GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

FACE MASK DETECTION USING DEEP LEARNING TECHNIQUES

A project report submitted in partial fulfilment of the requirement

for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. LOKESH KUMAR	**17341A0582**
K. SWAROOPA RANI	**17341A0572**
J. LAHASYA	**17341A0567**
K. RAKESH	**17341A0583**
M. MOUNICA SAI	**17341A05A7**

Under the esteemed guidance of

Ms. Y. DIVYA BHARATHI
Assistant Professor, Dept. of CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2008 Certified Institution)

GMR Nagar, Rajam – 532127,
Andhra Pradesh, India
Aug 2021

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Main Project entitled "**FACE MASK DETECTION USING DEEP LEARNING TECHNIQUES"** submitted by **K. LOKESH KUMAR, K. SWAROOPA RANI,J.LAHASYA, K. RAKESH, M. MOUNICA SAI** bearing Reg.No: **17341A0582, 17341A0572, 17341A0567, 17341A0583, 17341A05A7** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **GMRIT, Rajam** affiliated to **JNTUK, KAKINADA** is a record of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of Supervisor
Ms. Y. Divya Bharathi
Assistant Professor
Department of CSE
GMRIT, Rajam.

Signature of HOD
Dr. A. V. Ramana
Professor & Head
Department of CSE
GMRIT, Rajam.

The report is submitted for the viva-voce examination held on ……..

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to my guide **Ms. Y. Divya Bharathi,** Assistant Professor, Department of Computer Science and Engineering for her whole hearted and invaluable guidance throughout the project work. Without her sustained and sincere effort, this project work would not have taken this shape. She encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. A. V. Ramana**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr. C. L. V. R. S. V. Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of Computer Science and Engineering for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

K. LOKESH KUMAR **17341A0582**
K. SWAROOPA RANI **17341A0572**
J. LAHASYA **17341A0567**
K. RAKESH **17341A0583**
M. MOUNICA SAI **17341A05A7**

ABSTRACT

Now-a-days many people are suffering with COVID-19, Ultimately it is leading people to death. In order to overcome this problem, we should initially detect whether the person is wearing mask or not. Face Mask Detection had seen vital progress in the fields of Image processing and Computer vision. Many Faces Detection models have been designed using several deep learning techniques which can read faces and mask straight from the data provided by the user. Basically, this problem is divided into two stages: Face detection and Mask Detection. (1) Single Face can be identified using CNN while a group of faces can be identified using Multi-Task Cascaded Convolutional Neural Networks (MTCNN). (2) Faces with mask can be detected with the help of various Machine Learning Algorithms. The suggested approach in this project uses deep learning and frameworks like TensorFlow, Keras, and OpenCV to detect face masks. For Mask Detection we can use Deep Learning Techniques like CNN for feature extraction. Moreover, it can be employed for safety purposes since it is very resource-effective to deploy. The dataset has been presenting in this project, was collected from Kaggle, can be adopted by other researchers for further advanced models such as those of Face Recognition, Facial landmarks, and Facial part detection process.

Keywords: Convolutional Neural Network, Multi-Task Cascaded Convolutional Neural Networks, Face Mask, TensorFlow.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS & ABBREVIATIONS

NCoV	:	Novel Coronavirus
WHO	:	World Health Organization
CoV	:	Coronaviruses
MTCNN	:	Multi-Task Cascaded Convolutional Neural Networks
CNN	:	Convolutional Neural Networks
PCA	:	Principal Component Analysis
OpenCV	:	Open-Source Computer Vision Library
AI	:	Artificial Intelligence
DNN	:	Deep Neural Networks
GPU	:	Graphical Processing Unit
CPU	:	Control Processing Unit
ANN	:	Artificial Neural Network

1. INTRODUCTION

A new strain of novel coronavirus (nCoV) previously which is not identified in the humans. The virus was identified in human was found in December 2019. This coronavirus was risen up in Wuhan, China. On march 11, 2020 the world health organization (WHO) declared it as a deadly disease which is affected more than 114 countries all over the world. Coronaviruses (CoV) are group of viruses which cause illness that range from colds to deadly infections. The medical professionals, healthcare organizations and researchers are in search for a proper vaccines and medicines to overcome this deadly disease. This COVID-19 has become a pandemic all over the world.

People all over the world are facing challenging situations due to this pandemic. Every day the large number of people are affected by this coronavirus. The death rate is also high. The virus spreads through air. When an infected person sneezes or communicate with the other person, the water droplets from their nose or mouth passes through air and affect to the other people. The major symptoms of this virus which is declared by the WHO are fever, dry cough, tiredness, diarrhea, loss of taste, smell and shortness of breath. The main precautions to prevent the coronavirus is washing hands for 2mins, using sanitizer, maintaining social distance nearly about 6 feet and wearing a mask. To avoid touching nose and mouth, wearing the facemask is the simplest one.

1.1.WHY FACE MASK RECOGNITION SYSTEM?

Wearing a mask is one of the necessary precautions to protect ourselves from the virus. Whereas the spread of virus will be controlled from human to human. Wearing a mask is essential, particularly for those people who are at a greater risk of severe illness from COVID-19 diseases. It recommended to all the people to wear a mask in public areas, especially when social distancing is difficult. But some people are not obeying the rules properly. Due to this the passing of virus is becoming speed. By detecting the facemask, the growth of COVID-19 can reduce.

We are developing a system to identify whether the people are wearing a mask or not. We focus on unmasking of masked face because it is a very interesting problem of great practical value. It is similar to detect any object from a scene.

Deep learning architectures have shown a remarkable role in object detection. These architectures can be including in detecting the mask on a face. The images are used to identify the mask on the face. The suggested approach in this project uses deep learning techniques and frameworks like Tensor Flow, Keras, and OpenCV to detect face masks. For Mask Detection we can use Deep Learning Techniques like CNN for feature extraction from the images then these features are learned by multiple hidden layers while a group of faces can be identified by using MTCNN (Multi-Task Cascaded Convolutional Neural Networks).

1.2. PROBLEM STATEMENT

Spreading of corona virus can be reduced by wearing face mask. But some people are neglecting wearing the face mask. This is also a reason for affected by the virus and leads to the death. Manually, checking each and every person whether they are wearing a facemask or not is very difficult. It is very important to develop an effective model to detect the people are wearing the facemask or not. By this automated model we can easily detect. This study is focused on developing on the deep learning models to predict the facemask.

1.3. EXISTING WORK

Covid19 pandemic has huge impact on many countries. Wearing face mask is one of the main precautions to be taken to decrease spread of virus. So, face mask detection is implemented. Face mask detection system involves classification of people wearing masks and people not wearing the mask. Face mask detection mainly contains two steps they are first detection of face and second step is detection of face mask. This can be done using machine learning techniques. DNN module from OpenCV, SSD are used for detection of faces and per-trained method mobileNETV2, SVM is used for classifying the images with masks and without mask. This system is useful for giving alert to the people for not wearing mask and making people to wear the mask which helps in decreasing the spread of virus.

DISADVANTAGES:

- These methods can only detect in single face images.
- Computational time is high.
- Accuracy of this method is low.
- Low accuracy for low resolution images.

1.4. PROPOSED WORK

Wearing face mask is essential measure for preventing us from covid19 virus. So, face mask detection method is implemented to classify images with face mask and without mask. In our propose method we are using deep learning techniques for classifications. TensorFlow, keras and OpenCV frameworks are also used for the classification. In our methods we are using Multi Cascade Neural Network (MTCNN) for detecting faces which detects multiple faces in the images. This output is given as input to the Convolutional Neural Network (CNN) which is trained by the data set taken from the Kaggle for identifying faces with mask and without mask. Our method gives result as showing red rectangular box for faces without mask and green rectangular box for faces with mask. In our model, we had overcome the disadvantages of the existing work.

ADVANTAGES:

- This method is used for multiple face images.
- This method has high accuracy.
- This method can be used for low resolution images.
- It can detect faces with different sizes.
- This method is very robust.

2. LITERATURE SURVEY

1.Bosheng Qin,Dongxiao Li., et.al[2020] "**Identifying facemask-wearing condition using image super-resolution**" proposed an approach (SRCNet) for classifying face-mask wearing. The model is trained and evaluated on a dataset that contained a total of 3835 images, out of the 3835 images, 671 contain faces without masks, 134 images contain faces with incorrectly worn masks and 3030 images contain faces with correctly worn face-masks. An accuracy of 98.70% is reported for the proposed model. But, the disadvantage of this model was they do not focus on low-resolution faces and gives the accuracy of 73.5%.

2.Vinitha.V1, Velantina.V2.,et al [2020] "**FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION**" proposed system focuses on how to identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm by using the OpenCV, Tensor flow, Keras. An accuracy of 93.12% is reported for the proposed model. The drawback of using only MobileNetV2 is it takes more training time when compared to other approach.

3.Md.Sabbir Ejaz., et al[2020] " **Face Mask Detection using Deep Learning Techniques**" proposed that the method Principal Component Analysis (PCA) algorithm for masked and un-masked facial recognition. It was noticed that PCA is efficient in recognizing faces without a mask with an accuracy of 96.25% but its accuracy is decreased to 68.75% in identifying faces with a mask. There are some problems in the time of data processing and extracting feature of face mask images.

4. C.Jagadeeswari, M. Uday Theja., et al[2020] "**Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers**" used to detect faces in the images/video stream. MobileNetV2, ResNet50, and VGG16 are used to generate a trained model. It performs a prediction with in about 0.12s with an accuracy of 68.4%.

5. Insure B venkateswaralu, Jagadeshkakrla et al, [2020]" **Face mask detection using Mobilenet and global pooling**" in this paper they have proposed a pre trained mobile Net with a global pooling block for face detection. Mobile Net takes image and create multidimensional feature map. Global pooling layer convert this into feature vector. This method is trained with dataset consists of 3833 color images. Out of which 1918 images are without a mask and the remaining 1915 images are with a mask. this proposed method archives high accuracy but cannot be implemented on images with multiple faces.

6. Amit Chavda, Jason Dsouza et al,[2019]"**Face mask detection using multi stage CNN**" in this paper author praposed a r, a two-stage Face Mask Detector The first stage uses a pretrained RetinaFace model for robust face detection the NASNetMobile based model was selected for classifying faces as masked or non-masked. this Stage 1 and Stage 2 models can be easily replaced with improved models in the future, that would give better accuracy and lower latency.

7. Abdellahoumina et ai,[2020]"**face mask detector using Transfer learning**" in this paper they used deep convolutional neural network to extract deep features from images of the faces. These features are processed using SVM for classification of masked and un masked faces. This method achived good accuracy of 97.1%but it is not trained with the large dataset, it is dataset contains1736 images.

8.Mohammad Marufur Rahman, Md. MotalebHossenManik et al,[2020].**" An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network".**In this paper, they had proposed the deep learning architecture. This architecture highly depends on convolutional neural network for feature extraction. The extracted features are used by dense neural networks for classification purposes. The proposed system having high accuracy of 98.7%. The future work in this paper is, the people near to the person who is not wearing mask may be alerted by an alaram signal on that location to maintain a safe distance.

9.Samuel Ady Sanjaya, Suryo Adi Rakhmawan [2020]. **"Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic".** In this paper, the face mask recognition is developed with an machine learning algorithm via image classification method mobileNetV2. It is based on the convolutional neural networks. The proposed model having

the high accuracy of 96.8%. The future scope of this paper is, this model may integrate with a system which is implementing the social distancing.

10.Sunil Singh, Umang Ahuja et al, [2020]. **"Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment".** In this paper the F-RCNN and YOLOV3 algorithm is compared by the average precision. As the F-RCNN has the better precision. But the YOLO algorithm is used for the real world as it performs the single shot detection. As it also gives the high accuracy. The future scope of this paper is to improve the model by training on the large datasets.

11. Sneha Sen,Dr.Harish Patidar et al,[2020] **"Face Mask detection System for COVID_19 Pandemic Precautions using Deep Learning Method".** In this paper author proposed a mask detection system that is able to detect any type of mask and masks of different shapes from the video streams. Deep learning algorithm and framework of MObileNetV2is used here and the PyTorch library and OPENCV of python is used for mask detection from images/video streams. The accuracy for the training and validation set is compared and found to be of 79.24%.

12. ZekunWang, PengweiWang, PeterC, Louis, LeeE, Wheless and YuankaiHuo et al,[2020] **"WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19".** In this paper author proposed a Web based efficient AI recognition of masks. The contribution is to provide a holistic edge-computing framework in integrating (1) deep learning models (2) neural network inference computing network (3) a stack-based virtual machine. For end users (1) installation free deployment (2) low computing requirements (3) high detection speed. This would give high accuracy. The drawback is insufficient support of deep learning from JavaScript by aggregating NCNN and WASM.

13. PreetiNagrath, Rachna Jain, Agammadan, Rohan Arora, PiyushKataria, and Jude Hemanth et al [2020] **"SSDMNV2: A real time DNN-based face mask detection system using single shot multi box detector and MobileNetV2".** In this paper author proposed a Single Shot multi box 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 to perform real-time mask detection. This paper gives accuracy of 0.9264 and an FI score of 0.93. It generates better accuracy but has faced various wrong predictions.

3. DESIGN REQUIREMENTS

3.1. Hardware Requirements:

System: Laptop

Hard disk: 1TB

Ram:8gb(min)

3.2. Software Requirements:

Operating system: Windows 10

Libraries: TensorFlow, Cv2, Keras, MTCNN

Language: python

4. THEORETICAL STUDY

The main domain we preferred for detection of facemask is “Deep Learning”. Deep learning is a collection of algorithms used in machine learning, used to model high-level abstractions in data using model architectures, which are composed of multiple nonlinear transformations. Deep Learning take less time to test the data than Machine Learning. First, we need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, we need to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the dataset. Fifth, Final testing should be done on the dataset. Corona viruses (CoV) are a large family of viruses that cause illness ranging from the common cold to more severe diseases. The virus spreads mainly between people who are in close contact with each other. A person can be infected when droplets containing the virus are inhaled or come directly into contact with the eyes, nose, or mouth. Coronaviruses can cause colds with major symptoms, such as fever, breathing problem etc. As the corona virus mainly affect to the lungs. As it is affected to the lungs firstly, the breathing problem will occur and after it can leads to death. The total cases in worldwide, as of now the people affected by the corona virus are 187,775,434. The total cases will be changing day by day. As the death rate also changes. It is a very dangerous virus. The main precautions to prevent the coronavirus is using face mask and maintaining the social distance. But some people are neglecting to wear a mask. Due to this it may affect to the other people. Because of this problem we are developing a model using deep learning to detect the face mask for multiple faces whether the people are wearing a mask or not. So, in our model we are using the deep learning techniques are CNN and MTCNN. And also, we are using the frameworks which are TensorFlow, Keras and OpenCV.

4.1. INTRODUCTION TO DEEP LEARNING

Deep learning is a machine learning technique that teaches a computer to filter inputs through layers in order to learn how to detect and classify information. It is a part of the machine learning family that is based on *learning data representations*. It is not task specific algorithm. Deep learning algorithms are inspired by artificial neural networks. Deep learning techniques can supervise or semi supervised or unsupervised techniques.

4.2. INTRODUCTION TO KERAS, TENSORFLOW AND OPENCV

Keras is high level python library. It is made with focus of understanding deep learning techniques, such as creating layers for neural network. Data structure of keras is layers and the model. Keras consists of sequential model which contains linear stack of layers. Keras is used to develop new training procedures quickly.

TensorFlow is a software library or framework, designed by the Google to implement machine learning and deep learning techniques easily. It consists a feature that optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. The library has more than 2500 optimized algorithms. These algorithms can be used to detect and recognize faces, identify objects. OpenCV supports programming languages like java, python etc.

Keras:

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed for fast experimentation.

Deep learning is one of the major subfields of machine learning framework. Machine learning is the study of design of algorithms, inspired from the model of human brain. Deep learning is becoming more popular in data science fields like robotics, artificial intelligence (AI), audio & video recognition and image recognition. Artificial neural network is the core of deep learning methodologies. Deep learning is supported by various libraries such as Theano, TensorFlow, Caffe, Mxnet etc., Keras is one of the most powerful and easy to use

python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning models.

- Keras is user friendly, modular, easy to extend, and to work with Python.
- The API was designed for human beings, not machines.
- Keras is an industry-strength framework that can scale to large clusters of GPUs.
- Keras offers consistent & simple APIs. It provides clear & actionable error messages.

Neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files. You can export Keras models to JavaScript to run directly in the browser, to TF Lite to run on iOS, Android, and embedded devices.

It's also easy to serve Keras models as via a web API. Keras is a central part of the tightly-connected TensorFlow 2.0 ecosystem, covering every step of the machine learning workflow, from data management to hyperparameter training to deployment solutions.

Why Keras?

- The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly.
- Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MX Net, and PlaidML), and strong support for multiple GPUs and distributed training.
- Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

Keras APIs:

Models API:

- The Model classes
- The Sequential class
- Model training APIs

- Model saving & serialization APIs

Layers API:

- The base Layer classes
- Core layers
- Convolution layers
- Pooling layers
- Recurrent layers
- Pre-processing layers
- Normalization layers
- Regularization layers
- Attention layers
- Reshaping layers
- Merging layers
- Locally-connected layers
- Activation layers

Callbacks API:

- Base Callback class
- ModelCheckpoint
- TensorBoard
- EarlyStopping
- LearningRateScheduler
- ReduceLROnPlateau
- RemoteMonitor
- LambdaCallback
- TerminateOnNaN
- CSVLogger
- ProgbarLogger

Data Pre-processing:

- Image data pre-processing
- Timeseries data pre-processing
- Text data pre-processing

Optimizers:

- SGD
- RMSprop
- Adam
- Adadelta
- Adagrad
- Adamax
- Nadam
- Ftrl

Metrics:

- Accuracy metrics
- Probabilistic metrics
- Regression metrics
- Classification metrics based on True/False positives & negatives
- Image segmentation metrics
- Hinge metrics for "maximum-margin" classification

Losses:

- Probabilistic losses
- Regression losses
- Hinge losses for "maximum-margin" classification

Built-in small datasets:

- MNIST digits classification dataset
- CIFAR10 small images classification dataset

- CIFAR100 small images classification dataset
- IMDB movie review sentiment classification dataset
- Reuters newswire classification dataset
- Fashion MNIST dataset, an alternative to MNIST
- Boston Housing price regression dataset

Keras Applications:

- Xception
- EfficientNet B0 to B7
- VGG16 and VGG19
- ResNet and ResNetV2
- MobileNet and MobileNetV2
- DenseNet
- NasNetLarge and NasNetMobile
- InceptionV3
- InceptionResNetV2

Utilities:

- Model plotting utilities
- Serialization utilities
- Python & NumPy utilities
- Backend utilities

Architecture of Keras:

Keras API can be divided into three main categories –

- Model
- Layer
- Core Modules

In Keras, every ANN is represented by Keras Models. In turn, every Keras Model is composition of Keras Layers and represents ANN layers like input, hidden layer, output layers,

convolution layer, pooling layer, etc. Keras model and layer access Keras modules for activation function, loss function, regularization function, etc. Using Keras model, Keras Layer, and Keras modules, any ANN algorithm (CNN, RNN, etc.,) can be represented in a simple and efficient manner.

Features:

It supports the following features –

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- It has High scalability of computation.

Benefits:

Keras is highly powerful and dynamic framework and comes up with the following advantages

- It has larger community support.
- It is easy to test.
- Keras neural networks are written in Python which makes things simpler.
- Keras supports both convolution and recurrent networks.

Deep learning models are discrete components, so that, you can combine into many ways.

TensorFlow:

TensorFlow is an open-source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy-to-understand framework.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. TensorFlow is also called a “Google” product. It includes a variety

of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

Basics:

The word TensorFlow is made by two words, i.e., Tensor and Flow.

- Tensor is a multidimensional array
- Flow is used to define the flow of data in operation.

TensorFlow is used to define the flow of data in operation on a multidimensional array or Tensor.

Components of TensorFlow:

- Tensor
- Graphs

Tensors are used as the basic data structures in TensorFlow language. Tensors represent the connecting edges in any flow diagram called the Data Flow Graph. Tensors are defined as multidimensional array or list.

Tensors are identified by the following three parameters:

Rank:

Unit of dimensionality described within tensor is called rank. It identifies the number of dimensions of the tensor. A rank of a tensor can be described as the order or n-dimensions of a tensor defined.

Shape:

The number of rows and columns together define the shape of Tensor.

Type:

Type describes the data type assigned to Tensor's elements.

A user needs to consider the following activities for building a Tensor

- Build an n-dimensional array
- Convert the n-dimensional array.

Features:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Features:

- Responsive Construct
- Flexible
- Easily Trainable
- Parallel Neural Network Training
- Large Community
- Open Source
- Feature Columns
- Availability of Statistical Distributions
- Layered Components
- Visualizer (With Tensor Board)
- Event Logger (With Tensor Board)

Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas:

- It has a fast and efficient Data Frame object with the default and customized indexing.
- Used for reshaping and pivoting of the data sets.
- Group by data for aggregations and transformations.
- It is used for data alignment and integration of the missing data.
- Provide the functionality of Time Series.
- Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.
- Handle multiple operations of the data sets such as sub setting, slicing, filtering, group By, re-ordering, and re-shaping.
- It integrates with the other libraries such as SciPy, and scikit-learn.
- Provides fast performance, and if you want to speed it, even more, you can use the Python.

Benefits of Pandas:

The benefits of pandas over using other language are as follows:

- **Data Representation:** It represents the data in a form that is suited for data analysis through its Data Frame and Series.
- **Clear code:** The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear and concise code for the user.

Data Set

The data set that is used in implementing the model is taken from Kaggle. It contains a total of 11042 images. The images contain 2 different classes which are Mask and Without Mask. Some other data sets are also available such as JAFFE, CK+ etc. But this is mostly used and have vast number of images. They are colour images which are having their respective images in their respective class. Here, we need to classify the data for training and testing. In our project, we classified the training and test data in the ratio 70:30. It means 70% of the data is used for training and remaining 30% of the data is used for testing. So, the total training images are 7729 and the total testing images are 3313.The images values present in the numerical

format like ‘0’ as Without Mask, ‘1’ as Mask. The image is represented in the form of pixels in a row. The entire data is shuffled to train the data.

Data Visualization

As mentioned earlier, the dataset contains 11042 images of various classes. Each class has multiple number of images in the data set. The data set contains mask and without masks images. Now we look at the number of images of each class through visual graph.

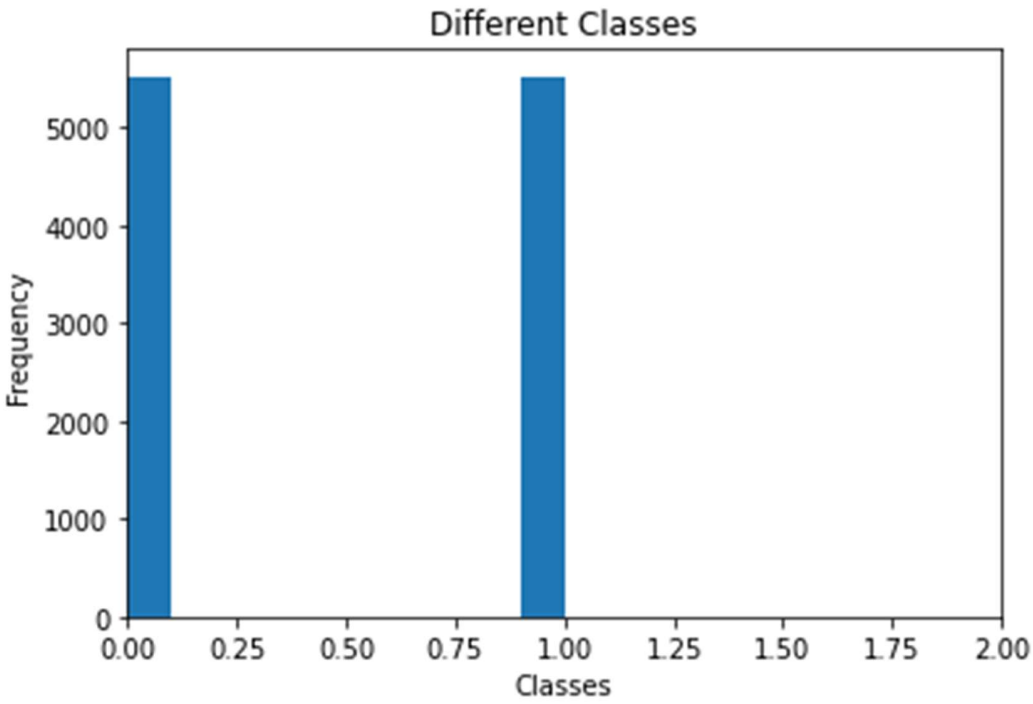

Fig 4.1: Visualization of dataset

In the **Fig 4.1**, the x-axis represents classes and y-axis represents the frequency which is nothing but the number of images. The dataset contains 2 classes. So, the graph shows variations which represents the number of images of each class. We can observe that the classes having same number of images. Then the number of images is for each class having 5521 images. So, total number of images 11042.

5. EXPERIMENTAL STUDY

In this project, we implemented a model that detects the facemask of a single person or a group of people in the image. We have two different segments for this model. As we are doing it for multiple faces in the image, we need to identify the faces. Then, for the detected faces we need to identify with mask and without mask using a deep learning model.

- Training the model
- Identify multiple faces and detect using the trained model

ARCHITECTURE OF OUR MODEL

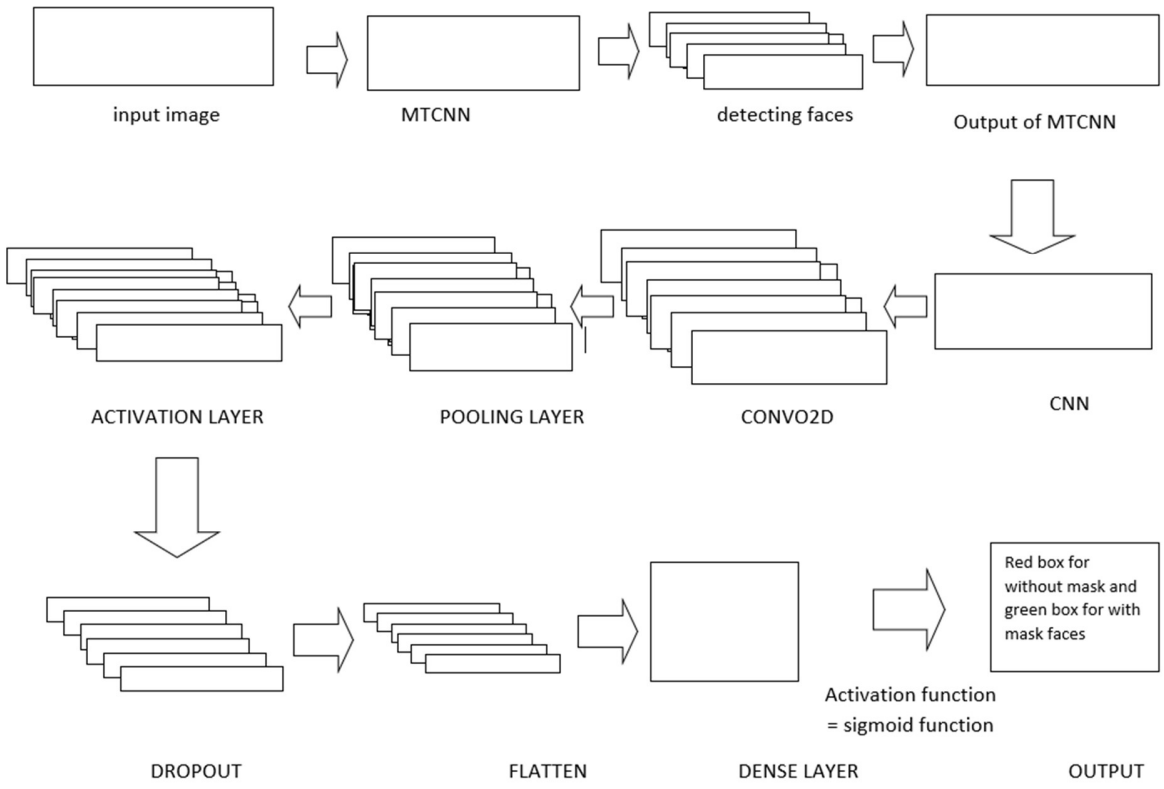

Fig 5.1 Architecture

5.1 Training the CNN model:

The model is implemented for multi-class classification i.e., for detecting multiple classes in an image. The first phase is to train the model by considering the required dataset. In this

project, we trained the model on dataset taken from Kaggle, which is mostly used for to detect Face Mask.

Table 5.1 Training set and testing set

Classification	Training Set	Testing Set
Mask	3867	1654
Without Mask	3867	1654

The data set contains images of various shapes, color and size. Initially, we need to normalize the data which the values will be converted in the range of 0 to 1. Also, we need to resize the data in to required pixels such as 48*48, 64*64, 128*128 etc.

Now, import the required packages and methods such as keras, NumPy etc. The data should be trained on train data where the entire dataset will be divided in to train and test data using methods available sklearn package. The CNN model is applied to the particular train data.

Layers used in the model:

- Conv. layer
- Activation layer
- Pooling layer
- Drop outs
- Dense layer
- Flatten layer
- Dense layer with Sigmoid function

Generally, CNN is a deep learning algorithm which takes an input image and assign some learnable weights and bias to various aspects of the image. In the proposed CNN model, the Conv. Network contains multiple layers to train the model. The accuracy depends upon the number of layers and parameters. If the layers are too less, the model doesn't train properly and it results in deviation and gives more error rate which is nothing but underfitting. If the layers are more, it undergoes overfitting. The various layers present in CNN model are Conv. Layer, Max pooling layer, Dense Layer, Flatten layer etc.

Firstly, the input is given to Convolution layer. The first convolution layer contains 32 filters with input image size 64*64. This layer extracts the important features from the input image. The mathematical operation takes place between image matrix and a filter or kernel. Then the output of CNN layer is given as input to Batch Normalization. This layer can be used before or after the activation layer. It helps us to stabilize the learning process and increases the learning rate. The main advantage of Batch Normalization is to reduce dependency on scale of the parameters like removing or lowering the drop-out rate and has regularization effect. We mention the batch size for training. If we don't pass any value, it takes a batch size of 32 by default. Then it is given to the activation layer which contains RELU as an activation function. RELU provides just enough non linearity and it is quite simple as linear representation. The more RELU units are used, the more it becomes nonlinear.

RELU Function:

$$y = \max(0, x)$$

The functionality of RELU is it avoids vanishing gradient problems is that it has much lower run time. It is much faster than sigmoid and tanh function. Then it is passed to the pooling layer. The pooling layer is to reduce the spatial size of the image representation and to reduce the number of parameters and computation. We have various types of pooling like Max Pooling. It down-samples the given input by considering the maximum element from the region of feature map. In our model, we considered pool-size of (2,2) which means the computation is done on 2*2 matrix. The computational cost is also reduced as the number of parameters to train gets decreased. Then drop outs are added. The purpose of drop-out is to avoid over-fitting. If we use drop-outs, the randomly selected neurons are ignored during the training process. They are dropped out randomly from a set of nodes. We mention the number of nodes that is to be dropped as parameter. In our model, we considered 0.20 which means 20% of the neurons are dropped and remaining neurons are activated. This means activation of those neurons are temporarily removed in the forward propagation and any weight updates are not applied in the back propagation.

In the same way, the convolution layers are added with filters which improves the training accuracy with increase in layers. Then we add dense layer i.e., fully connected layers along with activation and drop out layers. The usage of dense layer is it is difficult to map

large number of neurons to output layer of 2 classes. So, we implemented the dense layers and then given to the next layer. Finally, we add a Sigmoid layer, which is generally used for Binary-class classification. Here, we mention 2 in the parameters as we have 2 classes to classify the image. It is implemented just before the output layer in the neural network. It contains same number of nodes as the output layer. It normalizes the output of neural network to fit between 0 to 1. It represents certain probability for each class in the network output layer.

Sigmoid Function:

$$S(x)= 1/(1+e^{(-x)})$$

Then the model is trained for a fixed number of epochs by fitting the model. Epochs, which is a hyperparameter that defines the number of times the algorithm works on the entire training data. In each epoch, the entire data is used exactly once. An epoch is made of one or more batches, where we use part of the model to train the network. In our implementation, we used a batch size of 32 and has a total of 242 batches for each epoch.

The ReduceLROnPlateau is a class in which it reduces learning rate when a metric has stopped improving. The call back monitors the quantity and if there is no improvement is done for a 'patience' number of epochs, then automatically learning rate is reduced. Patience is nothing but number of epochs with no improvement after which the training will get stopped. By how much the learning rate to be reduced is based on the value of factor.

$$New\_lr=lr*factor \text{ (lr = learning rate)}$$

The Model Checkpoint is implemented which allows us to define where to check point the model weights, how the file should be named and what circumstances to make a checkpoint. We must specify which metric to monitor such as accuracy or loss on the training or validation data set. Validation loss, verbose, name to save model, mode is passed as an argument in the implementation. Mode is considered to be max to save the best model. Here, we also used call backs which is used as an argument to the fit () function to save a model at specific intervals.

FLOW CHART FOR CNN MODEL

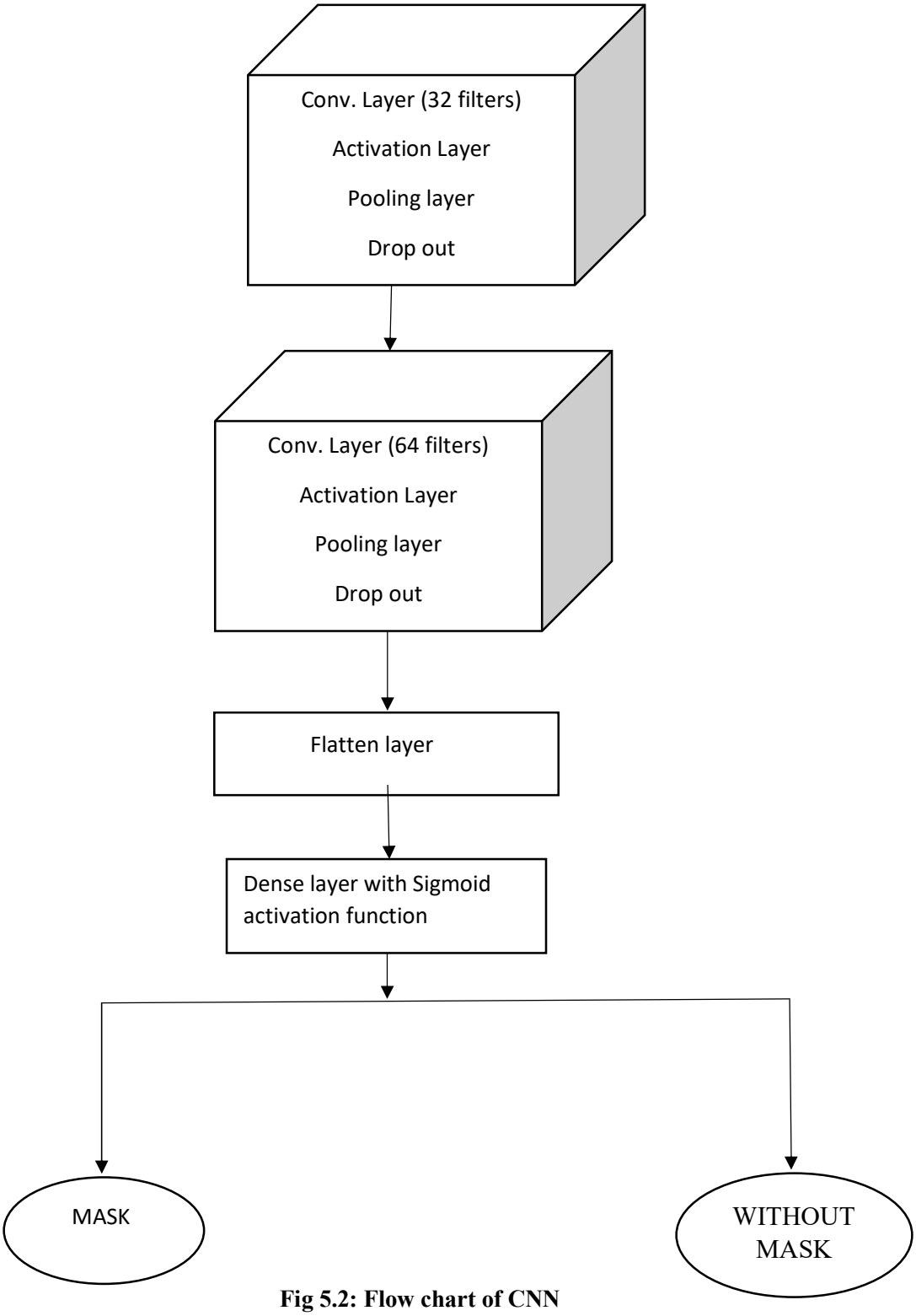

Fig 5.2: Flow chart of CNN

ALGORITHM:

INPUT: Image

OUPUT: Detecting Face Mask

STEP1: Load Images**.**

STEP2: Process the images, Resizing.

STEP3: Load the Filenames and respective labels.

STEP4: Shuffling the images**.**

STEP5: Split data into Training and Testing batches**.**

STEP6: Load CNN model and adding Kernel layers, Max pooling, Dropouts, Flatten**.**

STEP7: Compiling using Adam optimizer.

STEP8: Detection.

STEP9: Save the Model.

Model Summary:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 12544)             0
_________________________________________________________________
dense (Dense)                (None, 15000)             188175000
_________________________________________________________________
dropout_1 (Dropout)          (None, 15000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 7000)              105007000
_________________________________________________________________
dense_2 (Dense)              (None, 2000)              14002000
_________________________________________________________________
dense_3 (Dense)              (None, 300)               600300
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 301
=================================================================
Total params: 307,803,993
Trainable params: 307,803,993
Non-trainable params: 0
_________________________________________________________________
```

Fig 5.3: Summary of CNN

From the **Fig 5.3**, we can observe that we used 2 convolution layers, 2 pooling layers, 2 drop outs, 4 dense layers, flatten layer, and a Sigmoid layer.

It has a total of 30,78,03,993 parameters for training. But the number of trainable parameters is only 30,78,03,993 which are used for detect the wearing a mask or not.

Details of the proposed CNN Network:

Table 5.2: Details of CNN Network

TYPE	LAYER	OUTPUT SHAPE
Conv2d	Cov2d	(62, 62, 32)
Maxpooling2D	Max_pooling2D	(31, 31, 32)
Conv2D	Conv2d_1	(29, 29, 64)
Maxpooling2	Max_polling2d_1	(14, 14, 64)
Dropout	dropout	(14, 14, 64)
Flatten	flatten	(12544)
Dense	dense	(15000)
Dropout	Dropout_1	(15000)
Dense	Dense_1	(7000)
Dense	Dense_2	(2000)
Dense	Dense_3	(300)
Dense	Dense_4	(1)

Training process of the model:

```
Epoch 1/5
242/242 [==============================] - 52s 86ms/step - loss: 0.4127 - accuracy: 0.9024 - val_loss: 0.1900 - val_accuracy: 0.9303
Epoch 2/5
242/242 [==============================] - 19s 79ms/step - loss: 0.1167 - accuracy: 0.9614 - val_loss: 0.2579 - val_accuracy: 0.9221
Epoch 3/5
242/242 [==============================] - 20s 81ms/step - loss: 0.1116 - accuracy: 0.9620 - val_loss: 0.1022 - val_accuracy: 0.9746
Epoch 4/5
242/242 [==============================] - 19s 80ms/step - loss: 0.0902 - accuracy: 0.9713 - val_loss: 0.1020 - val_accuracy: 0.9656
Epoch 5/5
242/242 [==============================] - 19s 80ms/step - loss: 0.0713 - accuracy: 0.9746 - val_loss: 0.0741 - val_accuracy: 0.9795
```

Fig 5.4: Training process

Training the model

The above **Fig 5.4** represents the training process of the CNN model. It displays the epoch number and batch that is training out of total batches. Next, we can observe the time taken to run in the form ms/step. Then it also represents the loss and accuracy for each epoch. In the validation set, we gave test images for validating. It gives us the validation loss and validation accuracy.

5.2 Identifying multiple faces and detect using the trained model:

Generally, for detecting single face in the image, CNN can be applied to detect face and the expression. But, for detecting multiple we need a different model MTCNN which is mostly used for detecting multiple faces in the image. Multi-task Cascaded convolution networks (MTCNN) is one most popular and accurate face detection tool used today. MTCNN is a framework which was implemented for both face detection and face alignment. It recognizes the faces and also land mark locations such as eyes, nose, mouth.

The MTCNN uses a cascade structure with three networks. Firstly, the image is rescaled to a range of different sizes which is formed as an image pyramid. Then the first model proposal Network (P-Net) checks for the candidate facial regions, the second model Refine Network (R-Net) filters the bounding box for the faces and the third model Output Network (O-Net) gives us facial landmarks such as corners of eyes and mouth, nose. For our model, we just need to have only bounding box for faces and there is no requirement for the facial landmarks.

For the implementation of MTCNN, we need to install the mtcnn package through pip. Then we need to import the mtcnn method and we give the detector as mtcnn. Then we labelled the classes as Mask and No Mask.

Now, we input the required images to detect the face mask. Then, the method detector detects the faces present in the image. Now, we draw the bounding box for the obtained faces using ‘box’ for each face. We can also mention the color of the bounding box for each class in different ways. The image needs to be resized in to 64*64 because the data is trained in that particular shape. If we consider the size other than that, the image doesn’t match trained data

shape and display an error. The channel also needs to be mentioned when it gets reshaped because for a color image it has 3 channels and for a gray-scale it has only one channel.

The model will detect the face mask or not using predict method which has probabilities for all the classes. Then the highest of all the classes is considered to be best suitable for the face and is displayed. For the text to be displayed on the face, we need to mention color of the text, font, size.

For printing the output, we implemented a method that dynamically adjust the images in a plot of (128,128). Suppose if we give 3 test images, the entire 128*128 is divided in to 3 sub-plots and the images will be adjusted into those plots in-order as given in input. If we give only one image, the entire plot will be taken by this image. That means, based on the input data given it will be adjusted dynamically in the plot.

FLOW CHART FOR MTCNN MODEL

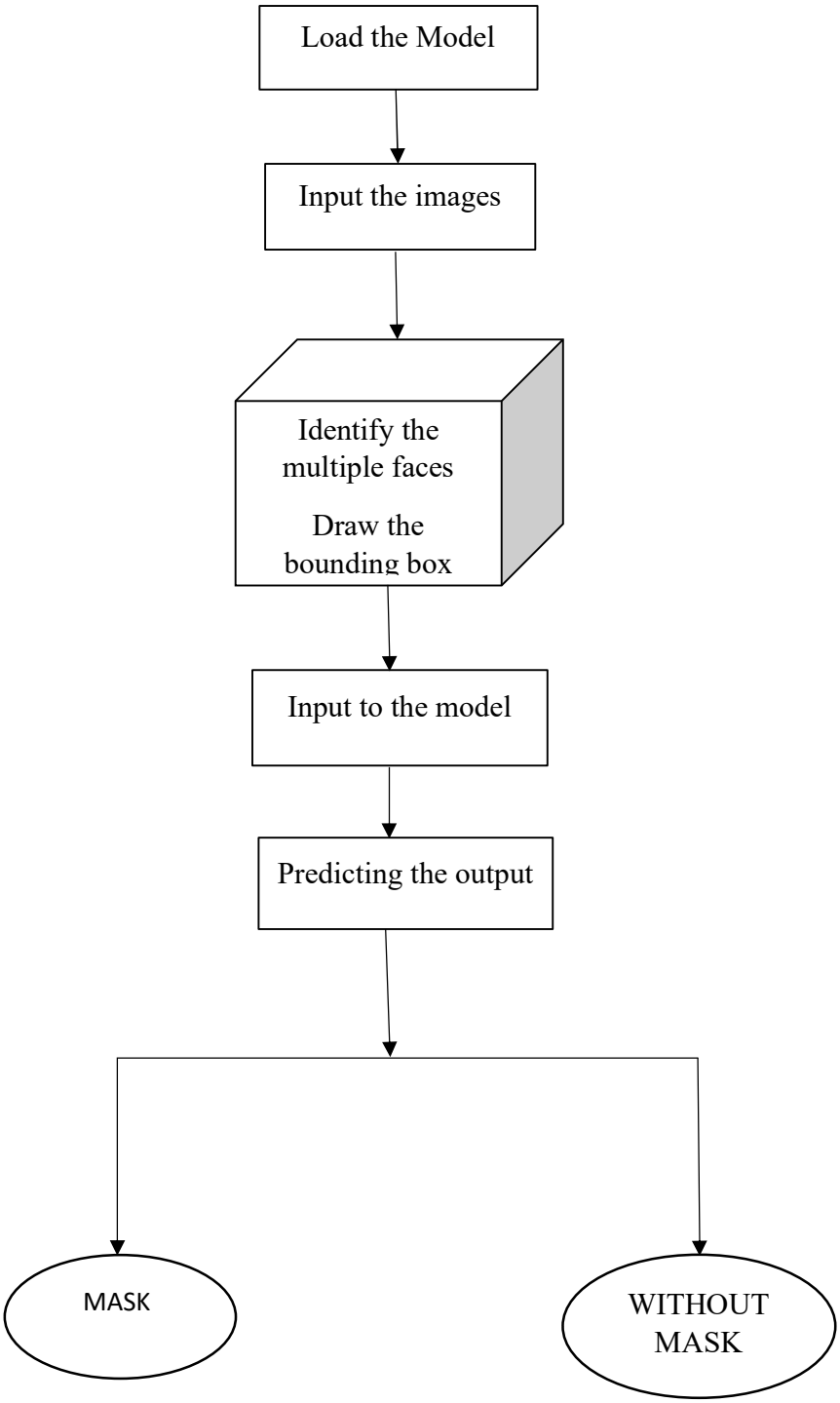

Fig 5.5: Flowchart of MTCNN

ALGORITHM:

INPUT: Image

OUTPUT: No of faces in an image

STEP1: Load the Model from the disk.

STEP2: Install MTCNN.

STEP3: Label the Classes.

STEP4: Process the images, Resizing.

STEP5: Detecting the Faces.

STEP6: Drawing a Rectangular Box around the face.

STEP7: Result.

6. RESULTS AND DISCUSSIONS

Face mask detection is implemented using CNN and MTCNN methods. Input image is given to MTCNN for detecting the multiple face. This model gives out as no of Faces in an image. This output is given as input to the CNN model. This CNN model is trained by the dataset. CNN model gives the result by classifying faces with mask and without mask. Red colour rectangular box is draw around face without mask and green rectangular box is drawn around the faces with mask.

MTCNN model:

The main usage of this MTCNN is to detect the multiple faces in the image. The faces will be identified by drawing a bounding box on the captured faces. The cropped faces will be the output from the model.

For the single image, the captured face and then the bounding box around it will be present. The below **Fig 6.1** shows the output for a single image from the model:

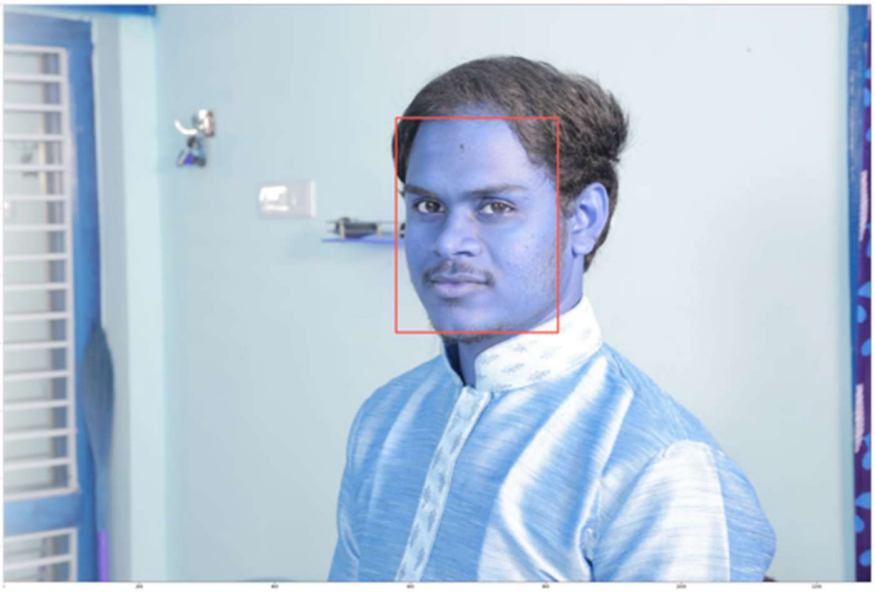

Fig 6.1: Output of MTCNN with Single face

Now, we observe the **Fig 6.2** below, with three faces in the image. Then also, it follows the same approach by identifying the three faces on the image and then bounding box will be drawn on the captured faces. The image contains three faces and the output obtained is as follows:

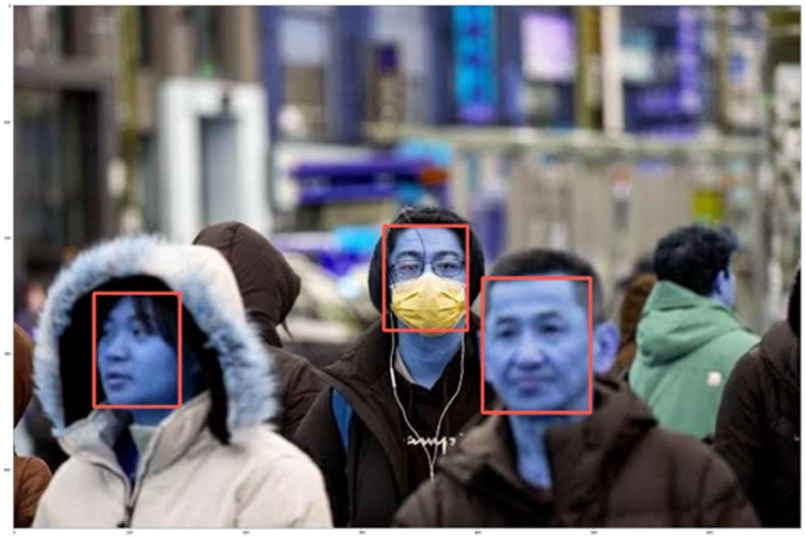

Fig 6.2: Output of MTCNN with Multiple faces

For the **Fig 6.3**, it contains multiple faces which are more than five. As mentioned earlier, it identifies all the faces from the image and then bounding box is drawn around all the faces in a group image. The following picture illustrates the output

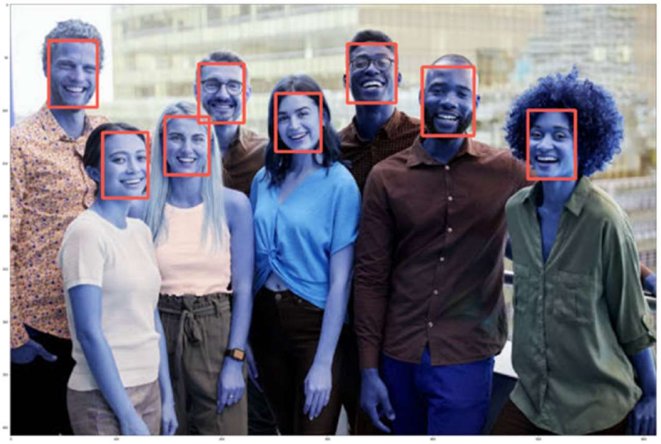

Fig 6.3 Output of MTCNN

The box will be in the form of rectangle around the face. We can use the required colour for displaying the box. Now for the identified faces we have detect whether they are wearing face or not.

CNN Model:

Here, we trained our dataset using CNN. The dataset contains images of two classes such as mask and without mask. Each class contains different number of images. After successful training, it detects the mask detection for the single person in an image. It gives the categorical data as output. The values are categorized 0 as without mask, 1 as mask. Here, as it is a binary-class classification, we convert the data into categorical form. If the value is less than 0.5 then it detects as a without mask. If the value greater than 0.5 then it detects as with mask.

Output from the CNN model:

The output is in the following **Fig 6.4** which indicates Mask.

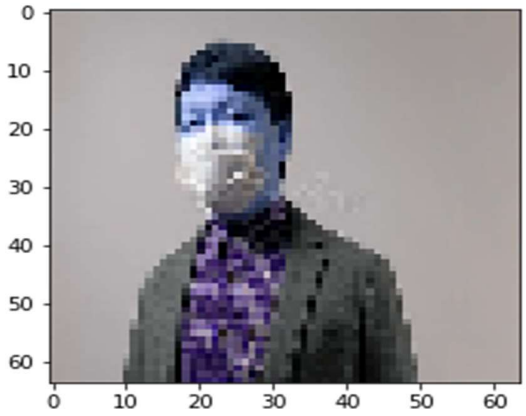

Fig 6.4: Output of CNN

Final Output

By combining above two models and final output our project, we implemented in such a way that the mask detection will be displayed on the top of the Bounding box.

The following **Fig 6.5** is for a single person after detection. It shows the detection as No Mask on the top of the face.

For Single Person in an Image

Fig 6.5: Prediction on Test Image

In the same way as previous i.e., for single image, it has been applied for multiple images. The following **Fig 6.6** shows that it identified whether the person wearing mask or no mask.

For Multiple Persons in an Image

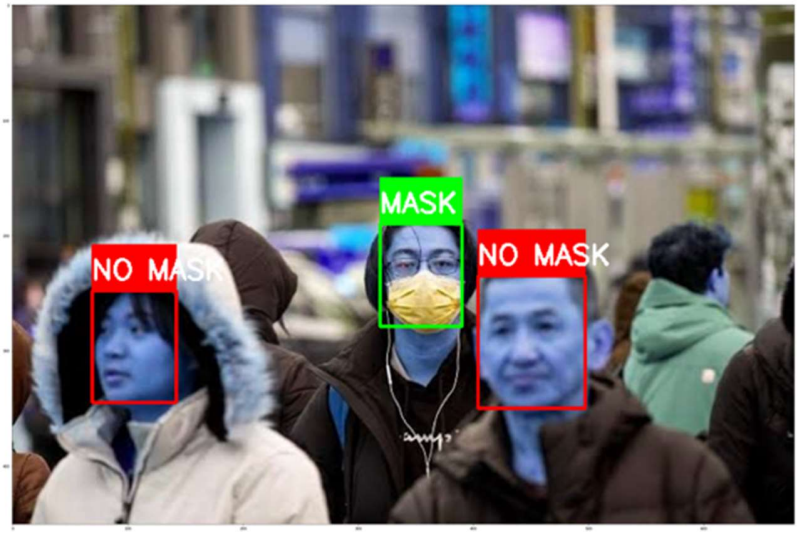

Fig 6.6: Prediction of Test image

The same can be applied to 'n' number of faces in the image. Below **Fig 6.7** illustrates for huge number of people, where mask detection is identified for each and every person in the image.

For Multiple Persons in an Image

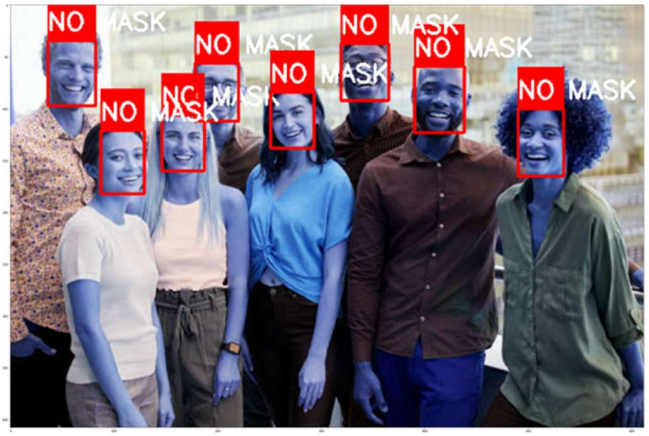

Fig 6.7: Prediction of Test image

Classification Accuracy:

Classification Accuracy is the ratio of correct predictions to the total number of predictions that is made.

classification accuracy = (correct predictions / total predictions) *100

In the same way, we can also identify the error rate. The error rate can be found by removing the classification accuracy. It is represented as follows

error rate = (1 - (correct predictions / total predictions)) * 100

If we do in the above process, we need to calculate every metric that is needed. To overcome this limitation, confusion matrix is introduced. We have used confusion matrix, to find the accuracy of the model

Accuracy graph

Below **Fig 6.8** shows the plotting of Accuracy of training and testing in each epoch. Accuracy calculated by dividing number of correct predictions by total inputs. This graph shows the accuracy calculated in each epoch. Red line represents the accuracy of testing data which indicates training accuracy is 96.8% and green line represents accuracy of training data which indicates accuracy 97.18%.

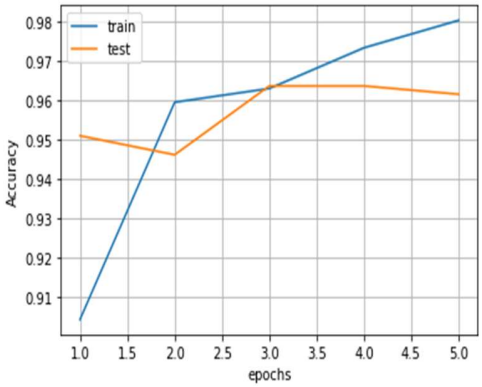

Fig 6.8: Accuracy graph

Loss Graph

Below **Fig 6.9** shows the plotting of loss of training and testing in each epoch. Loss calculated by subtraction of accuracy from 1. This graph shows the loss calculated in each epoch. Red line represents the loss of testing data which indicates training loss is 0.13 and green line represents loss of training data which indicates loss 0.7.

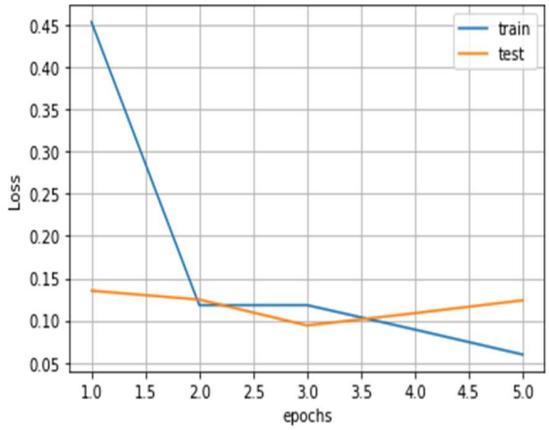

Fig 6.9: Loss graph

Confusion Matrix:

Confusion matrix is N*N matrix used to describe the performance of a classification model. Here, N represents the number of target classes. The matrix compares the actual target values with those predicted by machine learning model. A confusion matrix is a summary of prediction results on a classification data. It shows the ways in which how our classification

model is confused to make predictions. It gives the number of correct and incorrect predictions that is being made by the classification model.

Table 6.1: Confusion Matrix representation

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

Calculating a confusing matrix:

- You need a test data with expected output.
- Make predictions for the test data that is taken.

- Import the confusion matrix from sklearn package.
- Then give the predicted and expected values as parameters for confusion matrix method.
- Now print the values in the form of matrix.
- The expected output is as:
    1) The number of correct predictions for each class.
    2) The number of incorrect predictions for each class.

The confusion matrix is as follows:

Table 6.2: Confusion Matrix

	WITHOUT_MASK	MASK
WITHOUT_MASK	1605	43
MASK	25	1640

Graphical Representation:

We can also modify the confusion matrix by adding the required text and colors using seaborn package. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

In seaborn, we have method call heatmap where we can pass various number of arguments to display our data in the required format. **Heatmap** is defined as a graphical representation of data using colors to visualize the value of the matrix. Heatmaps in Seaborn can be plotted by using the seaborn. Heatmap () function. Here, we passed the data variable where the matrix is stored and the required colour as arguments. The more common values are represented in brighten colour and for less common values darker colors are preferred.
The plot will be as follows:

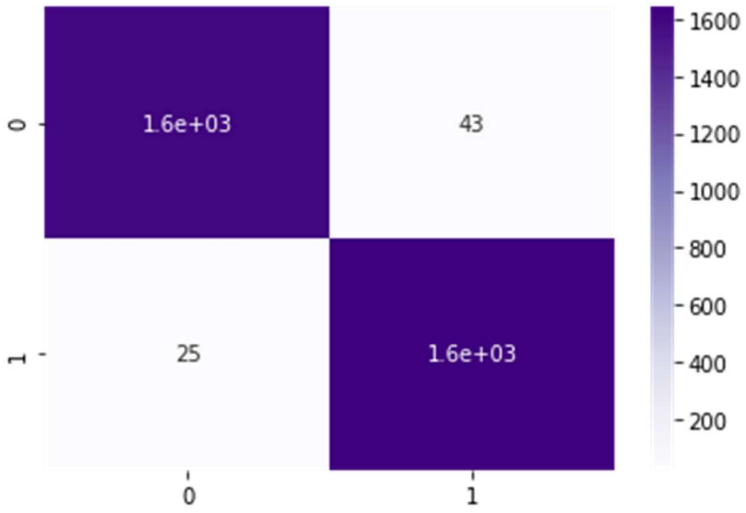

Fig 6.10: Graphical Representation of Confusion Matrix

Classification report:

A Classification report is used to measure the quality of predictions from a classification algorithm. The report shows the following classification metrics:

- accuracy
- precision
- recall and
- f1-score on a per-class basis
- support

The metrics are calculated by using true and false positives, true and false negatives.

Accuracy:

Accuracy can be defined as the correctly predicted data points out of all data points.

$$\textbf{Accuracy=(TP+TN)/(TP+FP+TN+FN)}$$

Precision:

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

$$\textbf{Precision=TP/(TP+FP)}$$

Recall:

Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

$$\textbf{Recall=TP/(TP+FN)}$$

F1-Score:

The f1-score conveys the balance between precision and recall. It is a harmonic mean of precision and recall. It gives us a combined idea of two metrics – precision and recall.

F1-score=2*precision*recall / (precision + recall)

Support:

Support is the number of actual occurrences of the class in the specified dataset.

The classification report can be imported sklearn package directly. Then it can be obtained by passing arguments as predicted output and expected output. Then the report will be printed in the dictionary format.

```
{'MASK': {'f1-score': 0.9086831143864146,
  'precision': 0.9840388619014573,
  'recall': 0.844047619047619,
  'support': 1680},
 'WITHOUT_MASK': {'f1-score': 0.9186875891583453,
  'precision': 0.8600427350427351,
  'recall': 0.9859154929577465,
  'support': 1633},
 'macro avg': {'f1-score': 0.91368535177238,
  'precision': 0.9220407984720962,
  'recall': 0.9149815560026828,
  'support': 3313},
 'weighted avg': {'f1-score': 0.9136143873422137,
  'precision': 0.9229203363474902,
  'recall': 0.913975249019016,
  'support': 3313}}
```

Fig 6.11: Report of Support

In the above **Fig 6.11**, we can observe it gives metrics i.e., f1-score, precision, recall, support for each and every class and also for macro average, weighted average. The above classification report can be represented in to a data frame which is easy to visualize and can be clearly understood.

The classification report is as follows:

Table 6.3: Classification report

	WITHOUT_MASK	MASK	macro avg	weighted avg
precision	0.860043	0.984039	0.922041	0.922920
recall	0.985915	0.844048	0.914982	0.913975
f1-score	0.918688	0.908683	0.913685	0.913614
support	1633.000000	1680.000000	3313.000000	3313.000000

In the above **Table 6.3**, we can observe that it gives us precision, recall, f1-score, support for all the classes. Along with that it also calculated neutral accuracy, macro average and weighted average.

Micro average:

Micro average is the precision/recall/f1-score calculated for all the classes.

$$\text{Micro avg Precision} = \frac{TP1 + TP2}{TP1 + TP2 + FP1 + FP2}$$

Macro average:

Macro average is the average of precision/recall/f1-score.

$$\text{Macro avg Precision} = \frac{P1 + P2}{2}$$

Weighted average :

Weighted average is just the weighted average of precision/recall/f1-score.

In the same way as for confusion matrix, we can also represent the data frame in the plot. The classification report will be as follows:

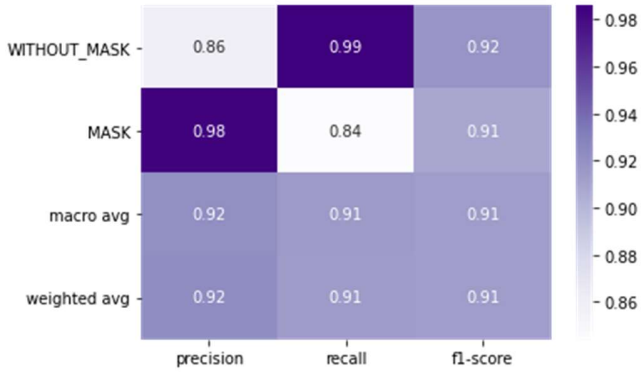

Fig 6.12: Graphical Representation of Classification Report

The training and testing accuracy is as follows:

Table 6.4: Training and Testing accuracy

	Accuracy
Training	97.42%
Testing	96.8%

From the **Table 6.4**, we can observer training accuracy that is obtained during is 97.42% and the accuracy that is obtained after passing the test data is 96.5% for our model.

From the **Table 6.5**, We compared our model with different models like Letnet-5, Alex net, Ssdmnv2, ResNet, mobileNetV2, PCA, VGG16. Accuracy of our model is high compared to these models i.e.,96.80%.

Table 6.5: Comparison of accuracy between different models.

MODEL	**ACCURACY**
CNN	**96.80%**
Letnet-5	84.6%
Alexnet	89.2%
Ssdmnv2	92.64%
ResNET	68.4%
MobileNetV2	93.12%
PCA	68.75%
VGG16	68.4%

7. CONCLUSION

Covid19 virus has been spreading rapidly across all the countries. Face mask and social distancing can prevent spreading of virus. So, we are proposing face mask detection method using datamining techniques. For this method we have taken data set from Kaggle which consists of 5521 images having face mask and 5521 images without face mask. In our method we used CNN for single face detection and MTCNN for multiple face detection. CNN is used for the mask detection. Our data set is divided in to 7:3 ratio.70%is used for training and 30% is used for testing. Data set is trained against CNN model. First, we are giving image to MTCNN model for face detection. output from this model is given as input the CNN model which classifies faces with mask and without masks. Red rectangular will show around face with mask and green box is shown around face with mask. Training accuracy of this method is 97.18 and testing accuracy is 96.80.

8. FUTURE SCOPE

A smart city means an urban area that consists of many IoT sensors to collect data. Firstly, CCTV cameras are used to capture real-time video footage of different public places in the city to find out whether a person is using a mask or not. This information is transferred through the city network to the corresponding authority to take necessary actions. The information about the violator is sent via SMS. People near to the other person not wearing a mask may be alerted by an alarm signal on that location. In further our model can be elaborated in this way.

9. APPENDIX

[1]
```
from google.colab import drive
drive.mount('/content/drive')
```

[2]
```
import os
import numpy as np
import random
from tensorflow.keras.layers import Dense,Dropout,Conv2D,MaxPooling2D,Flatten
from tensorflow.keras.models import Sequential
import cv2
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

[3]
```
pip install tensorflow
```

[4]
```
path='/content/drive/My Drive/Dataset/with_mask'
files=[]
for r,d,f in os.walk(path):
    for file in f:
        files.append(os.path.join(r, file))

path='/content/drive/My Drive/Dataset/without_mask'
for r,d,f in os.walk(path):
    for file in f:
        files.append(os.path.join(r, file))

print(len(files))
```

[5]
```
y=[1]*5521+[0]*5521
x=[]
img=cv2.imread(files[i])
res=cv2.resize(img,dsize=(64,64),interpolation=cv2.INTER_CUBIC)
image=np.array(res)/255
x.append([image,y[i]])
x=np.array(x)
```

[6]
```
np.random.shuffle(x)
np.random.shuffle(x)
x,y=np.array(list(map(lambda x: x[0],x))),np.array(list(map(lambda x: x[1],x)))
x.shape
```

[7]
```
fig, ax = plt.subplots()
ax.hist(y)
ax.set(xlim=(0, 2))
```

```
    ax.set_title('Different Classes')
    ax.set_xlabel('Classes')
    ax.set_ylabel('Frequency')
```

[8]
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

[9]
```
x_train.shape,x_test.shape
```

[10]
```
y_train.shape,y_test.shape
```

[11]
```
index=np.random.randint(11042)
plt.imshow(x[index])
y[index]
```

[12]
```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(15000,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(7000,activation='relu'))
model.add(Dense(2000,activation='relu'))
model.add(Dense(300,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

[13]
```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

[14]
```
from tensorflow.keras.callbacks import TensorBoard
import time
tb=TensorBoard(log_dir='logs/new4')
```

[15]
```
his=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,callbacks=[tb])
```

[16]
```
model.evaluate(x_test,y_test)
```

[17]
```
index=random.randrange(9211,11042)
plt.imshow(x[index].reshape(64,64,3))
res=model.predict(np.array([x[index]]))
if res[0][0]>0.5:
    print('with_mask')
else:
    print('without_mask')
res[0][0]
```

[18] model.summary()

[19] his.history.keys()

[20]
```
plt.plot(range(1,6),his.history['loss'])
plt.plot(range(1,6),his.history['val_loss'])
plt.legend(['train','test'])
plt.ylabel('Loss')
plt.xlabel('epochs')
plt.grid()
plt.show()
```

[21]
```
plt.plot(range(1,6),his.history['accuracy'])
plt.plot(range(1,6),his.history['val_accuracy'])
plt.legend(['train','test'])
plt.ylabel('Accuracy')
plt.xlabel('epochs')
plt.grid()
plt.show()
```

[22] y_pred = model.predict(x_test)

[23] y_pred = list(map(lambda x: 1 if x>=0.5 else 0,y_pred))

[24]
```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(y_test,y_pred)
sns.heatmap(conf, annot=True,cmap='Purples')
```

[25]
```
pd.DataFrame(conf,columns=['WITHOUT_MASK','MASK'],index=['WITHOUT_MASK','MASK'])
```

[26]
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,
y_pred,output_dict=True,target_names=['WITHOUT_MASK','MASK'])
clf_report
```

[26] pd.DataFrame(clf_report)

[27] sns.heatmap(pd.DataFrame(clf_report).iloc[:-1, :].T, annot=True, cmap='Purples')

[28] model.save('Face_Mask_Prediction.h5')

[29]
```
from keras.models import load_model

new_model=load_model('Face_Mask_Prediction.h5')
```

```
new_model.evaluate(x_test,y_test,verbose=2)
```

[30]
```
new_model.summary()
```

[31]
```
img = cv2.imread('/content/drive/MyDrive/Dataset/with_mask/3.jpg')
img = cv2.resize(img, (64,64))
img = np.reshape(img, (1,64,64,3))
print(img.shape)
print(new_model.predict(img))
```

[32]
```
!pip install mtcnn
```

[33]
```
from mtcnn.mtcnn import MTCNN
detector = MTCNN()
```

[34]
```
labels={0:'MASK',1:'NO MASK', 2:'NO FACE FOUND'}
color={0:(0,255,0),1:(255,0, 0)}
```

[35]
```
test_images =
['/content/drive/MyDrive/Dataset/with_mask/11.jpg','/content/drive/MyDrive/Dataset/without_mask/11.jpg','/content/drive/MyDrive/Dataset/with_mask/11.jpg']
```

[36]
```
def show_images(images):
    n: int = len(images)
    f = plt.figure(figsize=(128,128))
    for i in range(n):
      # Debug, plot figure
        f.add_subplot(1, n, i + 1)
        plt.imshow(images[i])
  plt.show(block=True)
```

[37]
```
output_images = []
for file in test_images:
    img = cv2.imread(file)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    faces = detector.detect_faces(img_rgb)
    index = 1
    try:
    for face in faces:
            (x,y,w,h) = face['box']
            face_img = img_rgb[y:y+h, x:x+w]
    # print(face_img.shape)
            img_resized = cv2.resize(face_img, (64,64))
            img_resized = np.reshape(img_resized, (1,64,64,3))

    result = int(model.predict(img_resized)[0][0])
    cv2.rectangle(img,(x,y),(x+w,y+h),color[result],2)
```

```
            cv2.rectangle(img,(x,y-40),(x+w,y),color[result],-1)
            cv2.putText(img, labels[result], (x, y-
            10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
        output_images.append(img)
    except Exception as e:
        print(e)
    index+=1

show_images(output_images)
```

10. REFERENCES

1. B. QIN and D. Li, identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19, May 2020, doi: 10.21203/rs.3.rs-28668/v1.

2. 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.

3. M.S. Ejaz, M.R. Islam, M. Sifatullah, A.Sarker Implementation of principal component analysis on masked and non-masked face recognition 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (2019), pp. 15, 10.1109/ICASERT.2019.8934543

4. C.Jagadeeswari, M.UdayTheja, "Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers", International Journal of Advanced Science and Technology Vol. 29, No. 11s, (2020), pp. 3074-3082.

5. Venkateswarlu, I. B., Kakarla, J., & Prakash, S. (2020, December). Face mask detection using MobileNet and Global Pooling Block. In 2020 IEEE 4th Conference on Information &Communication Technology (CICT) (pp. 1-5). IEEE.

6. Chavda, A., Dsouza, J., Badgujar, S., & Damani, A. (2021, April). Multi-stage CNN architecture for face mask detection. In 2021 6th International Conference for Convergence in Technology (I2CT) (pp. 1-8). IEEE.

7. Oumina, A., El Makhfi, N., & Hamdi, M. (2020, December). Control The COVID-19 Pandemic: Face Mask Detection Using Transfer Learning. In 2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS) (pp. 1-5). IEEE.

8. Rahman, M. M., Manik, M. M. H., Islam, M. M., Mahmud, S., & Kim, J. H. (2020, September). An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network. In 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS) (pp. 1-5). IEEE.

9. Sanjaya, S. A., & Rakhmawan, S. A. (2020, October). Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic. In 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) (pp. 1-5). IEEE.

10. Singh, S., Ahuja, U., Kumar, M., Kumar, K., & Sachdeva, M. (2021). Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. Multimedia Tools and Applications,1-16.

11. Sen, S., & Sawant, K. (2021, February). Face mask detection for covid_19 pandemic using pytorch in deep learning. In IOP Conference Series: Materials Science and Engineering (Vol. 1070, No. 1, p. 012061). IOP Publishing.

12. Wang, Z., Wang, P., Louis, P. C., Wheless, L. E., &Huo, Y. (2021). WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19. arXiv preprint arXiv:2101.00784.

13.Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. Sustainable cities and society, 66, 102692.