Techniques. Oriental Journal of Computer Science and Technology, 10(2), 520–528. https://doi.org/10.13005/ojcst/10.02.38
3. Brownlee, J. (2016). Naive Bayes for Machine Learning. Retrieved March 4, 2019, from https://machinelearningmastery.com/naive-bayes-for-machine-learning/