CHAPTER I

1.INTRODUCTION

1.1 GENERAL

The world has not yet fully Recover from this pandemic and the vaccine that can effectively treat Covid-19 is yet to be discovered. However, to reduce the impact of the pandemic on the country's economy, several governments have allowed a limited number of economic activities to be resumed once the number of new cases of Covid19 has dropped below a certain level. As these countries cautiously restarting their economic activities, concerns have emerged regarding workplace safety in the new post-Covid-19 environment. To reduce the possibility of infection, it is advised that people should wear masks and maintain a distance of at least 1 meter from each other. Deep learning has gained more attention in object detection and was used for human detection purposes and develop a face mask detection tool that can detect whether the individual is wearing mask or not. This can be done by evaluation of the classification results by analyzing real-time streaming from the Camera. In deep learning projects, we need a training data set. It is the actual dataset used to train the model for performing various actions.

1.1 Motivation of Work:

To reduce the possibility of infection, it is advised that people should wear masks and maintain a distance of at least 1 meter from each other. Deep learning has gained more attention in object detection and was used for human detection purposes and develop a face mask detection tool that can detect whether the individual is wearing mask or not. This can be done by evaluation of the classification results by analyzing real-time streaming from the Camera. In deep learning projects, we need a training data set. It is the actual dataset used to train the model for performing various actions

1.2 Objectives and scope of the projects

Global pandemic COVID-19 circumstances emerged in an epidemic of dangerous disease in all over the world. Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs. Using Face Mask Detection System, one can monitor if the people are wearing masks or not. Here HAAR-CASACADE algorithm is used for image detection. Collating with other existing algorithms, this classifier produces a high recognition rate even with varying expressions, efficient feature selection and low assortment of false positive features. HAAR feature-based cascade classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 85-95%. According to this motivation we demand mask detection as a unique and public health service system during the global pandemic COVID-19 epidemic. The model is trained by face mask image and non-face mask image

1.3 PROBLEM STATEMENT

The main objective of the face detection model is to detect the face of individuals and conclude whether they are wearing masks or not at that particular moment when they are captured in the image

1.3.1 EXISTING SYSTEM

face detection problem has been approached using Multi-Task Cascaded Convolutional Neural Network (MTCNN). Then facial features extraction is performed using the Google Face Net embedding model. 1.This system is capable to train the dataset of both persons wearing masks andwithout wearing masks. After training the model the system can predicting whether the person is wearing the mask or not wearing mask

1.3.1.1 DISADVANTAGE OF EXISTING SYSTEM

- It is not recognize the mask properly
- It is the camera is low
- It is only store the data in local space

1.3.2 PROPOSED SYSTEM

1.This system is capable to train the dataset of both persons wearing masks and without wearing masks.

2.After training the model the system can predicting whether the person is wearing the mask or not .

3.It also can access the webcam and predict the result.

CHAPTER 2

LITERATURE SURVEY

1.TITLE: Face Mask Detection Using Deep Learning

AUTHOR: Anith at el.

YEAR:2020

DESCRIPTION: In order to prevent the spread of CORONA virus, everyone must wear a mask during the pandemic. In these tough times of COVID-19 it is necessary to build a model that detects people with and without mask in real-time as it works as a simple precautionary measure to prevent the spread of virus. If deployed correctly, this machine learning technique helps in simplifying the work of frontline warriors and saving their lives. A basic Convolutional Neural Network (CNN) model is built using TensorFlow, Keras, Scikit-learn and OpenCV to make the algorithm as accurate as possible. Javascript API helps in accessing webcam for real-time face mask detection. Since Google Colab runs on web browser it can't access local hardware like a camera without APIs. The proposed work contains three stages: (i) pre-processing, (ii) Training a CNN and (iii) Real-time classification. The first part is the Pre-processing section, which can be divided into "Grayscale Conversion" of RGB image, "image resizing and normalization" to avoid false predictions. Then the proposed CNN, classifies faces with and without masks as the output layer of proposed CNN architecture contains two neurons with Softmax activation to classify the same. Categorical cross-entropy is employed as loss function. The proposed model has Validation accuracy of 96%. If anyone in the video stream is not wearing a protective mask a Red coloured rectangle is drawn around the face with a dialog entitled as NO MASK and a Green coloured rectangle is drawn around the face of a person wearing MASK.

2.TITLE: Face Mask Detection using Convolutional Neural Network

AUTHOR: Preeti Tuli , Priyanka Sahu.

YEAR:2019

DESCRIPTION: During pandemic CoVID 19, people must use face masks in public areas to prevent and reducing the risk of transmission and spread of the virus. Computer Vision can help to monitor the use of face masks based on images captured via CCTV. Several public areas have installed CCTV that can monitor using masks, but too many people in the area would create problems. Face and side masked face detection is a challenge, given the removal of facial features such as the mouth and nose. A previous study built a mask detection system using Convolutional Neural Networks (CNN) based models, which produced high accuracy but was limited to the front face. This research proposed the CNN method to detect masks based on facial images taken from cameras in public areas. Images containing faces from CCTV are segmented, each faces first using the Retina Face. Experiments were carried out on a single face image in mask detection, resulting in an accuracy of 97.33%. These excellent results are not surprising given CNN's ability to recognize patterns. The most important thing is the segmentation of the face region from one image, which is then tested to produce an accuracy of 82.46%. We selected the best configuration from the two experiments, combined into a mask detection from an image containing multiple faces. The results also showed a significant effect between the face detection method and the learning rate value on the accuracy of the mask use detection system, with the best results of 79.45% using the RetinaFace face detection model.

3.TITLE: Control The COVID-19 Pandemic: Face Mask Detection Using Transfer Learning

AUTHOR: Kavya .C , Suganya.R

YEAR:2021

DESCRIPTION: Currently, in the face of the health crisis caused by the Coronavirus COVID-19 which has spread throughout the worldwide. The fight against this pandemic has become an unavoidable reality for many countries. It is now a matter involving many areas of research in the use of new information technologies, particularly those related to artificial intelligence. In this paper, we present a novel contribution to help in the fight against this pandemic. It concerns the detection of people wearing masks because they cannot work or move around as usual without protection against COVID-19. However, there are only a few research studies about face mask detection. In this work, we investigated using different deep Convolutional Neural Networks (CNN) to extract deep features from images of faces. The extracted features are further processed using various machine learning classifiers such as Support Vector Machine (SVM) and K-Nearest Neighbors (K-NN). Were used and examined all different metrics such as accuracy and precision, to compare all model performances. The best classification rate was getting is 97.1%, which was achieved by combining SVM and the MobileNetV2 model. Despite the small dataset used (1376 images), we have obtained very satisfactory results for the detection of masks on the faces.

4.TITEL: Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV

AUTHOR: Arjya Das , Mohammad Wasif Ansari , Rohini Basak

YEAR:2020

DESCRIPTION: COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task 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. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

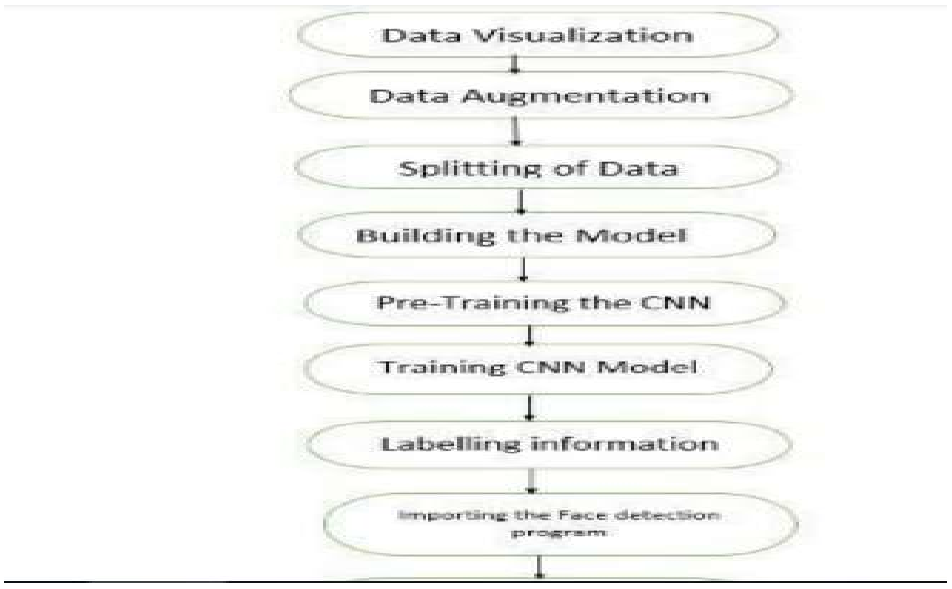
CHAPTER 3
PROJECT DESCRIPTION

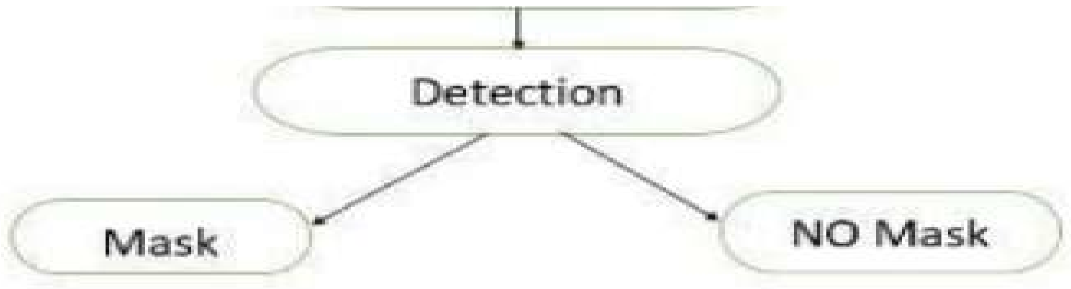
3.1 INTRODUCTION

The outbreak of COVID-19 has taught everyone the importance of face masks in their lives. SARS-COV-2(Severe Acute Respiratory Syndrome) is a communicable virus that is transmitted from a person while speaking, sneezing in the form of respiratory droplets. It spreads by touching an infected surface or by being in contact with an infected person. Healthcare officials from the World Health Organization and local authorities are propelling people to wear face masks as it is one of the comprehensive strategies to overcome the transmission. Amid the advancement of technology, deep learning and computer vision have proved to be an effective way in recognition through image processing. This system is a real-time application to detect people if they are wearing a mask or are without a mask. It has been trained with the dataset that contains around 4000 images using 224x224 as width and height of the image and have achieved an accuracy rate of 98%. In this research, this model has been trained and compiled with 2 CNN for differentiating accuracy to choose the best for this type of model.It can be put into action in public areas such as airports, railways, schools, offices, etc. to check if COVID-19 guidelines are being adhered to or not.

3.2MODULE DIAGRAM

3.3.1 DETAILED ARCHITECTURE





MODULES USED

TENSORFLOW FRAMEWORK - Tensor flow is an open-source software library. Tensor flow was originally developed by researchers and engineers. It is working on the Google Brain Team within Google's Machine Intelligence research organization the purposes of conducting machine learning and deep neural networks research. It is an opensource framework to run deep learning and other statistical and predictive analytics workloads. It is a python library that supports many classification and regression algorithms and more generally deep learning. TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. 5 Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors

OPENCV –

1.It is a cross-platform library using which we can develop real-time computer vision applications.

2.It mainly focuses on image processing, video capture and analysis including feature like face detection and object detection.

3. Currently Open CV supports a wide variety of programming languages like C++, Python, Java etc. and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.

4. Also, interfaces based on CUDA and OpenCL are also under active

development for high-speed GPU operations. Open CV-Python is the Python API of Open CV.

5. It combines the best qualities of Open CV C++ API and Python language.

6. OpenCV (Open-Source Computer Vision Library) is an opensource computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

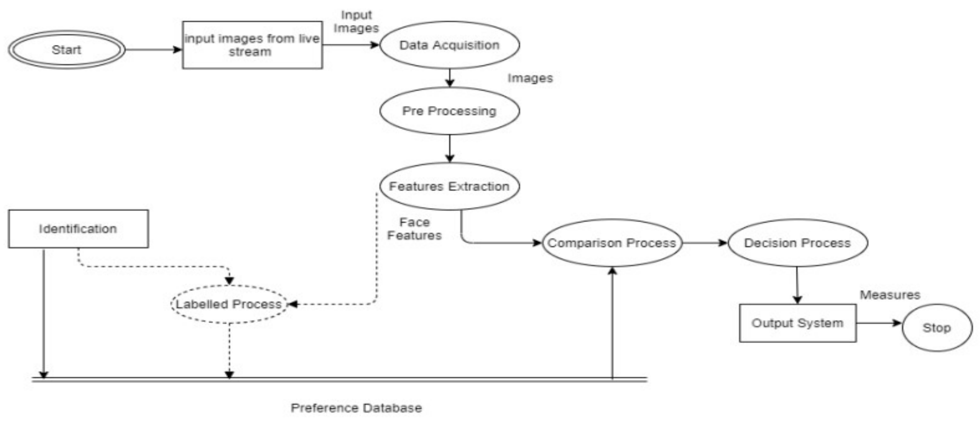
7. The library has more than 2500optimized algorithms, which includes a comprehensive set of both classic and state-of -the-art computer vision and machine learning algorithms.

8.Algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

NUMPY- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is opensource software and has many contributors. The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier. An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric also variously called Numerical Python extensions or NumPy Hugunin, a graduate student at Massachusetts Institute of Technology (MIT) joined the Corporation for National Research Initiatives (CNRI) to work on J Python in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer. In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported num-array's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy.

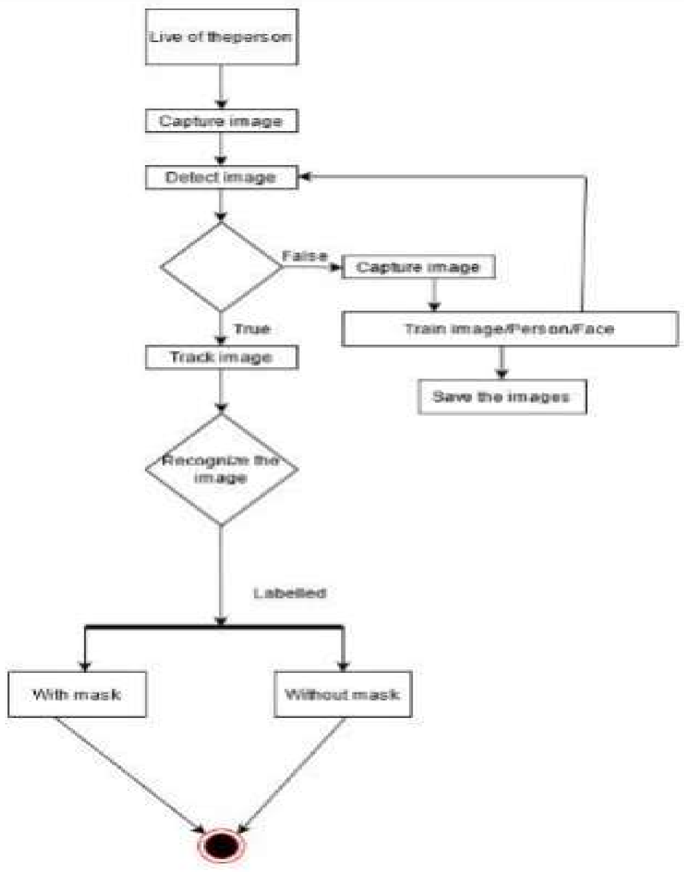
3.2.1 USER DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays. Data Flow Data is a key and basic tool for the architecture of all other objects. Bubble-bubble or data flow graph is another name for DFD. DFDs are a model of the proposed system. They should indicate the requirements on which the new system should be built in a clear and direct manner. This is used as a basis for building chart structure plans over time during the design process. The following is the Basic Notation for building DFDs:



3.3.2 ACTIVITY DIAGRAM

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

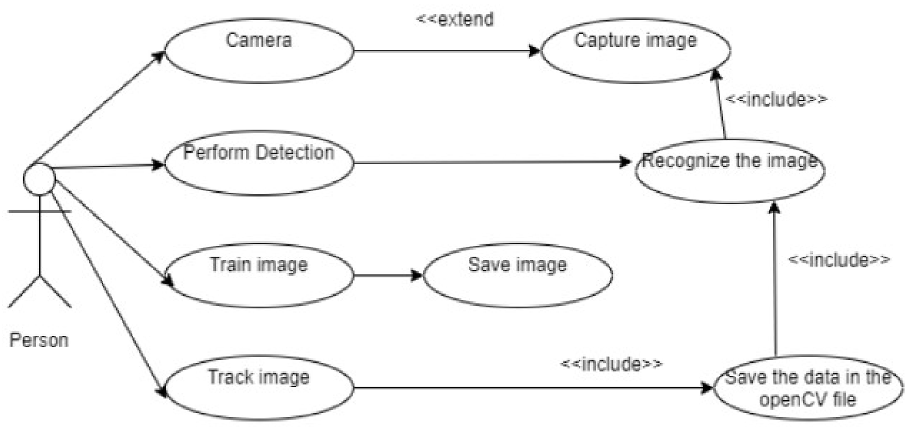


3.3.3 USECASE DIAGRAM

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

● Scenarios in which your system or application interacts with people, organizations, or external systems.

● Goals that your system or application helps those entities (known as actors) achieve.The scope of your system



4.SOFTWARE SPECIFICATION

4.1 SYSTEM REQUIFICATION

4.2 HARDWARE REQUIFICATION

The hardware requirements may provide as the basis for a contract for the implementation of the system and should as a result and be a complete and consistent specification of the whole system. They are used by software engineers as the initial point for the system design. It must what the method do and not how it should be implemented

• Processor	: Intel I5
• RAM	: 4GB
• Hard disk	: 40 GB

• Mouse	: logistech.
• Keyboard	: Dell

SOFTWARE REQUIREMENT

The software requirements document is the requirement of the system.it should include both a description and a specification of requirements. It is a set of what the system should do slightly than how it should do it. The software requirements give a basis for creating the software requirements specification. It is useful in estimating cost, planning group activities, performing tasks and tracking the teams and tracking the teams development throughout the development activity.

• Python Ide	: pycharm , vs code
• Coding language	: Python
• Technology	: Advanced programming using Python

3.2.3 LANGUAGE SPECIFICATION

PYTHON

- Python is a powerful multi-purpose programming language created byGuido van Rossum.
- It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

Features Of Python

1. Easy to code: Python is high level programming language. Python is very easy to learn language as compared to other language like c, c#, java script, java etc. It is very easy to code in python language and anybody can learn python basic in few hours or days. It is also developer-friendly language.

2. Free and Open Source: Python language is freely available at official website and you can download it from the given download link below click on the Download Python keyword. Since, it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

3. Object-Oriented Language: One of the key features of python is Object- Oriented programming. Python supports object oriented language and concepts of classes, objects encapsulation etc.

4. GUI Programming Support: Graphical Users interfaces can be made using a module such as PyQt5,PyQt4,wxPython or Tk in python.PyQt5 is the most popular option for creating graphical apps with Python.

5. High-Level Language: Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nordo we need to manage the memory.

6. Extensible feature: Python is a Extensible language. we can write our some python code into c or c++ language and also we can compile that

code in c/c++ language.

7. Python is Portable language: Python language is also a portable language. for example, if we have python code for windows and if we want to run this code on other platform such as Linux, Unix and Mac then we do not need to changeit, we can run this code on any platform.

8. Python is Integrated language: Python is also an Integrated language because we can easily integrated python with other language like C, C++ etc.

9. Interpreted Language: Python is an Interpreted Language. because python code is executed line by line at a time. like other language C, C++, java etc., there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.

10. Large Standard Library Python has a large standard library which provides rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers etc.

11. Dynamically Typed Language: Python is dynamically-typed language. That means the type (for example- int, double, long etc.,) for a variable is decided at run time not in advance. because of this feature we don’t need to specify the type of variable. Python is an Interpreted Language. because python code is executed line by line at a time. like other language C, C++, java etc., there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called byte code.

Python is a powerful multi-purpose programming language created by Guido van Rossum. It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time. Python features are

- Easy to code
- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Python is Portable language
- Python is Integrated language
- Interpreted
- Large Standard Library

CHAPTER 5

IMPLEMENTATION

GENERAL

Python is a program that was originally designed to simplify the implementation of numerical linear algebra routines. It has since grown into something much bigger, and it is used to implement numerical algorithms for a wide range of applications. The basic language used is very similar to standard linear algebra notation, but there are a few extensions that will likely cause you some problems at first.

5.1. CODE IMPLEMENTATION

```
#Import necessary packages
import cv2
import uuid
import mediapipe as mp
# Define mediapipe Face detector
face_detection = mp.solutions.face_detection.FaceDetection()
# Detection function for Face Mask Detection
def get_detection(frame):

height, width, channel = frame.shape

  # Convert frame BGR to RGB colorspace
  imgRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
  # Detect results from the frame
  result = face_detection.process(imgRGB)

  # Extract data from result
  try:
      for count, detection in enumerate(result.detections):
          # print(detection)
     # Extract bounding box information
```

```
        box = detection.location_data.relative_bounding_box
        x, y, w, h = int(box.xmin*width), int(box.ymin * height),
   int(box.width*width), int(box.height*height)

  # If detection is not available then pass
  except:
      pass

  return x, y, w, h
count = 0
class_path = 'mask'
while True:
  _, frame = cap.read()
  img = frame.copy()
  try:
     # Make detection
     x, y, w, h = get_detection(frame)
     cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)

     # Crop only the face part from the frame
     crop_img = img[y:y+h, x:x+w]
     filename = "Data/"+class_path+"/"+str(uuid.uuid4())+".jpg"

     # Save the image to the destination path
     cv2.imwrite(filename, crop_img)
     cv2.imshow("frame", crop_img)
     count+=1
  except:
      pass
  if cv2.waitKey(1) == ord('q') or count>=500:
      break

cap.release()
cv2.destroyAllWindows()
```

train.py

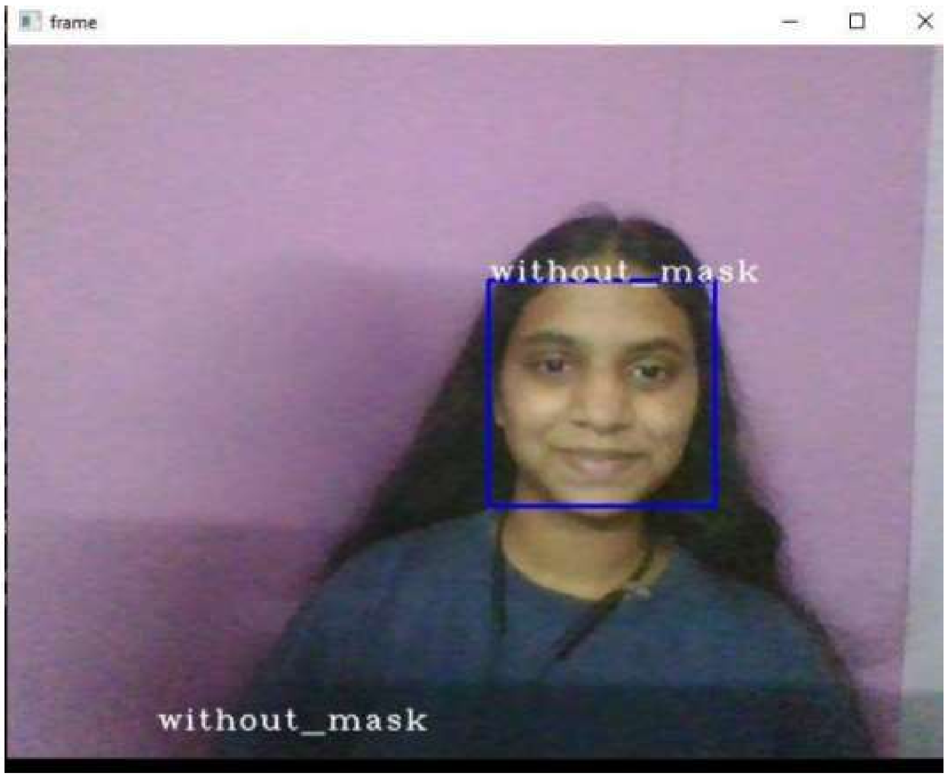
```
# import necessary packages
import os
import cv2
import time
from tqdm import tqdm
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, Dense, MaxPooling2D, Flatten,
Activation, Dropout
img_size = 100
datadir = r'Data'  # root data directory
CATEGORIES = os.listdir(datadir)
print(CATEGORIES)
['no_mask', 'mask']

   # Define two empty list to contain image data
x, y = [], []

def PreProcess():
for category in CATEGORIES:
     path = os.path.join(datadir, category)
     classIndex = CATEGORIES.index(category)
     print(path)
     for imgs in tqdm(os.listdir(path)):
        img_arr = cv2.imread(os.path.join(path, imgs))

        # resize the image
        resized_array = cv2.resize(img_arr, (img_size,
img_size))
        cv2.imshow("images", resized_array)
        cv2.waitKey(1)
        resized_array = resized_array/255.0
        x.append(resized_array)
        y.append(classIndex)

PreProcess()
```

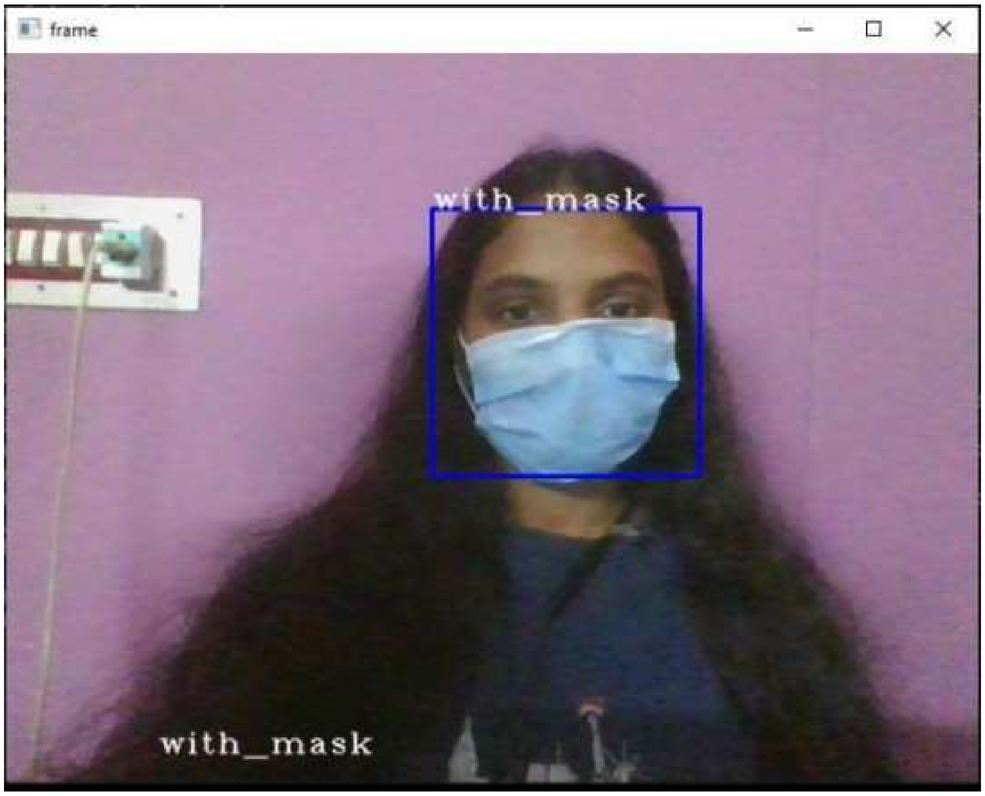
```
cv2.destroyAllWindows()
# Split data for training and testing
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
                random_state=42)

 # Convert and resize the data to a numpy array
X_train = np.array(X_train).reshape(-1, img_size, img_size, 3)
y_train = np.array(y_train)
X_test = np.array(X_test).reshape(-1, img_size, img_size, 3)
y_test = np.array(y_test)
# Create the model architecture

model = Sequential()

model.add(Conv2D(64,(3, 3), input_shape=(img_size, img_size, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(16, activation='relu'))

model.add(Dense(len(CATEGORIES)))
model.add(Activation('softmax'))

# compile the model
```

```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',
metrics=['accuracy'])
model.summary()
batch_size = 32
epochs = 15

t1 = time.time()
# fit the model
model.fit(X_train, y_train, batch_size = batch_size, epochs=epochs,
validation_split=0.3, verbose = 1)
model.save('{}.h5'.format("model"))
t2 = time.time()
print('Time taken: ',t2-t1)
# Create the model architecture

model = Sequential()

model.add(Conv2D(64,(3, 3), input_shape=(img_size, img_size, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(16, activation='relu'))

model.add(Dense(len(CATEGORIES)))
model.add(Activation('softmax'))
```

```
# compile the model

model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',
metrics=['accuracy'])
model.summary()
prediction = model.predict(crop_img)
      index = np.argmax(prediction)
      res = CATEGORIES[index]
      if index == 0:
            color = (0, 0, 255)
      else:
            color = (0, 255, 0)
      cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
      cv2.putText(frame, res, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX,
0.8, color, 2, cv2.LINE_AA)
except:
      pass

  cv2.imshow("frame", frame)
  if cv2.waitKey(1) == ord('q'):
        break
```

5.2. SNAPSHOTS





CHAPTER 6
CONCLUSION AND REFERENCES

CONCLUSION

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare. The architecture consists of Mobile Net as the backbone it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. We used OpenCV, tensor flow, and NN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. The accuracy of the model is achieved and, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyper parameters. This specific model could be used as a use case for edge analytics. Furthermore, the proposed method achieves state-of-the-art results on a public face mask dataset. By the development of face mask-detection we can detect if the person is wearing a face mask and allow their entry would be of great help to the society

REFERENCES

[1] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020

[2] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in Advances in neural information processing systems, 2014, pp. 198hy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” 2014. [4] F. S. Samaria and A. C. Harter, “Parameterisation of a stochastic model for human face identification,” in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pp. 138–142, IEEE, 1994.

[3] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning face representation from scratch,” CoRR abs/1411.7923, 2014.

[4] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, “A practical transfer learning algorithm for face verification,” in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IE8–1996.

[5] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, “Face recognition: A literature survey,” ACM computing surveys (CSUR), vol. 35, no. 4, pp. 399–458, 2003.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. KarpatEE, 2013.

[7] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection,” Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(7), 720, 1997.

[8] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, “A practical transfer

learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IEEE, 2013.

[9] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. CoRR, abs/1406.4773, 2014. 1, 2, 3

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 1986. 2, 4