FACE MASK DETECTION SYSTEM USING DEEP LEARING

A Project report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE ENGINEERING

Submitted by

M. RAJA SEKHAR	317126510151
P. RUSHITHA	317126510159

Under the guidance of

B. SIVA JYOTHI

(ASSISTANT PROFESSOR)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Affiliated to Andhra University)

SANGIVALASA, VISAKHAPATNAM - 531162

2017-2021

ACKNOWLEDGMENT

An endeavor over a long period can be successful with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We owe our tributes to Dr. R. SIVARANJANI, Head of the Department, Computer Science & Engineering, ANITS, for her valuable support and guidance during the period of project implementation.

We wish to express our sincere thanks and gratitude to our project guides Mrs. Siva Jyothi, Assistant Professor, Department of Computer Science and Engineering, ANITS, for the simulating discussions, in analyzing problems associated with our project work and for guiding us throughout the project. Project meetings were highly informative.

We express our warm and sincere thanks for the encouragement, untiring guidance and the confidence they had shown in us. We are immensely indebted for their valuable guidance throughout our project.

We also thank all the staff members of CSE department for their valuable advices. We also thank supporting staff for providing resources as and when required.

M. Raja Sekhar Naik	317126510151
P. Rushitha	317126510159

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Affiliated to Andhra University)

SANGIVALASA, VISAKHAPATNAM - 531162

2017-2021

CERTIFICATE

This is to certify that this project report "**Face Mask Detection system using Deep Learning**" submitted by **M. Raja Sekhar Naik (317126510151), B. Manoj Varma (317126510130), P. Rushitha (317126510159)**of **IV/IV CSE** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering of Anil Neerukonda Institute of technology and sciences (A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Siva Jyothi

Mrs. B. Siva Jyothi	**Dr. R. SIVARANJANI**
ASSISTANT PROFESSOR	**HEAD OF THE DEPARTMENT**
COMPUTER SCIENCE & ENGINEERING	COMPUTER SCIENCE & ENGINEERING
ANITS	ANITS

DECLARATION

We **M. Raja Sekhar Naik (317126510151),B. ManojVarma(317126510130), P. Rushitha (317126510159)** , of final semester B.Tech.in the department of Computer Science and Engineering from ANITS, Visakhapatnam, hereby declare that the project work entitled "**Face Mask Detection system using Deep Learning**" is carried out by us and submitted in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering, Anil Neerukonda Institute of Technology & Sciences(A) during the academic year 2017-2021 and has not been submitted to any other university for the award of any kind of degree.

M. Raja Sekhar Naik	-	317126510151
B. Manoj Varma	-	317126510130
P. Rushitha	-	317126510159

ABSTRACT

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.

Keywords: COVID-19 epidemic, HAAR-CASACADE algorithm, mask detection, face mask image, non-face mask image

TABLE OF CONTENTS

LIST OF SYMBOLS

h	-	Height
w	-	Width
f	-	Filter
d	-	Dimension

LIST OF FIGURES

LIST OF ABBREVATIONS

MTCNN	-	Multi-Task Cascaded Convolutions Neural Networks
CNN	-	Convolutional Neural Network
CCTV	-	Closed-Circuit Television
MIT	-	Massachusetts Institute Technology
CNRI	-	Corporation for National Research Initiatives
LLNL	-	Lawrence Livermore National Laboratory
PyPI	-	Python Package Index
SIFT	-	Scale-Invariant Feature Transform
HOG	-	Histogram of Oriented Gradients
UML	-	Unified Modelling Language

CHAPTER 1

INTRODUCTION

1.2 Motivation of Work:

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 Covid-19 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.

1.3 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.

CHAPTER 2

LITERATURE SURVEY

2.1 An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network [1]: COVID-19 pandemic caused by novel coronavirus is continuously spreading until now all over the world. The impact of COVID-19 has been fallen on almost all sectors of development. The healthcare system is going through a crisis. Many precautionary measures have been taken to reduce the spread of this disease where wearing a mask is one of them. In this paper, we propose a system that restrict the growth of COVID-19 by finding out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras. While a person without a mask is detected, the corresponding authority is informed through the city network. A deep learning architecture is trained on a dataset that consists of images of people with and without masks collected from various sources. The trained architecture achieved 98.7% accuracy on distinguishing people with and without a facial mask for previously unseen test data. It is hoped that our study would be a useful tool to reduce the spread of this communicable disease for many countries in the world.

2.2 Masked Face Recognition Using Convolutional Neural Network [2]: Recognition from faces is a popular and significant technology in recent years. Face alterations and the presence of different masks make it too much challenging. In the real-world, when a person is uncooperative with the systems such as in video surveillance then masking is further common scenarios. For these masks, current face recognition performance degrades. An abundant number of researches work has been performed for recognizing faces under different conditions like changing pose or illumination, degraded images, etc. Still, difficulties created by masks are usually disregarded. The primary concern to this work is about facial masks, and especially to enhance the recognition accuracy of different masked faces. A feasible approach has been proposed that consists of first detecting the facial regions. The occluded 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.

2.3 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.

CHAPTER 3
METHODOLOGY

3.1 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.

3.2 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.

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.

3.3 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.

3.4 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 high-level 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 open-source 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.

3.5 MATPLOT:

Mat plot is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, WX Python, Qt, or GTK+. There is also a procedural "Pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and further joined by Thomas Caswell.

3.6 IPYTHON

What exactly is Python? You may be wondering about that. You may be referring to this book because you wish to learn editing but are not familiar with editing languages. Alternatively, you may be familiar with programming languages such as C, C ++, C #, or Java and wish to learn more about Python language and how it compares to these "big word" languages.

3.7 PYTHON CONCEPTS

You can skip to the next chapter if you are not interested in how and why Python. In this chapter, I will try to explain why I think Python is one of the best programming languages available and why it is such a great place to start.

Python was developed into an easy-to-use programming language. It uses English words instead of punctuation, and has fewer syntax than other languages. Python is a highly developed, translated, interactive, and object-oriented language.

Python translated - Interpreter processing Python during launch. Before using your software, you do not need to install it. This is similar to PERL and PHP editing languages.
Python interactive - To write your own applications, you can sit in Python Prompt and communicate directly with the interpreter.

Python Object-Oriented - Python supports the Object-Oriented program style or method, encoding the code within objects.

Python is a language for beginners - Python is an excellent language for beginners, as it allows for the creation of a variety of programs, from simple text applications to web browsers and games.

3.7.1Python Features

Python features include -

1.Easy-to-learn - Python includes a small number of keywords, precise structure, and well-defined syntax. T This allows the student to learn the language faster

2.Easy to read - Python code is clearly defined and visible to the naked eye.

3.Easy-to-maintain - Python source code is easy to maintain.

4.Standard General Library - Python's bulk library is very portable and shortcut compatible with UNIX, Windows, and Macintosh.

5.Interaction mode - Python supports interaction mode that allows interaction testing and correction of captions errors.

6.Portable - Python works on a variety of computer systems and has the same user interface for all.

7.Extensible - Low-level modules can be added to Python interpreter. These modules allow system developers to improve the efficiency of their tools either by installing or customizing them.

8.Details - All major commercial information is provided by Python ways of meeting.

9.GUI Programming - Python assists with the creation and installation of a user interface for images of various program phones, libraries, and applications, including Windows MFC, Macintosh, and Unix's X Window.

10.Scalable - Major projects benefit from Python building and support, while Shell writing is not

Aside from the characteristics stated above, Python offers a long list of useful features, some of which are described below. −

●It supports OOP as well as functional and structured programming methodologies.
●It can be used as a scripting language or compiled into Byte-code for large-scale application development.
●It allows dynamic type verification and provides very high-level dynamic data types.
●Automatic garbage pickup is supported by IT.

ADVANTAGES/BENEFITS OF PYTHON:

The diverse application of the Python language is a result of the combination of features which
give this language an edge over others. Some of the benefits of programming in Python include:

1. Presence of Third-Party Modules:

The Python Package Index (PyPI) contains numerous third-party modules that make Python
capable of interacting with most of the other languages and platforms.

2. Extensive Support Libraries:

Python provides a large standard library which includes areas like internet protocols, string
operations, web services tools and operating system interfaces. Many high use programming tasks
have already been scripted into the standard library which reduces length of code to be written
significantly.

3. Open Source and Community Development:

Python language is developed under an OSI-approved opensource license, which makes it free
to use and distribute, including for commercial purposes. Further, its development is driven by the
community which collaborates for its code through hosting conferences and mailing lists, and
provides for its numerous modules.

4. Learning Ease and Support Available:

Python offers excellent readability and uncluttered simple-to-learn syntax which helps
beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of
rules to facilitate the formatting of code. Additionally, the wide base of users and active developers
has resulted in a rich internet resource bank to encourage development and the continued adoption of
the language.

5. User-friendly Data Structures:

Python has built-in list and dictionary data structures which can be used to construct fast
runtime data structures. Further, Python also provides the option of dynamic high-level data typing
which reduces the length of support code that is needed.

6. Productivity and Speed:

Python has Clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications.

As can be seen from the above-mentioned points, Python offers a number of advantages for
software development. As upgrading of the language continues, its loyalist base could grow as well.

Python has five standard data types −

●Numbers
●String
●List
●Tuple
●Dictionary

3.7.2 Python Numbers

Numeric values are stored in number data types. When you give a number a value, it becomes a number object.

3.7.3 Python Strings

In this python uses a string is defined as a collection set of characters enclosed in quotation marks. Python allows you to use any number of quotes in pairs. The slice operator ([] and [:]) can be used to extract subsets of strings, with indexes starting at 0 at the start of the string and working their way to -1 at the end.

3.7.4 Python Lists

The most diverse types of Python data are lists. Items are separated by commas and placed in square brackets in the list ([]). Lists are similar to C-order in some ways. Listings can be for many types of data, which is one difference between them.

The slide operator ([] and [:]) can be used to retrieve values stored in the list, the indicators start with 0 at the beginning of the list and work their way to the end of the list. The concatenation operator of the list is a plus sign (+), while the repeater is an asterisk (*).

3.7.5 Python Tuples

A cone is a type of data type similar to a sequence of items. Cone is a set of values separated by commas. The pods, unlike the list, are surrounded by parentheses.

Lists are placed in parentheses ([]), and the elements and sizes can be changed, but the lumps are wrapped in brackets (()) and cannot be sorted. Powders are the same as reading lists only.

3.7.6 Python Dictionary

Python dictionaries in Python are a way of a hash table. They are similar to Perl's combination schemes or hashes, and are made up of two key numbers.

The dictionary key can be any type of Python, but numbers and strings are very common. Prices, on the other hand, can be anything you choose Python.

Curly braces () surround dictionaries, and square braces ([) are used to assign and access values.

Different modes in python

Python Normal and interactive are the two basic Python modes.

The scripted and completed.py files are executed in the Python interpreter in the regular manner.

Interactive mode is a command line shell that provides instant response for each statement while simultaneously running previously provided statements in active memory.

The feed programme is assessed in part and whole as fresh lines are fed into the interpreter.

3.8 PANDAS

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

Pandas is a python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real-world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible opensource data analysis/manipulation tool available in any language**. It is already well on its way toward this goal.

Explore data analysis with Python. Pandas Data Frames make manipulating your data easy, from selecting or replacing columns and indices to reshaping your data.

Pandas is well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet

Ordered and unordered (not necessarily fixed-frequency) time series data.

Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels

Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a Pandas data structure.

3.9 KERAS

KERAS is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages.

It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

FOUR PRINCIPLES:

• Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

• Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.

• Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

• Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

3.10 Machine Learning approaches:

1.Viola–Jones object detection framework based on HAAR Features
2.Scale-invariant feature transform (SIFT)
3.Histogram of oriented gradients (HOG) features

Machine learning (ML)is the study of computer algorithms that improve automatically through experience. Itis seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Approaches for Machine Learning:

3.10.1 Viola-Jones Object detection framework based in HAAR features:

The Viola-Jones algorithm is one of the most popular algorithms for objects recognition in an image. This research paper deals with the possibilities of parametric optimization of the Viola-Jones algorithm to achieve maximum efficiency of the algorithm in specific environmental conditions. It is shown that with the use of additional modifications it is possible to increase the speed of the algorithm in a particular image by 2-5 times with the loss of accuracy and completeness of the work by not more than the 3-5%.

3.10.2 Scale-invariant feature transform (SIFT):

The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images. SIFT key points of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of key points that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform . Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded, Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

3.10.3 Histogram of oriented gradients (HOG):

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object-detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a

An algorithm is involved in this proposed system

HAAR Feature-Based Cascade Classifiers

3.11 HAAR Feature-Based Cascade Classifiers

It is an Object Detection **Algorithm** used to identify faces in an image or a real time video.

Dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

It is an effective way for object detection.

In this approach, lot of positive and negative images are used to train the classifier.

In this, a model is pre-trained with frontal features is developed and used in this experiment to detect the
faces in real-time.

HAAR Cascade is a machine learning-based approach where a lot of positive and negative images are use to train the classifier.

Positive Images: These images contain the images which we want our classifier to identify.

Negative Images: Images of everything else, which do not contain the object we want to detect.

WHY HAAR FEATURE BASED CASCADE CLASSIFIERS IS PREFFERD?

The key advantage of a HAAR-like feature over most other features is its calculation speed.

A HAAR-like feature of any size can be calculated in constant time (approximately 60 microprocessor instructions).

3.12 DEEP LEARNING

1.Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.

2.Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.

3.In this, face mask detection is built using Deep Learning technique called as Convolution Neural Networks (CNN).

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower-level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

3.13 NEURAL NETWORKS VERSUS CONVENTIONAL COMPUTERS:

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem-solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don&#39;t exactly know how to do. Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem.

Neural networks learn by example. They cannot be programmed to is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high-level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

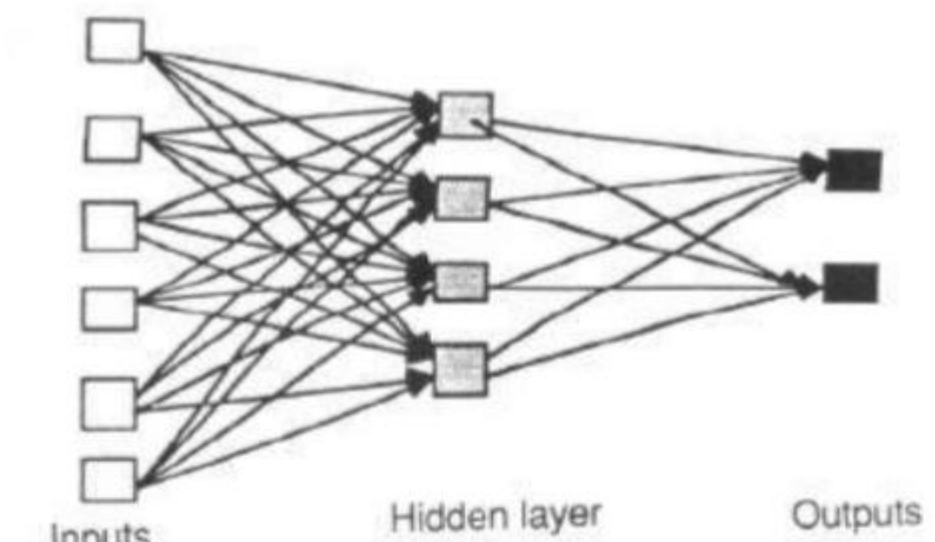
3.14 ARCHITECTURE OF NEURAL NETWORKS:

3.14.1 FEED-FORWARD NETWORKS:

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down. to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

3.14.2 FEEDBACK NETWORKS:

Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organization



Layers in NN

3.14.3 NETWORK LAYERS:

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of input units is connected to a layer of hidden units, which is connected to a layer of output units.
The activity of the input units represents the raw information that is fed into the network.
The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.
This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

Also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

3.15 Convolution Neural Network

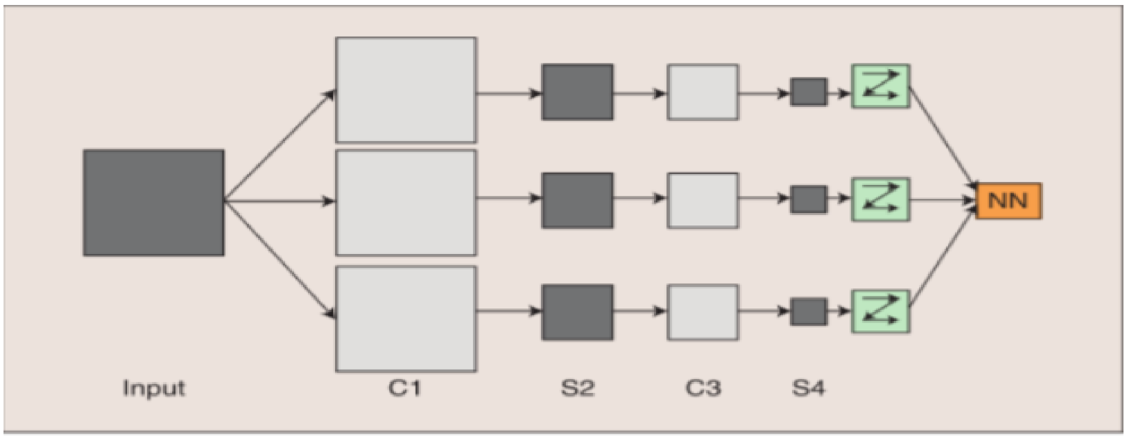
A convolution neural network is a special architecture of artificial neural network proposed by yann lecun in 1988. One of the most popular uses of the architecture is image classification. CNNs have wide applications in image and video recognition, recommender systems and natural language processing. In this article, the example that this project will take is related to Computer Vision. However, the basic concept remains the same and can be applied to any other use-case!

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs. In more detail the image is passed through a series of convolution, nonlinear, pooling layers and fully connected layers, then generates the output.

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.

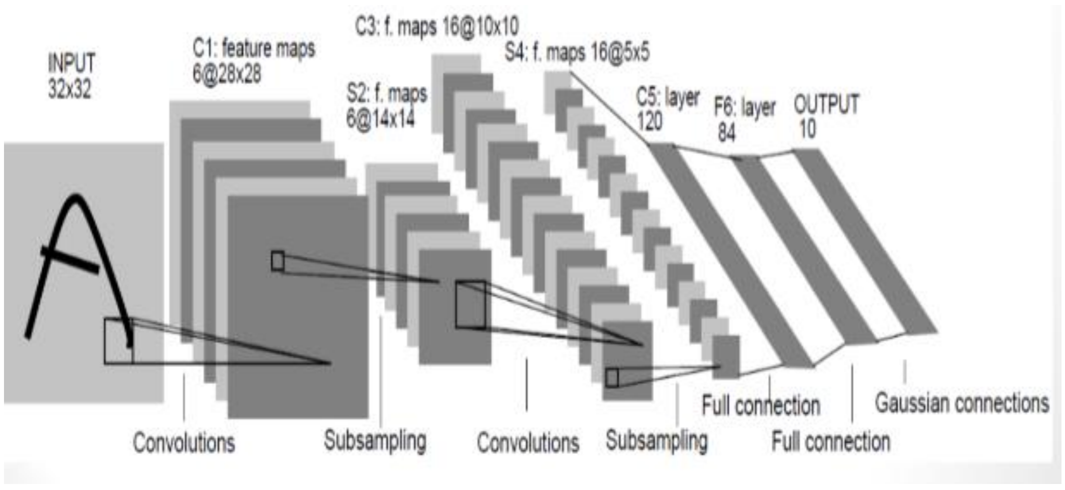
Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the visual cortex. CNNs use relatively little pre-processing compared to other image classification algorithms. CNN is a special kind of multi- layer NNs applied to 2-d arrays (usually images), based on spatially localized neural input. CNN Generate 'patterns of patterns' for pattern recognition.

Each layer combines patches from previous layers. Convolutional Networks are trainable multistage architectures composed of multiple stages Input and output of each stage are sets of arrays called feature maps. At output, each feature map represents a particular feature extracted at all locations on input. Each stage is composed of: a filter bank layer, a non-linearity layer, and a feature pooling layer. A ConvNet is composed of 1, 2 or 3 such 3-layer stages, followed by a classification module.



Basic structure of CNN, where C1, C3 are convolution layers and S2, S4 are pooled/sampled layers.

Filter: A trainable filter (kernel) in filter bank connects input feature map to output feature map Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.



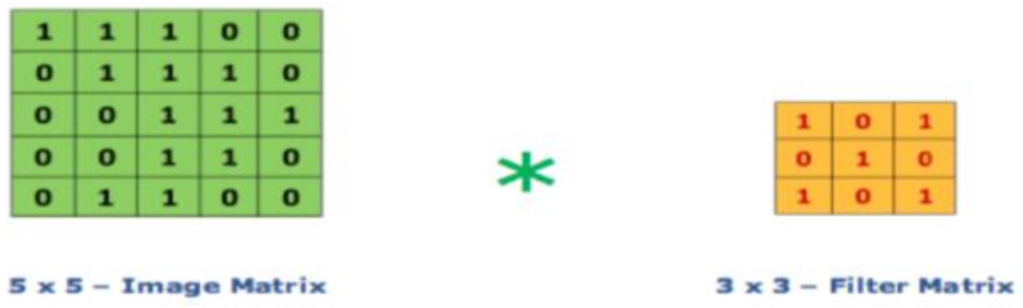
3.16 CONVOLUTIONAL LAYER

It is always first. The image (matrix with pixel values) is entered into it. Image that the reacting of the input matrix begins at the top left of image. Next the software selects the smaller matrix there, which is called a filter. Then the filter produces convolution that is moves along the input image. The filter task is to multiple its value by the original pixel values. All these multiplications are summed up and one number is obtained at the end. Since the filter has read the image only in the upper left corner it moves further by one unit right performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller than a input matrix.

This operation, from a human perspective is analogous to identifying boundaries and simple Colors on the image. But in order to recognize the fish whole network is needed. The network will be -consists of several convolution layers mixed with nonlinear and pooling layers. Convolution is the first layer to extract features from an input image. Convolution features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

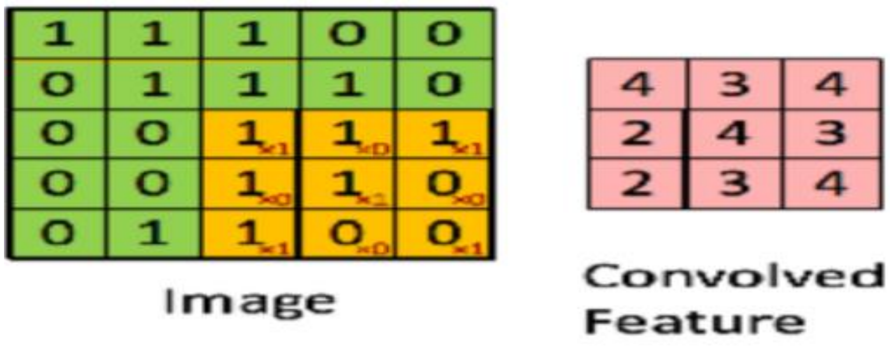
- An image matrix of dimension (h x w x d)
- A filter (fh x fw x d)
- Outputs a volume dimension (h-fh+1) x (w-fw+1) x1.

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below



convolution with a filter example

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called "Feature Map" as output shown in below



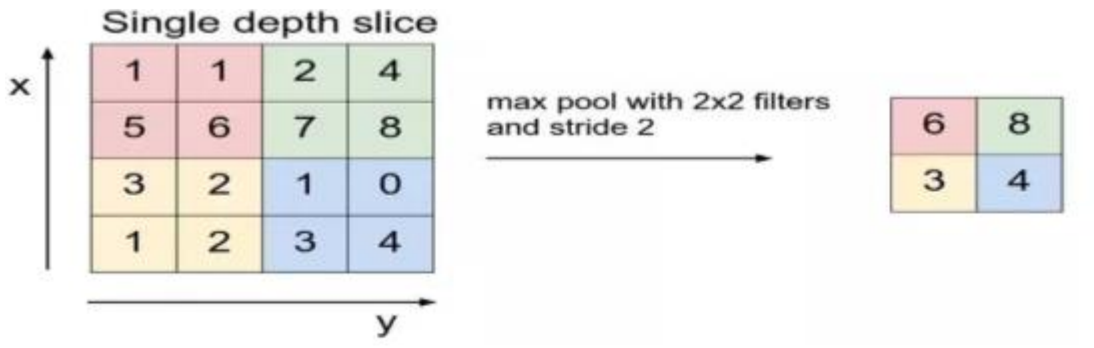
Output of Convolution layer

3.17 THE NON-LINEAR LAYER:

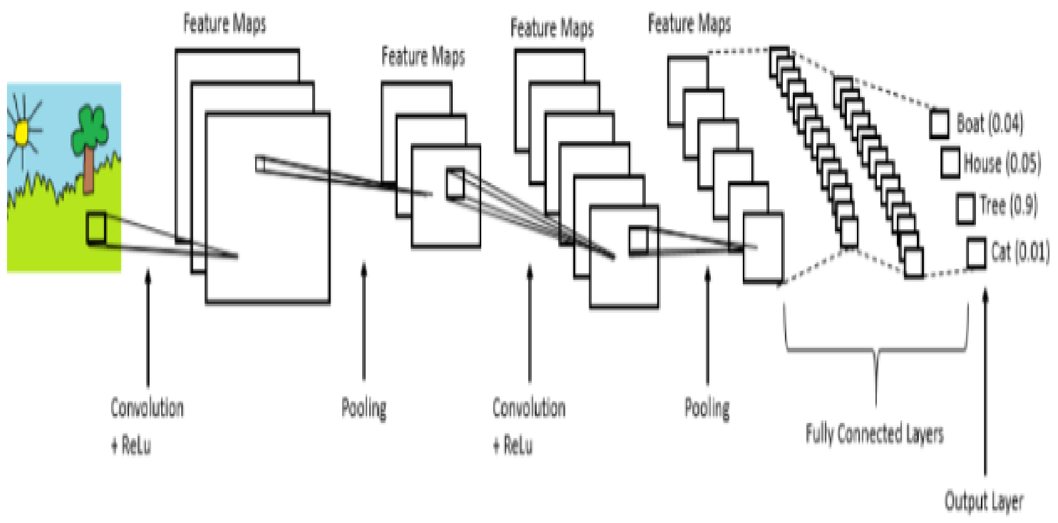
It is adder after each convolution operation. It has the activation function, which brings nonlinear property, without this property a network would not be, sufficiently intense and will not be able to model the response variable.

3.18 THE POOLING LAYER:

It follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result image volume is reduced. This means that if some features already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing and is compressed to less detailed pictures.



Max Pooling Layer



Overall structure of CNN

3.19 FULLY CONNECTED LAYER:

After completion of series of convolution, non-linear and pooling layer, it’s necessary to attach a fully connected layer. This layer takes the output information from convolution network. Attaching a fully connected layer to the end of the network results in N dimensional vector, where N is the amount of classes from which the model selects the desired class.

3.20 CNN MODEL

1.This CNN model is built using the Tensorflow framework and the OpenCv library which is highly used for real-time applications.

2.This model can also be used to develop a full-fledged software to scan every person before they can enter the public gathering.

3.20.1 LAYERS IN CNN MODEL

Conv2D
MaxPooling2D
Flatten ()
Dropout
Dense

1.Conv2D Layer:
It has 100 filters and the activation function used is the 'ReLu'. The ReLu function stands for Rectified Linear Unit which will output the input directly if it is positive ,otherwise it will output zero.

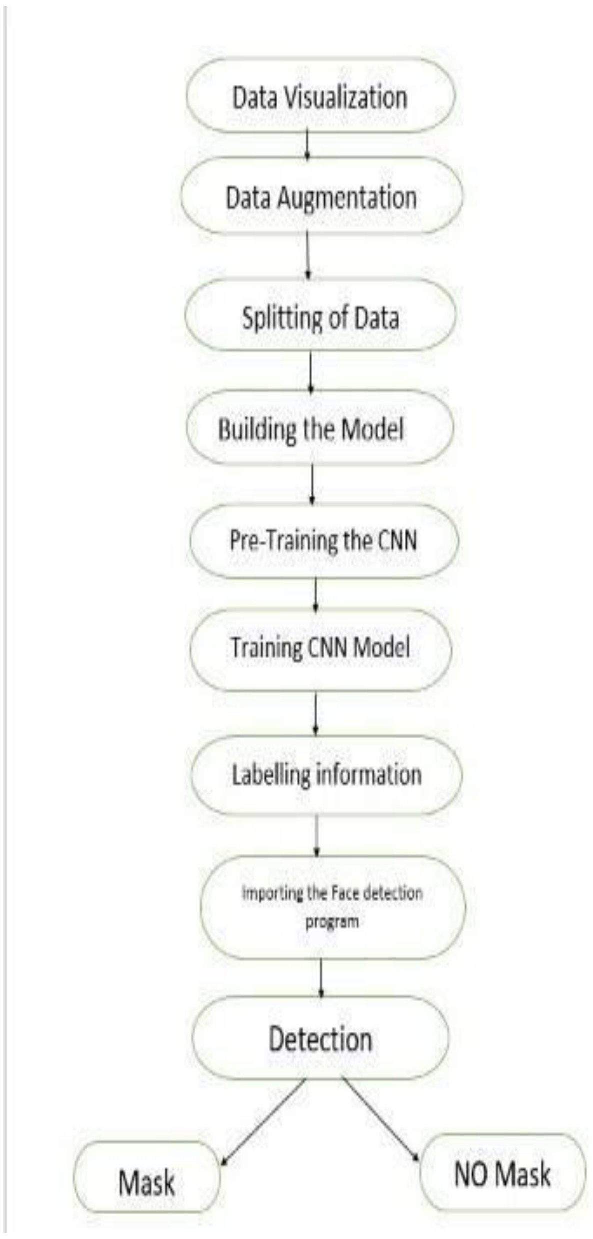
2.MaxPooling2D:
It is used with pool size or filter size of 2*2.

3.Flatten () Layer:
It is used to flatten all the layers into a single 1D layer.

4.Dropout Layer:
It is used to prevent the model from overfitting.

5.Dense Layer:
The activation function here is soft max which will output a vector with two probability distribution values.

3.21 SYSTEM ARCHITECTURE



Data Visualization.

Data Augmentation.

Splitting the data.

Labeling the Information.

Importing the Face detection.

Detecting the Faces with and without Masks.

Data Visualization

In the first step, let us visualize the total number of images in our dataset in both categories. We can see that there are ***690*** images in the '***yes***' class and ***686*** images in the '***no***' class.

Data Augmentation

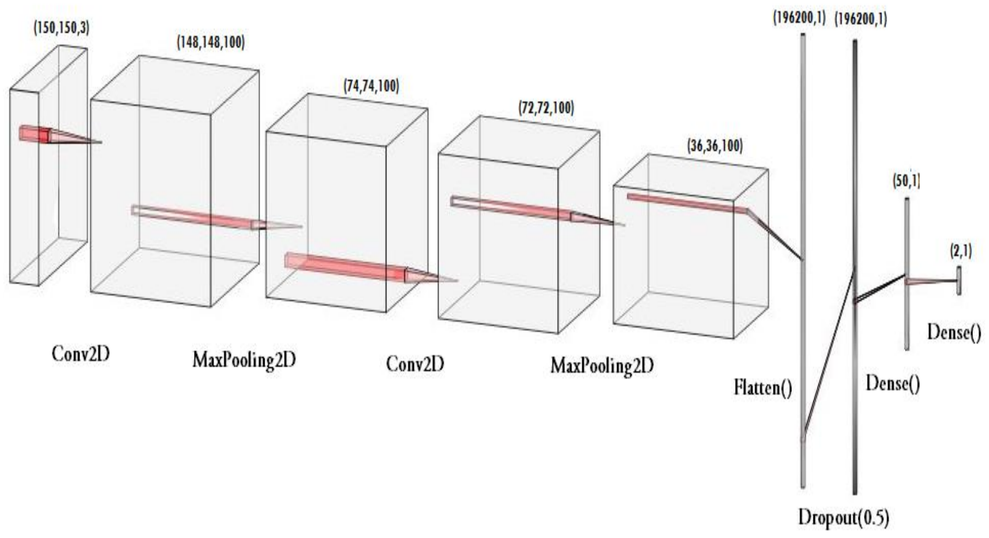
In the next step, we ***augment*** our dataset to include more number of images for our training. In this step of *data augmentation*, we *rotate* and *flip* each of the images in our dataset.

Splitting the data

In this step, we ***split*** our data into the ***training set*** which will contain the images on which the CNN model will be trained and the ***test set*** with the images on which our model will be tested.

Building the Model

In the next step, we build our ***Sequential CNN model*** with various layers such as *Conv2D, MaxPooling2D, Flatten, Dropout* and *Dense*.



Pre-Training the CNN model

After building our model, let us create the '***train_generator***' and '***validation_generator***' to fit them to our model in the next step.

Training the CNN model

This step is the main step where we fit our images in the training set and the test set to our Sequential model we built using ***keras*** library. I have trained the model for ***30 epochs*** (iterations). However, we can train for more number of epochs to attain higher accuracy lest there occurs *over-fitting*.

Labeling the Information

After building the model, we label two probabilities for our results. ***['0'** as **'without_ ask'** and **'1'** as **'with_ mask']***. I am also setting the boundary rectangle color using the RGB values.

Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the *Haar Feature-based Cascade Classifiers* for detecting the features of the face.

Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the *Cascade Classifier*.

CHAPTER 4

DESIGN

4.1 UML Diagrams:

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

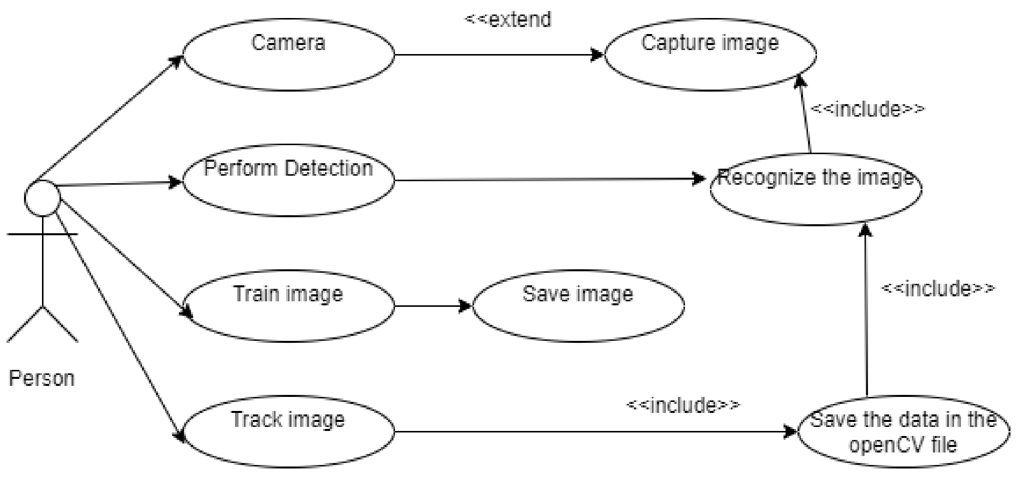
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

4.1.1 Use Case 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

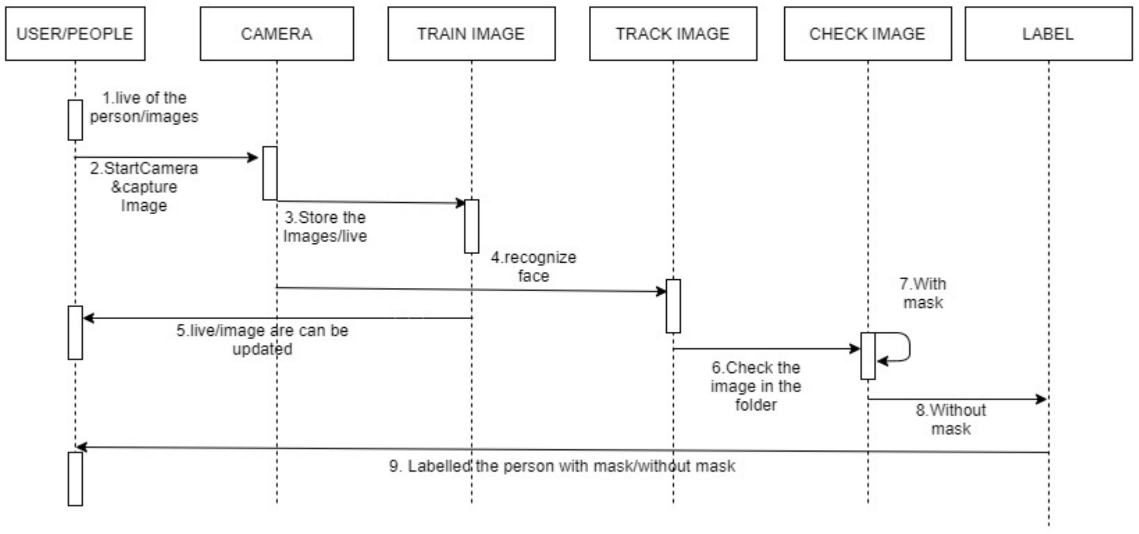


4.1.2 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

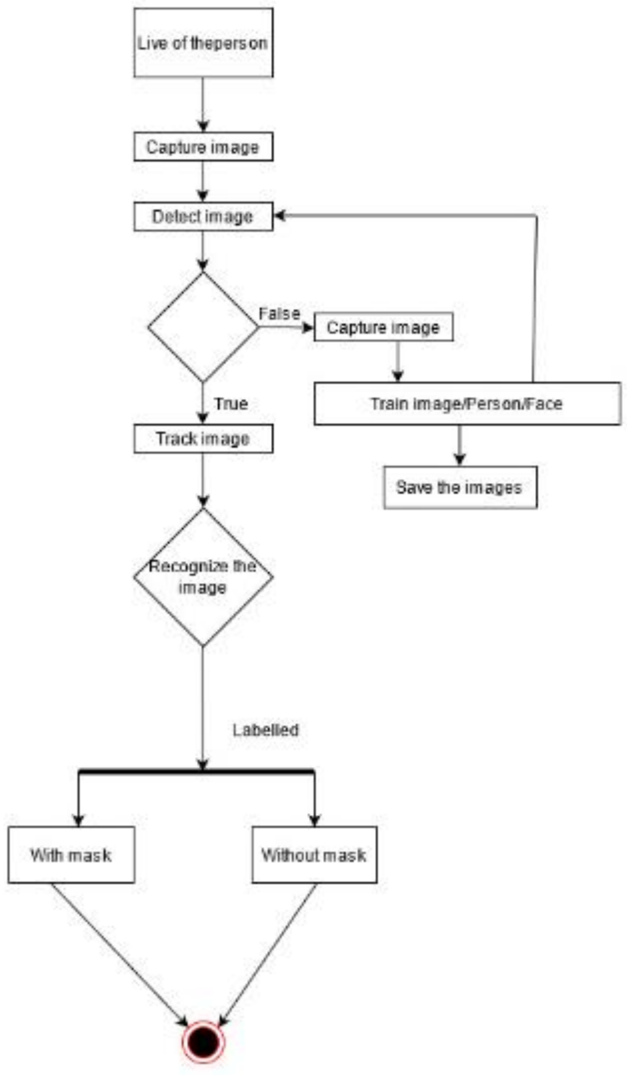
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.


USER/PEOPLE
CAMERA
TRAIN IMAGE
TRACK IMAGE
CHECK IMAGE
LABEL
1.live of the person/images
2.StartCamera &capture Image
3.Store the Images/live
4.recognize face
5.live/image are can be updated
6.Check the image in the folder
7.With mask
8.Without mask
9. Labelled the person with mask/without mask

4.1.3 Activity Diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system.

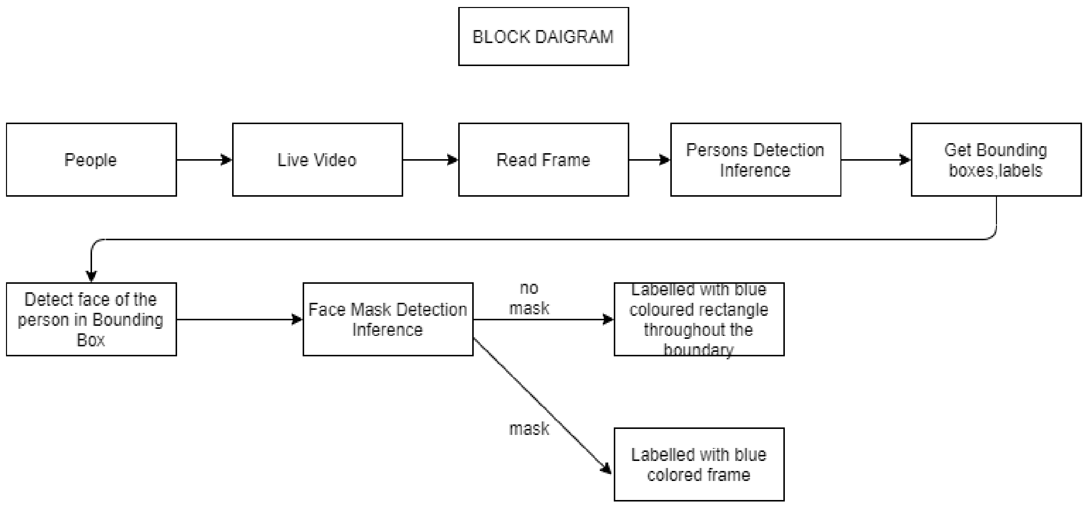
An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.



4.1.4 BLOCK DIAGRAM

A block diagram is a graphical representation of a system – it provides a functional view of a system. Block diagrams give us a better understanding of a system's functions and help create interconnections within it.

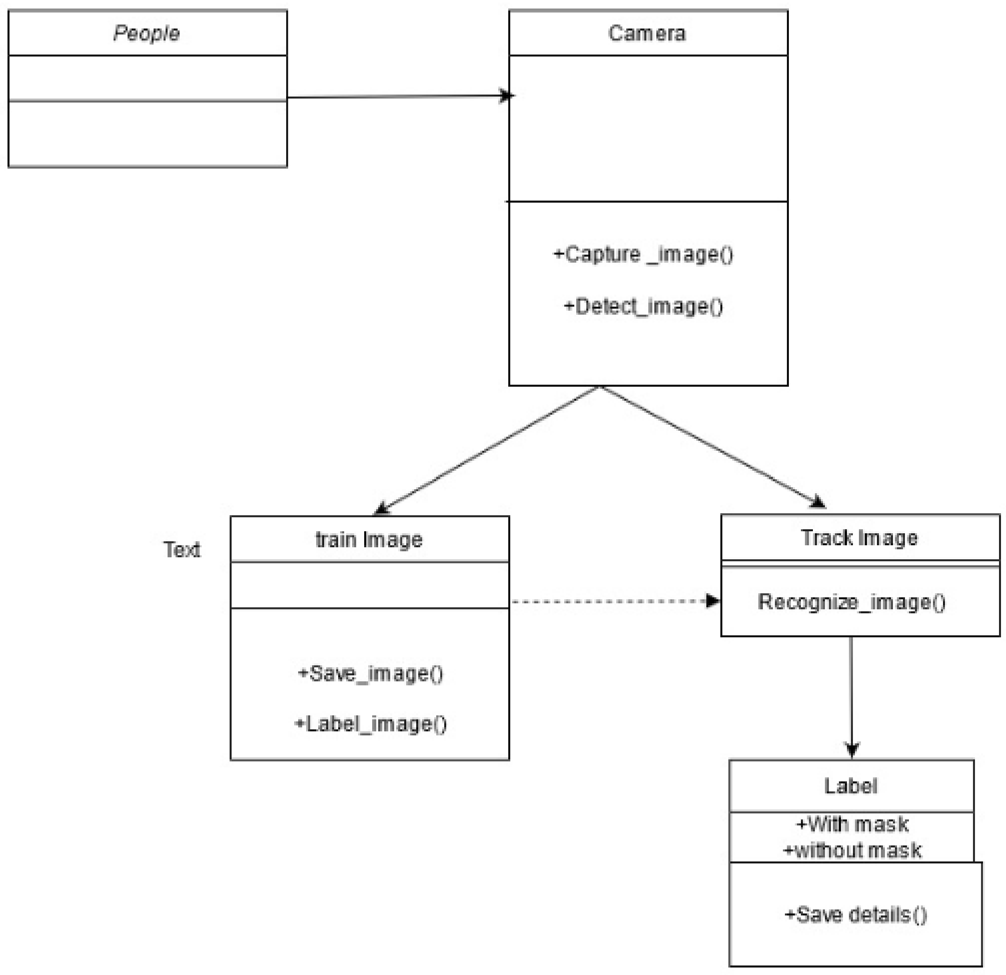
They are used to describe hardware and software systems as well as to represent processes.


BLOCK DAIGRAM
People
Live Video
Read Frame
Persons Detection Inference
Get Bounding boxes,labels
Detect face of the person in Bounding Box
Face Mask Detection Inference
no mask
Labelled with blue coloured rectangle throughout the boundary
mask
Labelled with blue colored frame

4.1.5 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application.

Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.



4.1.6 DATA FLOW DIAGRAM

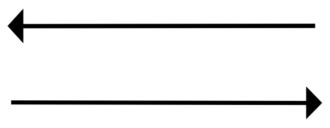
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

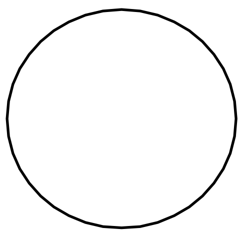
A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays. Data Flow Data is a key and basic tool for the architecture of all other objects. Bubble-bubble or data flow graph is another name for DFD.

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:

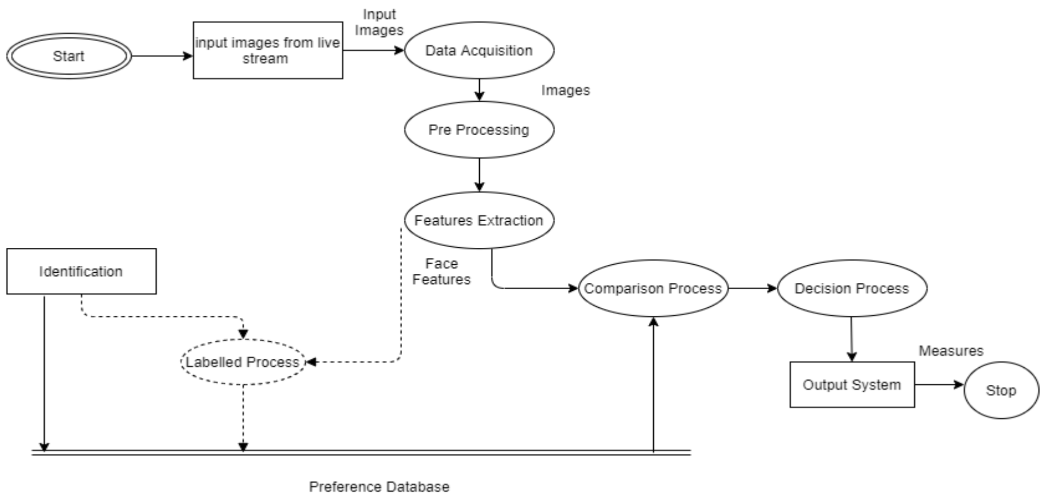
1. **Dataflow:**Data flows in a certain direction from the source to the destination.



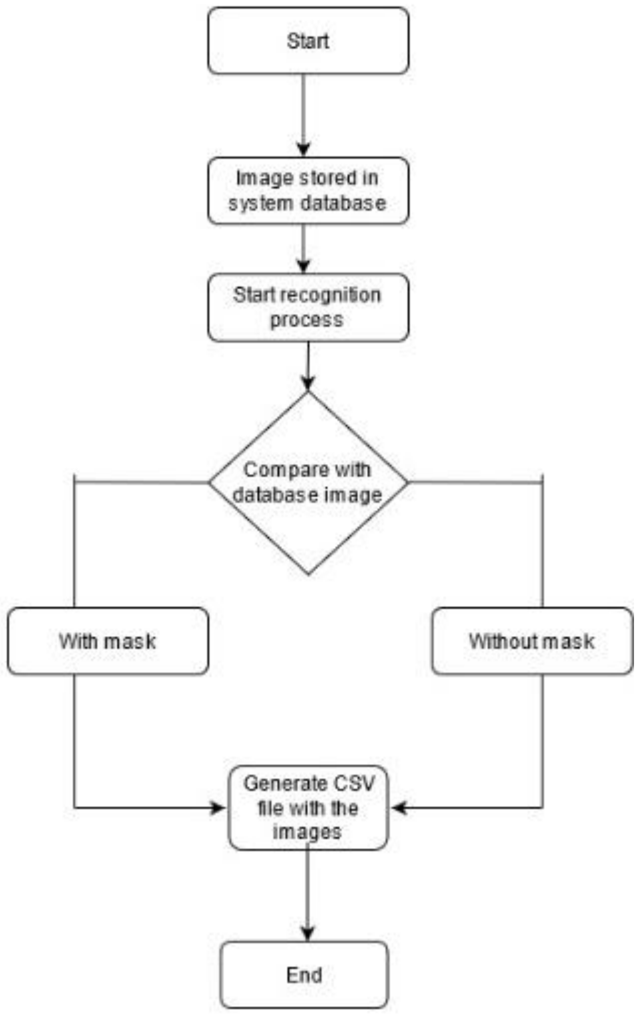
2.Process: People, processes, or technologies that use or produce(Transforming) Information No information about a body part.



3.Source: People, programs, organizations, and other things can be external data sources or locations.


Start
input images from live stream
Input Images
Data Acquisition
Images
Pre Processing
Features Extraction
Identification
Face Features
Comparison Process
Decision Process
Labelled Process
Measures
Output System
Stop
Preference Database

4.1.7 FLOWCHART DIAGRAM



Start
Image stored in system database
Start recognition process
Compare with database image
With mask
Without mask
Generate CSV file with the images
End

CHAPTER 5
EXPERIMENT ANALYSIS

5.1MODULES

1.Creating image datasets a data-loaders for train and test using the experiments folder split

- Training Dataset: A dataset that we feed into our algorithm to train our model.
- Testing Dataset: A dataset that we use to validate the accuracy of our model but is not used to train the model. It may be called the validation dataset.

2.Training the model

3. Visualizing images

5.2 DATASET LINK

https://github.com/prajnasb/observations/tree/master/experiements/data

5.3 CODE IMPLEMENTATION

5.3.1 Creating image datasets an data loaders for train and test using the experiments folder split

```
experiments_path = 'observations/experiements/dest_folder/'
data_path = 'observations/experiements/data/'
data_transforms = {
'train': transforms.Compose([
 transforms.RandomResizedCrop(224),
 transforms.ToTensor(),
 transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
 ]),
 'test' : transforms.Compose([
 transforms.Resize(256),
 transforms.CenterCrop(224),
 transforms.ToTensor(),
 transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
 ])
}
def get_train_files_path(experiments_path, data_path, phase):
 if phase == 'train':
 file_name = 'train.csv'
 elif phase == 'test':
 file_name = 'test.csv'
 else:
 print("phase can only have train and test as parameter values")
 exit()
 file_path = os.path.join(experiments_path, file_name)
 train_df = pd.read_csv(file_path, delimiter=',')
  files_path = []
  fonts_class = []
for row in train_df.iterrows():
files_path.append(os.path.join(data_path, row[1]['class'], row[1]['filename']))
fonts_class.append(row[1]['class'])

return files_path, fonts_class
def copy_images_to_path(file_path, file_class, destination_dir):
font_folder = os.path.join(destination_dir, file_class)
if os.path.exists(font_folder) == False:
os.makedirs(font_folder)

print("File being copied from {}:{}".format(file_path, font_folder))
shutil.copy(file_path, font_folder)
#shutil.copyfile(file_path, font_folder)

X_train, y_train = get_train_files_path(experiments_path, data_path, phase='train')
X_test, y_test = get_train_files_path(experiments_path, data_path, phase='test')
train_dir = os.path.join(experiments_path, 'train')
test_dir = os.path.join(experiments_path, 'test')

if not os.path.exists(train_dir):
 os.makedirs(train_dir)

if not os.path.exists(test_dir):
```

```
os.makedirs(test_dir)

image_datasets = {x: datasets.ImageFolder(os.path.join(experiments_path, x), data_tra
nsforms[x]) for x in ['train', 'test']}

image_datasets['train']

image_datasets['test']

dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x],batch_size=16,shuffle
=True,num_workers=4)

for x in ['train', 'test']}

dataloaders

class_names = image_datasets['train'].classes

class_names

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

device

dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'test']}
```

5.3.2 Training the model

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=20):

since = time.time()

best_acc = 0.0

best_model = copy.deepcopy(model.state_dict())


new_freeze_state = None

prev_freeze_state = False

for epoch in range(num_epochs):

print("Epoch {}/{}".format(epoch, num_epochs - 1))
print('-' * 10)

for phase in ['train', 'test']:




if phase == 'train':
scheduler.step()
model.train()
else:
model.eval()

running_loss = 0.0
running_corrects = 0

for inputs, labels in dataloaders[phase]:
inputs = inputs.to(device)
labels = labels.to(device)
optimizer.zero_grad()

with torch.set_grad_enabled(phase == 'train'):
 outputs = model(inputs)
  _, preds = torch.max(outputs, 1)
  loss = criterion(outputs, labels)

    if phase == 'train':
    loss.backward()
    optimizer.step()
 running_loss += loss.item() * inputs.size(0)
 running_corrects += torch.sum(preds == labels.data)

 epoch_loss = running_loss / dataset_sizes[phase]
 epoch_acc = running_corrects.double() / dataset_sizes[phase]




print('{} Loss: {:.4f} Acc:{:.4f}'.format(
   phase, epoch_loss, epoch_acc))

 if phase == 'test' and epoch_acc > best_acc:
 best_acc = epoch_acc
 best_model = copy.deepcopy(model.state_dict())

 print()

 time_elapsed = time.time() - since
 print('Training complete in {:0f}m {:.0f}s'.format(
 time_elapsed // 60, time_elapsed % 60))
 print('Best val acc: {:4f}'.format(best_acc))

  model.load_state_dict(best_model)
  return model

  import ssl
  ssl._create_default_https_context = ssl._create_unverified_context
  model_ft = models.resnet101(pretrained=True)
```

```
    num_frts = model_ft.fc.in_features
    model_ft.fc = nn.Linear(num_frts, len(class_names))

    model_ft = model_ft.to(device)
    criterion = nn.CrossEntropyLoss()

#optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.01, momentum=0.9)
  optimizer_ft = optim.Adagrad(model_ft.parameters(), lr=0.001)
  exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)

model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler, num_epoc
hs=20)

def visualize_model(model, num_images=6):
was_training = model.training
model.eval()
images_so_far = 0
#fig = plt.figure(figsize=(10,10))

with torch.no_grad():
for i, (inputs, labels) in enumerate(dataloaders['test']):
inputs = inputs.to(device)
labels = labels.to(device)

outputs = model(inputs)
_, preds = torch.max(outputs, 1)
```

```
print(preds,"predicitons")

for j in range(inputs.size()[0]):
images_so_far +=1

#ax = plt.subplot(num_images//len(labels)-1, len(labels), images_so_far)
#ax.axis('off')

#ax.set_title('true: {} predicted: {}'.format(class_names[labels[j]], class_names[preds[
j]]))

print('\n\n\ntrue: {} predicted: {}'.format(class_names[labels[j]], class_names[preds[j]
]))
imshow(inputs.cpu().data[j])

if images_so_far == num_images:
model.train(mode=was_training)
return
model.train(mode=was_training)

visualize_model(model_ft)
```

Visualizing images

```
def imshow(inp, title=None):

inp = inp.numpy().transpose((1, 2, 0))
```

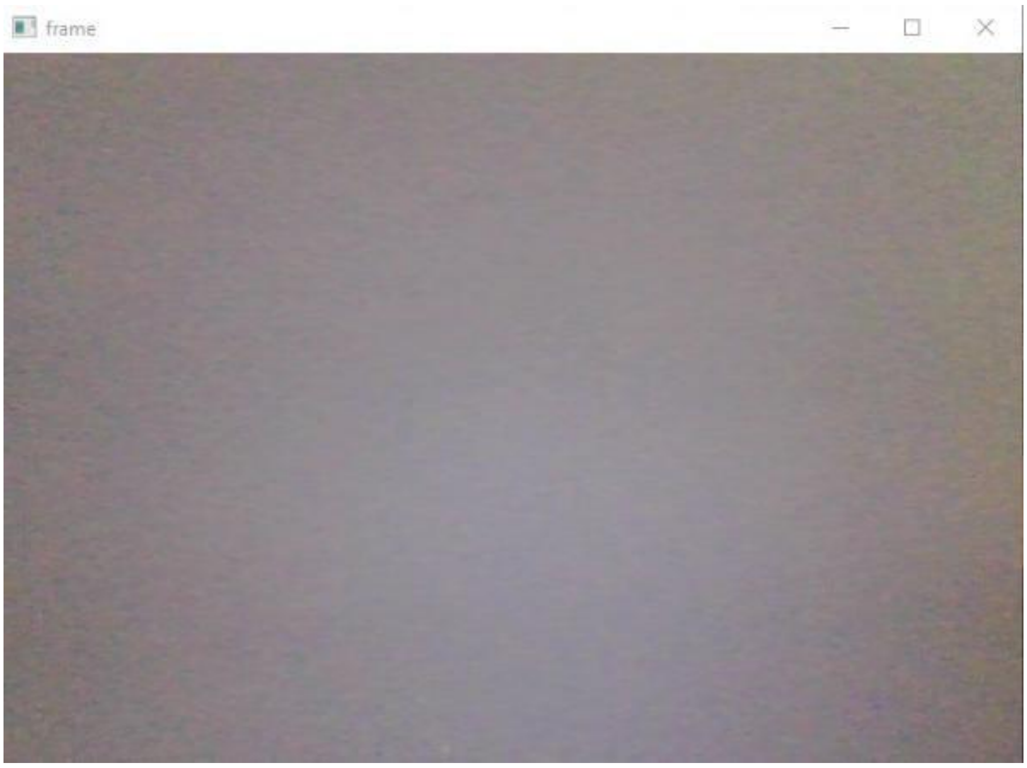
```
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
inp = std * inp + mean
inp = np.clip(inp, 0, 1)

plt.figure(figsize=(5,5))
plt.imshow(inp)

if title is not None:
plt.title(title)
plt.pause(0.001)
inputs, classes = next(iter(dataloaders['train']))





out = torchvision.utils.make_grid(inputs)
imshow(out, title=[class_names[x] for x in classes])
```

5.3.3 Webcam

```
import cv2
from PIL import Image
import torch
import torchvision
from torchvision import datasets, models, transforms

class_names = ['with_mask', 'without_mask']
```

```
filepath = 'model.pth'
model = torch.load(filepath, map_location='cpu')

def process_image(image):
    pil_image = image

    image_transforms = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

 ])

    img = image_transforms(pil_image)
    return img




def classify_face(image):
    device = torch.device("cpu")
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    #im_pil = image.fromarray(image)
    #image = np.asarray(im)
    im = Image.fromarray(image)
    image = process_image(im)

# print('image_processed')
    img = image.unsqueeze_(0)
    img = image.float()

    model.eval()
    model.cpu()

  output = model(image)
    # print(output,'#############output###########')
    _, predicted = torch.max(output, 1)
    # print(predicted.data[0],"predicted")

   classification1 = predicted.data[0]
    index = int(classification1)
    # print(class_names[index])
    return class_names[index]

face =
cv2.CascadeClassifier('C:/Users/rushitha/Documents/project/haarcascade_frontalface_
default.xml')
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
score=0
thicc=2




#
while(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]
    label = classify_face(frame)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        cv2.putText(frame, str(label), (x, y), font, 1, (255,255,255), 1, cv2.LINE_AA)

    cv2.putText(frame, str(label), (100, height -20), font, 1, (255,255,255), 1,
cv2.LINE_AA)

    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

LIVE WEB IMAGE:



Face Mask Detection in webcam stream:

The flow to identify the person in the webcam wearing the face mask or not. The process is two-fold.

1. To identify the faces in the webcam
2. Classify the faces based on the mask.

Identify the Face in the Webcam:

To identify the faces a pre-trained model provided by the OpenCV framework was used.

The model was trained using web images. OpenCV provides 2 models for this face detector.

5.3.4 FUNCTIONAL REQUIREMENTS

The primary purpose of computer results is to deliver processing results to users. They are also employed to maintain a permanent record of the results for future use.
In general, the following are many types of results:

External results are those that are exported outside the company.

- Internal results, which are the main user and computer display and have a place within the organization.
- Operating results used only by the computer department.
- User-interface results that allow the user to communicate directly with the system.
- Understanding the user's preferences, the level of technology and the needs of his or her business through a friendly questionnaire.

5.3.5 NON-FUNCTIONAL REQUIREMENTS

5.3.6 SYSTEM CONFIGURATION

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

5.3.7 HARDWARE REQUIREMENTS

• RAM: 4 GB

• Storage: 500 GB

• CPU: 2 GHz or faster

• Architecture: 32-bit or 64-bit

5.3.8 SOFTWARE REQUIREMENTS

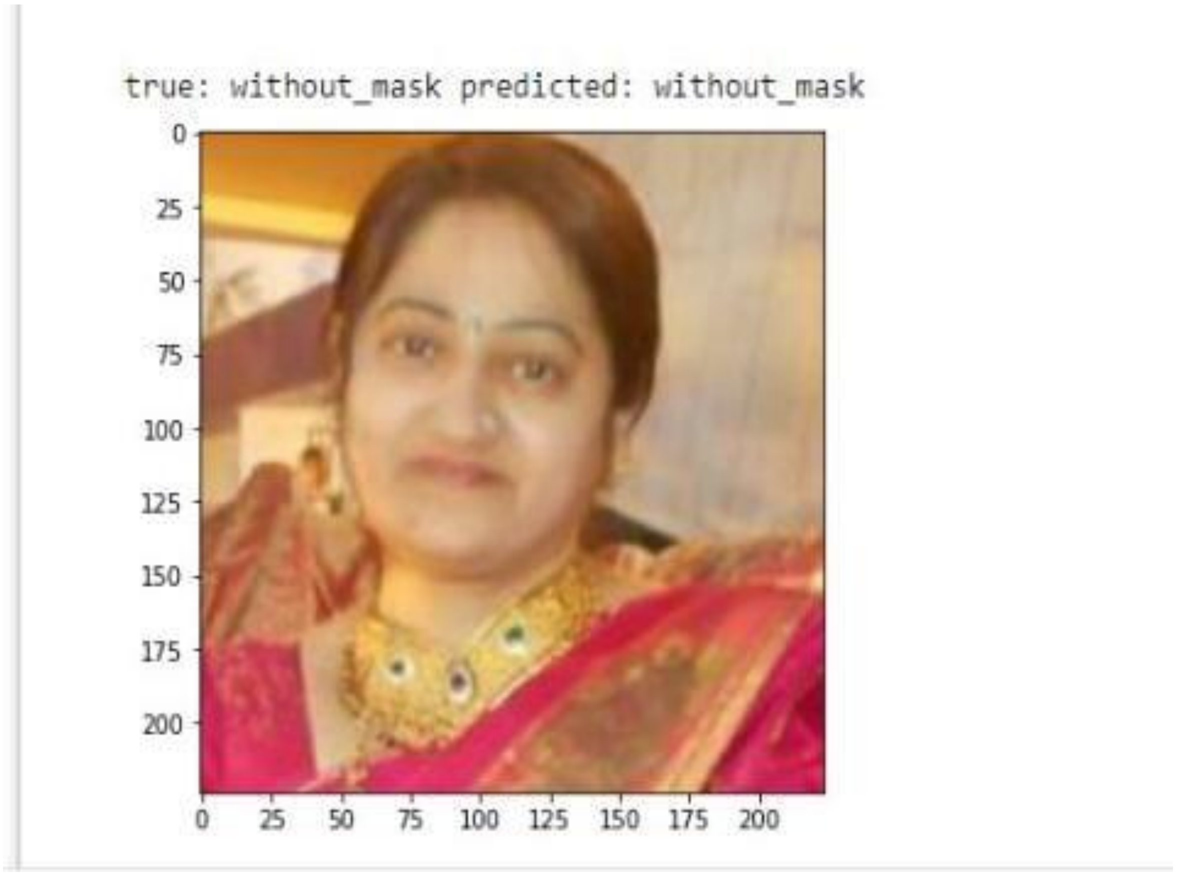
- Python 3.5 in Google Colab is used for data pre-processing, model training and prediction
- Operating System: windows 7 and above or Linux based OS or MAC OS
- Coding Language **:** Python.

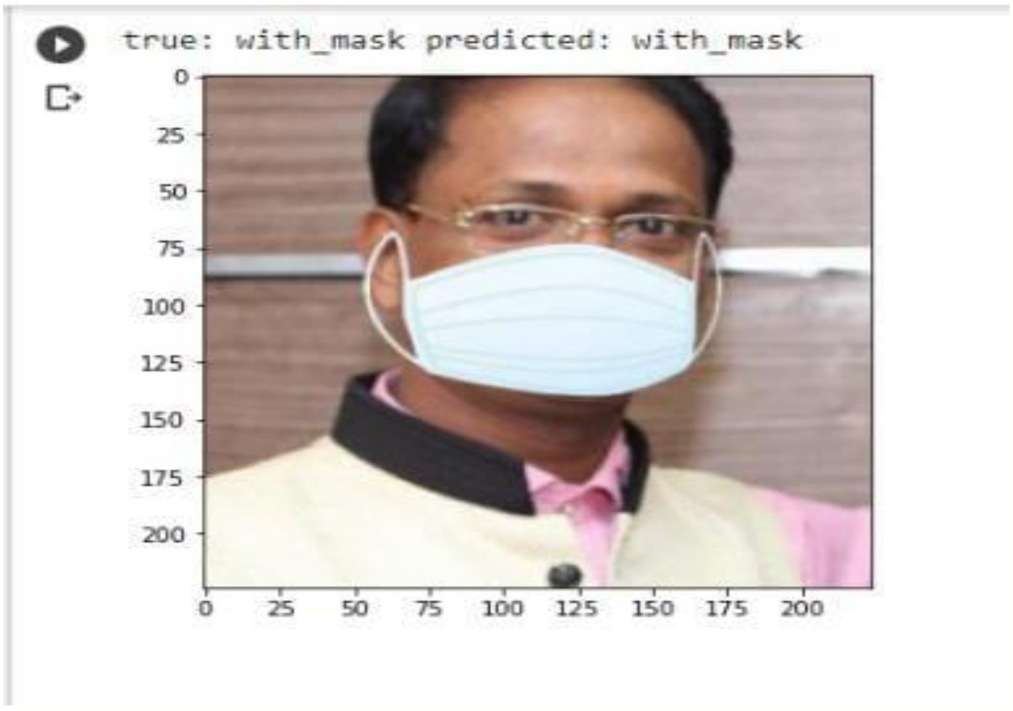
5.4 INPUT AND OUTPUT

INPUT DATASET

https://github.com/prajnasb/observations/tree/master/experiements/data

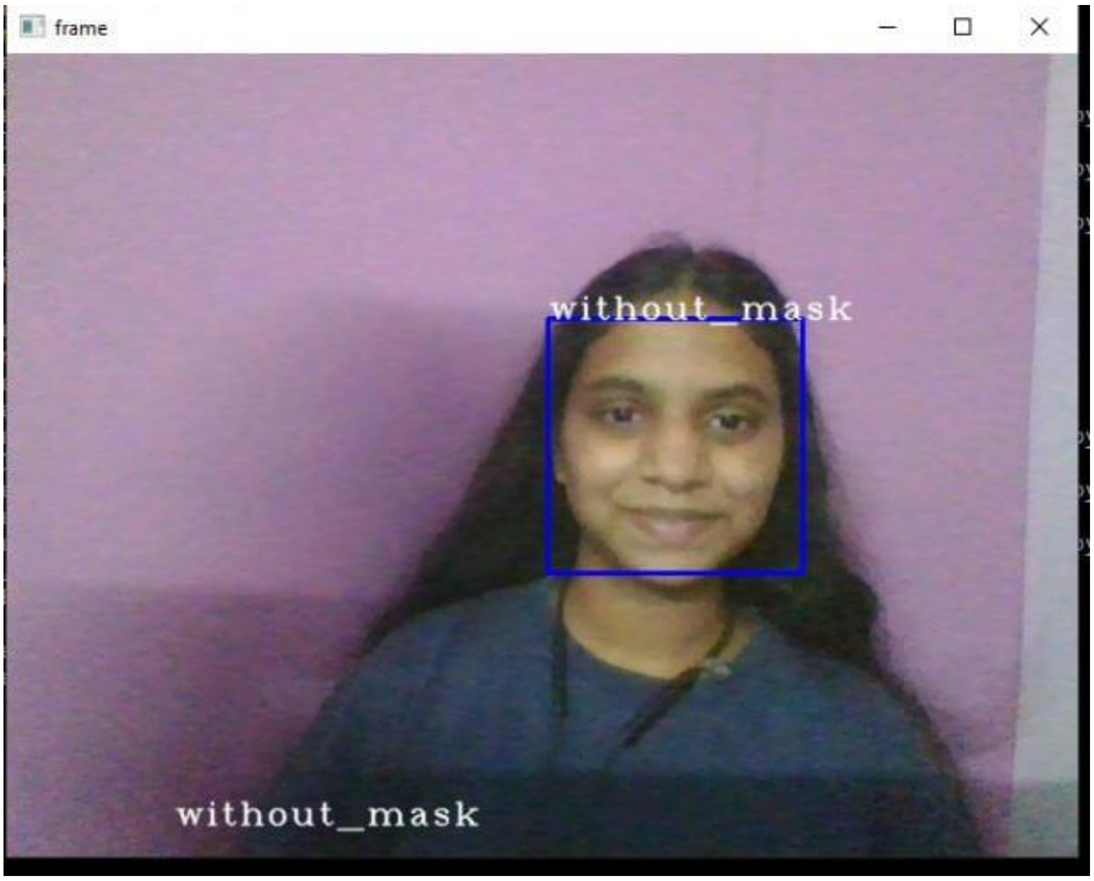
SAMPLE OUTPUT



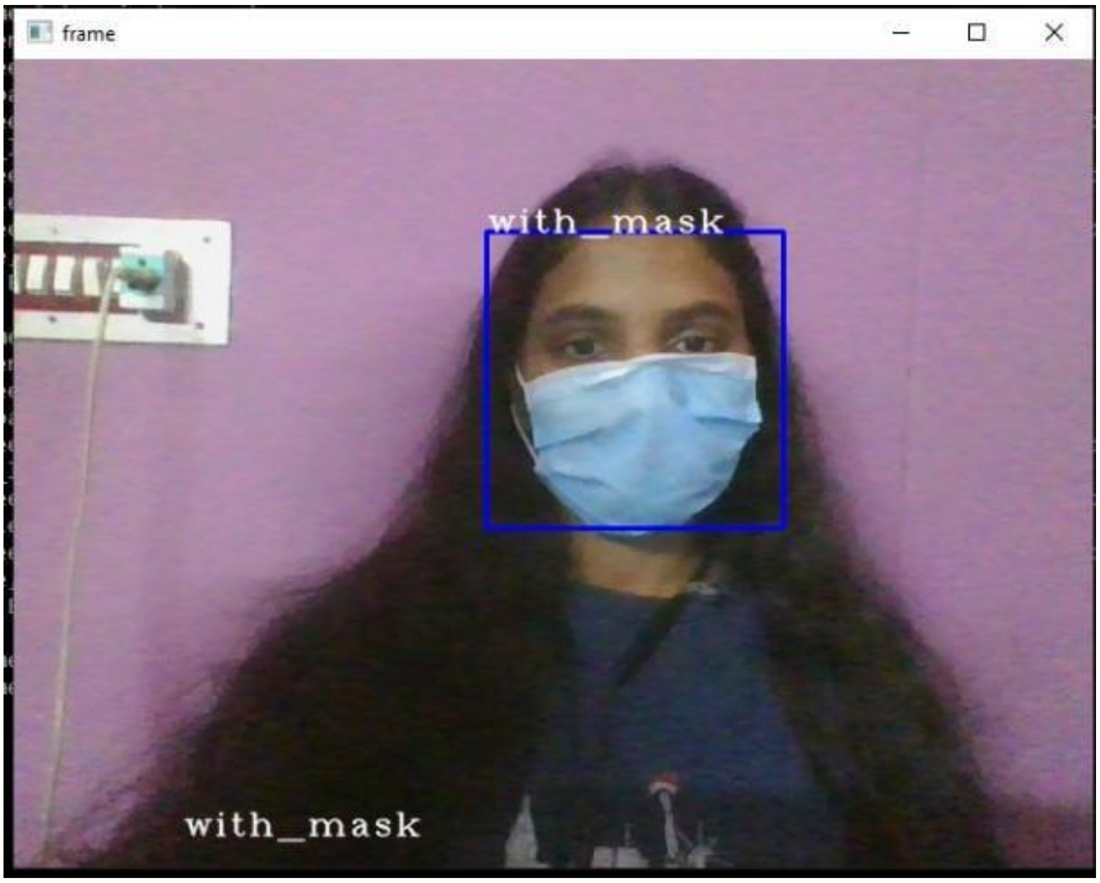


```
true: with_mask predicted: with_mask
```

REAL TIME INPUT



REAL TIME OUTPUT



CHAPTER 6
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

FUTURE ENHANCEMENT

Jingdong's recognition accuracy is stronger than 99 percent. We created the MFDD, RMFRD, and SMFRD datasets, as well as a cutting-edge algorithm based on them. The algorithm will provide contactless face authentication in settings such as community access, campus governance, and enterprise resumption. Our research has given the world more scientific and technological strength.

REFERENCES

[1]M. S. Ejaz and M. R. Islam, "Masked Face Recognition Using Convolutional Neural Network," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, doi: 10.1109/STI47673.2019.9068044.

[2] M. R. Bhuiyan, S. A. Khushbu and M. S. Islam, "A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)

[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

[1] 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.

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

[6] 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.

[2] 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.

[3] 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), pp. 711– 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

Team Members:

M. Raja Sekhar Naik (317126510151)

[illegible]

P.Rushitha (317126510159)

P. Rushitha

UNDER THE GUIDENCE OF

Mrs. B. Siva Jyothi

REFERNCE PAPER

COVID-19 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION

Vinitha.V[1], Velantina.V[2]

[1]*Student, Department of Master of Computer Application, AMC Engineering College Bangalore, Karnataka, India*
[2]*Student, Department of Computer Science and Engineering (M.Tech), C.B.I.T Kolar, Karnataka, India*

---***---

Abstract - *The corona virus COVID-19 pandemic is causing a global health crisis so the effective protection methods is wearing a face mask in public areas according to the World Health Organization (WHO). The COVID-19 pandemic forced governments across the world to impose lockdowns to prevent virus transmissions. Reports indicate that wearing face masks while at work clearly reduces the risk of transmission. An efficient and economic approach of using AI to create a safe environment in a manufacturing setup. A hybrid model using deep and classical machine learning for face mask detection will be presented. A face mask detection dataset consists of with mask and without mask images , we are going to use OpenCV to do real-time face detection from a live stream via our webcam. We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, OpenCV, and Tensor Flow and Keras. Our goal is to identify whether the person on image/video stream is wearing a face mask or not with the help of computer vision and deep learning.*

***Key Words*: Deep Learning, Computer Vision, OpenCV, Tensorflow, Keras.**

1. INTRODUCTION

The trend of wearing face masks in public is rising due to the COVID- 19 corona virus epidemic all over the world. Before Covid-19, People used to wear masks to protect their health from air pollution. While other people are self-conscious about their looks, they hide their emotions from the public by hiding their faces. Scientists proofed that wearing face masks works on impeding COVID-19 transmission. COVID-19 (known as corona virus) is the latest epidemic virus that hit the human health in the last century. In 2020, the rapid spreading of COVID-19 has forced the World Health Organization to declare COVID- 19 as a global pandemic.

More than five million cases were infected by COVID-19 in less than 6 months across 188 countries. The virus spreads through close contact and in crowded and overcrowded areas.

The corona virus epidemic has given rise to an extraordinary degree of worldwide scientific cooperation. Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight Covid-19 in many ways. Machine learning allows researchers and clinicians evaluate vast quantities of data to forecast the distribution of COVID-19, to serve as an early warning mechanism for potential pandemics, and to classify vulnerable populations.The provision of healthcare needs funding for emerging technology such as artificial intelligence, IoT, big data and machine learning to tackle and predict new diseases. In order to better understand infection rates and to trace and quickly detect infections, the AI's power is being exploited to address the Covid-19 pandemic. People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult. The monitoring process involves the detection of anyone who is not wearing a face mask.

Here we introduce a mask face detection model that is based on computer vision and deep learning. The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with opencv, tensor flow and keras. We have used deep transfer leering for feature extractions and combined it with three classical machine learning algorithms. We introduced a comparison between them to find the most suitable algorithm that achieved the highest accuracy and consumed the least time in the process of training and detection.

1.1 MACHINE LEARNING

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.

Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

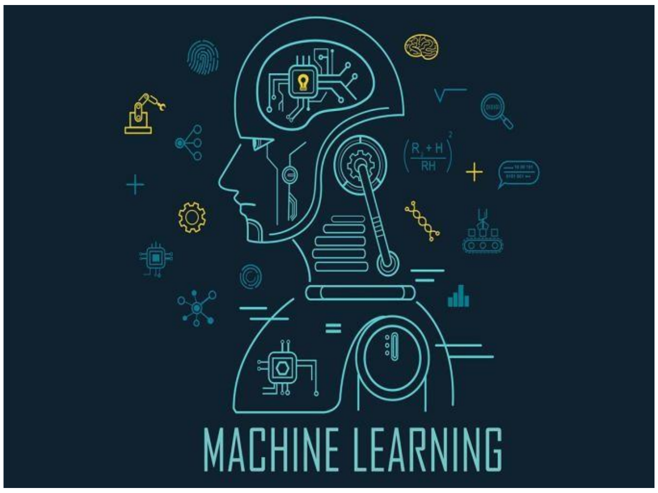


Fig -1: **Machine learning outlook**

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches have been developed which don't fit neatly into this three-fold categorization, and sometimes more than one is used by the same machine learning system.

1.2 COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do, Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions, Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multi-dimensional data from a 3D scanner or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Fig – 2: Computer Vision

Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images.

The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

1.3 DEEP LEARNING

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

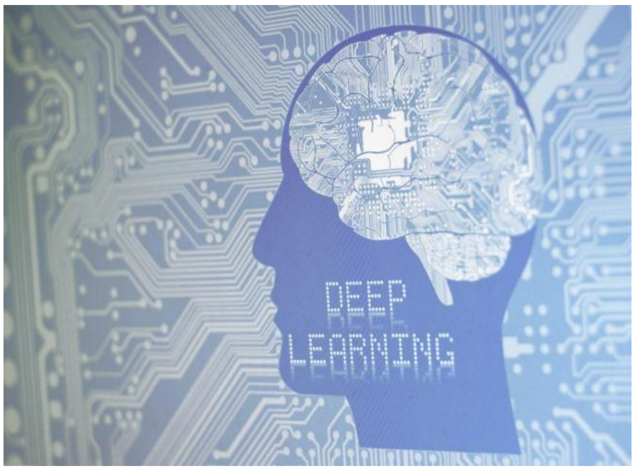


Fig – 3: Deep learning

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are **images of pixel data, documents of text data or files of audio data**. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

1.4 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source 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.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These 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. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, Video Surf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

1.5 TENSORFLOW

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. 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*. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least RankBrain in Google search and the fun DeepDream project.It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

1.6 KERAS

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:

- **Modularity**: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- **Minimalism**: The library provides just enough to achieve an outcome, no frills and maximizing readability.
- **Extensibility**: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
- **Python**: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

1.7 PyTorch

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Face book's AI Research lab (FAIR).

It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface. Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU).

- Deep neural networks built on a tape-based automatic differentiation system

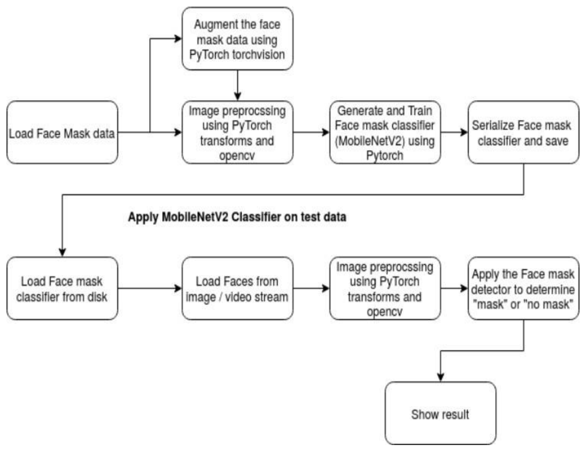
PyTorch defines a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors.

2. PROPOSED SYSTEM

The 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 and PyTorch library.

Approach
1.Train Deep learning model (MobileNetV2)
2.Apply mask detector over images / live video stream

Flow Chart


Augment the face mask data using PyTorch torchvision
Load Face Mask data
Image preprocssing using PyTorch transforms and opencv
Generate and Train Face mask classifier (MobileNetV2) using Pytorch
Serialize Face mask classifier and save
Apply MobileNetV2 Classifier on test data
Load Face mask classifier from disk
Load Faces from image / video stream
Image preprocssing using PyTorch transforms and opencv
Apply the Face mask detector to determine "mask" or "no mask"
Show result

Data at Source

The majority of the images were augmented by OpenCV. The set of images were already labeled "mask" and "no mask". The images that were present were of different sizes and resolutions, probably extracted from different sources or from machines (cameras) of different resolutions.

Data preprocessing

preprocessing steps as mentioned below was applied to all the raw input images to convert them into clean versions, which could be fed to a neural network machine learning model.

1.Resizing the input image (256 x 256)
2.Applying the color filtering (RGB) over the channels (Our model MobileNetV2 supports 2D 3 channel image)
3.Scaling / Normalizing images using the standard mean of PyTorch build in weights
4.Center cropping the image with the pixel value of 224x224x3
5.Finally Converting them into tensors (Similar to NumPy array)

Deep Learning Frameworks

To implement this deep learning network we have the following options.

1. Tensor Flow

2.Keras

3.PyTorch

4.Caffee

5.MxNet

6.Microsoft Cognitive Tool Kit

We are using the PyTorch because it runs on Python, which means that anyone with a basic understanding of Python can get started on building their deep learning models, and also it has the following advantage compared with Tensor Flow

1. Data Parallelism

2. It looks like a Framework.

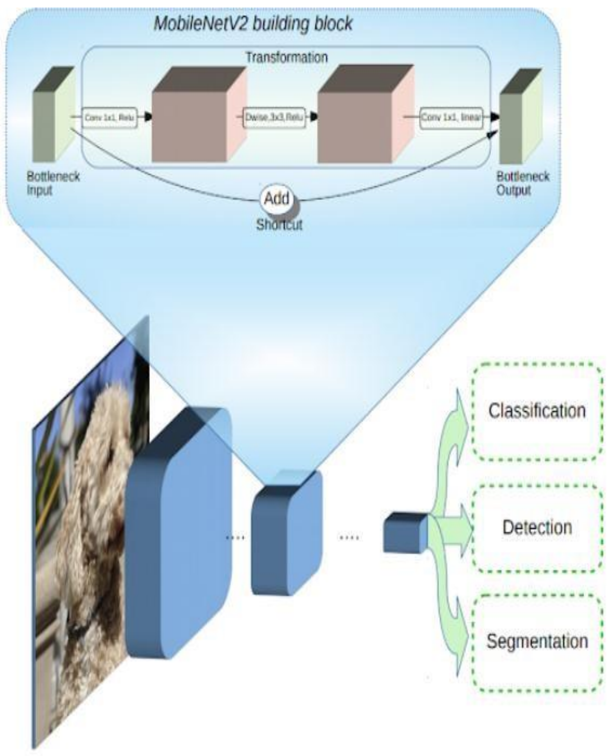
MobileNetV2

MobileNetV2 builds upon the ideas from MobileNetV1, using depth wise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture:

1) Linear bottlenecks between the layers, and

2) Shortcut connections between the bottlenecks.

The basic structure is shown below



The typical MobilenetV2 architecture has as many layers listed below, In Pytorch we can use the models library in TorchVision to create the MobileNetV2 model instead of defining/building our own model. The weights of each layer in the model are predefined based on the ImageNet dataset. The weights indicate the padding, strides, kernel size, input channels and output channels.MobileNetV2 was chosen as an algorithm to build a model that could be deployed on a mobile device. A customized fully connected layer which contains four sequential layers on top of the MobileNetV2 model was developed. The layers are

1. Average Pooling layer with 7×7 weights

2. Linear layer with ReLu activation function

3. Dropout Layer

4. Linear layer with Softmax activation function with the result of 2 values.

The final layer softmax function gives the result of two probabilities each one represents the classification of "mask" or "not mask".

Face Mask Detection in webcam stream

The flow to identify the person in the webcam wearing the face mask or not. The process is two-fold.

1. To identify the faces in the webcam
2.Classify the faces based on the mask

Identify the Face in the Webcam

To identify the faces a pre-trained model provided by the OpenCV framework was used. The model was trained using web images. OpenCV provides 2 models for this face detector:

1. Floating-point 16 version of the original Caffe implementation.

2.8 bit quantized version using Tensor flow
The Caffe model in this face mask detector. There has been a lot of discussion around deep learning based approaches for person detection. This encouraged us to come up with our own algorithm to solve this problem. Our work on face mask detection comprises of data collection to tackle the variance in kinds of face masks worn by the workers. The face mask detection model is a combination of face detection model to identify the existing faces from camera feeds and then running those faces through a mask detection model.

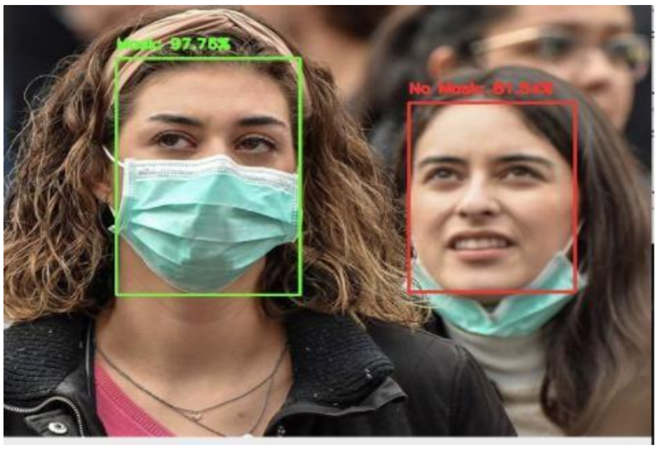


Fig – 4: Detection of "Mask" and "No Mask"

3. 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, keras , Pytorch and CNN 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] P. A. Rota, M. S. Oberste, S. S. Monroe, W. A. Nix, R. Campagnoli, J. P. Icenogle, S. Penaranda, B. Bankamp,K. Maher, M.-h. Chenet al., "Characterization of a novel coronavirus associated with severe acute respiratorysyndrome,"science, vol. 300, no. 5624, pp. 1394–1399, 2003.

[2] Z. A. Memish, A. I. Zumla, R. F. Al-Hakeem, A. A. Al-Rabeeah, and G. M. Stephens, "Family cluster of middleeast respiratory syndrome coronavirus infections,"New England Journal of Medicine, vol. 368, no. 26, pp.2487–2494, 2013.

[3] Y. Liu, A. A. Gayle, A. Wilder-Smith, and J. Rocklöv, "The reproductive number of covid-19 is higher comparedto sars coronavirus,"Journal of travel medicine, 2020.

[4] Y. Fang, Y. Nie, and M. Penny, "Transmission dynamics of the covid-19 outbreak and effectiveness of governmentinterventions: A data-driven analysis,"Journal of medical virology, vol. 92, no. 6, pp. 645–659, 2020.

[5] N. H. Leung, D. K. Chu, E. Y. Shiu, K.-H. Chan, J. J. McDevitt, B. J. Hau, H.-L. Yen, Y. Li, D. KM, J. Ipet al.,"Respiratory virus shedding in exhaled breath and efficacy of face masks."

[6] S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, "Rational use of face masks in the covid-19pandemic,"The Lancet Respiratory Medicine, 2020.

[7] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Peiet al., "Masked facerecognition dataset and application,"arXiv preprint arXiv:2003.09093, 2020.[10]Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review,"IEEE transactions on neural networks and learning systems, vol. 30, no. 11, pp. 3212–3232, 2019.

[8] A. Kumar, A. Kaur, and M. Kumar, "Face detection techniques: a review,"Artificial Intelligence Review, vol. 52,no. 2, pp. 927–948, 2019.D.-H. Lee, K.-L. Chen, K.-H. Liou, C.-L. Liu, and J.-L. Liu, "Deep learning and control algorithms of direct perception for autonomous driving,2019

BASE PAPER

FACE MASK DETECTION SYSTEM USING DEEP LEARNING

Mrs.B.Siva Jyothi1 , P.Rushitha2 , B.Chandu2 , M.Raja Sekhar2 , B. Manoj2
1Assistant Professor at Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India
2Final year students of Department of CSE, Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam-531162, India

Abstract— Pandemic on a global scale COVID-19 An epidemic of hazardous sickness erupted in 19 countries throughout the world. Wearing a face mask can help reduce the spread of infection and the transmission of infectious germs through the air. Face Mask Detection System can detect whether or not people are wearing masks. For image detection, the Haar-Cascade method is utilized. This classifier, when combined with other current algorithms, produces a high recognition rate even with varying expressions, efficient feature selection, and a low number of false positive features. The Haar feature-based cascade classifier method uses only 200 characteristics out of 6000 to achieve an 85-95 percent recognition rate. We require mask detection as a unique and public health service system during the global pandemic COVID-19 outbreak based on this rationale. Face mask and non-face mask images are used to train the model.

Index Terms— Key Words: Corona virus disease 2019, Face mask detection, CNN, Machine learning

1 Introduction

The world is still recovering from the widespread of COVID-19, and a vaccine for it's cure has yet to be discovered. To lessen the economic burden of the epidemic, several countries have allowed a restricted number of economic activities to restart once the number of new cases has decreased. Covid-19 has fallen below a particular threshold. Concerns about worker safety have surfaced in the new post-Covid-19 climate as these countries cautiously recommence their economic activity..

It is recommended that people wear masks and keep a distance of at least 1 metre between them to limit the risk of infection. Deep learning has gotten a lot of interest in the field of object detection, and it's been used to produce a face mask identification tool that can tell if someone is wearing a mask or not. This can be done by studying real-time streaming from the Camera and evaluating the categorization findings. A training data set is required for deep learning applications. This is the dataset that was used to train the model to execute various tasks.
As a result, detecting face masks has become a very important and difficult task. Face recognition without a mask is simpler, but face recognition for a normal face can be efficient for feature extraction than a masked face.Many facial characteristics, such as the nose, lips, and chin, are missing from the covered face. In the medical industry, wearing a mask minimizes the danger of being exposed to an infected person, whether or not they exhibit symptoms. In two phases, a large number of face masks can be detected.

1) Face Recognition

2) Feature Extraction
The first stage is facial recognition, which entails detecting a person's face from a picture. The most common issue is detecting several masks and uncovered faces in an image. A typical object can be used to solve the problem. method of detection Face recognition as we've known it
There are algorithms in use. Adaptive Boost, Viola-Jones Algorithm HOG and Algorithm (Histogram of Gradient). In this case, the The method of detecting objects is categorized as multi-stage. single short detectors and detectors (SSD) Multi-stage detectors use a faster RCNN, while Single Stage Detectors use Haar cascade and Single-Short Detection (SSD). Face mask detection is the subject of numerous studies here. Video analytic, picture semantic segmentation, from finger prints, DWT (Discreet Wavelet transform), and LBP are some of the approaches utilized for mask detection (Local Binary Pattern). All of these procedures are examined in order to determine whether or not a person is wearing a mask and to determine whether or not a person's face can be recognized.

2 Literature survey:

In a Smart City Network, an Automated System to Limit COVID-19 Using Facial Mask Detection[1]: COVID-19, a pandemic caused by a novel coronavirus, has been spreading over the world for a long time. COVID-19 has had an impact on practically every aspect of development. The healthcare system is in a state of emergency. Wearing a mask is one of the many preventative steps adopted to minimize the spread of this disease. In this paper, we will look upon

In a smart city network where all public places are monitored by Closed-Circuit Television (CCTV) cameras, we propose a technique to limit COVID-19 growth by identifying people who are not wearing any facial mask. When a person without a mask is spotted, the city network notifies the appropriate authority. A dataset of photos of people with and without masks acquired from diverse sources is used to train a deep learning architecture. For previously unreported test data, the trained architecture distinguished people with and without a facial mask with 98.7% accuracy.

Our research is intended to be effective in reducing the spread of this infectious disease in many areas throughout the world.

Face Recognition using a Masked Convolutional Neural Network[2]: In recent years, face recognition has become a popular and important technique. Face changes and the use of several masks make it far too difficult. Masking is another prevalent case in the real world when a person is uncooperative with equipment, such as in video surveillance. For these masks, current face recognition The quality of the work suffers. A large number of studies have been conducted on recognizing faces in a variety of situations, such as shifting stance or light, degraded photos, and so on. Nonetheless, the challenges posed by masks are sometimes overlooked. The main focus of this research is on facial masks, specifically how to improve the recognition accuracy of various masked faces. A workable strategy has been developed, which involves detecting the facial regions first. A Multi-Task Cascaded Convolutional Neural Network was used to solve the obstructed face identification problem (MTCNN). The Google FaceNet embedding model is then used to extract facial traits.

Existing system:

A Multi-Task Cascaded Convolutional Neural Network was used to solve the face detection challenge (MTCNN). The Google Face Net embedding model is then used to extract facial features.. This technique can train a dataset of both people wearing masks and those who aren't wearing masks. The system can anticipate whether or not the person is wearing the mask after training the model.

3 Methodology:

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.

DEEP LEARNING

- ✓ Deep learning is an artificial intelligence function that mimics the human brain's processing of data to detect objects, recognise speech, translate languages, and make judgments.
  - ✓ 'Deep learning' is a term that refers to AI can learn without the need for human intervention, using both organised and unlabeled data.

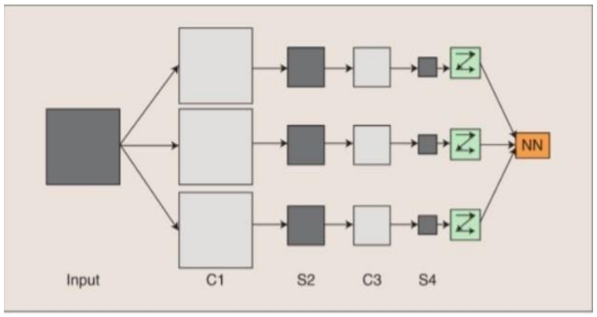


Fig 1:- deep learning model

Face mask detection is done using Convolution Neural Networks, a Deep Learning approach (CNN). The connectivity pattern between neurons in convolutional networks is similar to the organisation of the visual cortex, which was inspired by biological processes. In comparison to other image classification methods, CNNs require very little pre-processing. CNN is a type of multilayer neural network that is applied to 2-dimensional arrays (typically pictures) and is based on spatially localised neural input. CNN For pattern recognition, create "patterns of patterns." Patches from previous layers are combined in each layer. Convolutional Networks are multistage topologies with numerous stages that can be trained. Enter a Each stage produces feature maps, which are collections of arrays. Each feature map on the output represents a single feature taken from all input locations. A filter bank layer, a non-linearity layer, and a feature pooling layer make up each stage. A ConvNet is made up of three layers in which each one is followed by a classification module.
Basic structure of CNN, where it consists of two C1,C3 are convolution layers and two S2,S4 are pooled/sampled layers. In a filter bank, a trainable filter (kernel) connects the input feature map to the output feature map.Convolutional layers perform a convolution on the input before forwarding the output to the next layer. The convolution simulates a single neuron's reaction to visual input.

HAAR CASCADE

For object detection haar cascade classifier are one of the effect way.It was proposed by Michael jones and Paul Viola.It is used in Boosted cascade for rapid object detection of simplest features so this machine learning approach are used to train these classifier with lot s positive and negatives images.The images which the classifier identifies are positive images and all the other images which it could not detect are negative images.

We use this haar like features for human face detection which are divided into three formations.The edge feature is the first format,line is the second format and the last is four-rectangle feature.This haar like principle provides fast computation using the integral.So this haar cascade specific features of a face can be identified using this algorithm.Using this detection the image can be converted into a window 24X24 pixels. Initially lot of positive images and negative images are given as a data set to train this classifier .

CONVOLUTIONAL LAYER

It always comes first. It receives the image (a matrix of pixel values). Assume that the input matrix's reaction starts at the top left of the image. The software then chooses the smaller matrix there, which is referred to as a filter. The filter then generates convolution that moves over the input image. The filter's job is to multiply the original pixel values by its value. All of these multiplications are added together, yielding a single number. The filter moves because it only reads the image in the upper left corner.Additionally, one unit on the right performs a similar operation. A matrix is created after passing the filter through all points, however it is less than the input matrix. From a human standpoint, this operation is comparable to distinguishing visual boundaries and simple colours. However, in order to recognise the fish, the entire network is required. Several convolution layers will be blended with nonlinear and pooling layers in the network. The first layer to extract features from an input image is convolution. Small squares of input data are used in convolution. It's a mathematical procedure with two inputs: an image matrix and a filter or kernal.

- ✓ Dimension of an image matrix (h x w x d)
- ✓ A filter (fh x fw x d)
- ✓ Outputs a volume dimension (h-fh+1) x (w-fw+1) x1.

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

Convolution with a filter example

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called "Feature Map" as output shown in below

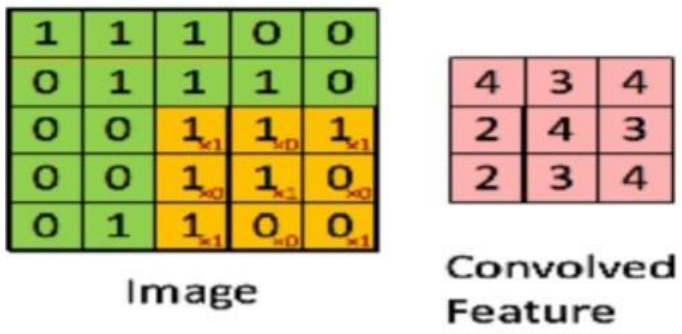


Fig 2:- Output of Convolution layer

THE NON-LINEAR LAYER:

After each convolution process, it is adder. It features an activation function that provides a non linear property; without this trait, a network would be insufficiently intense and unable to simulate the response variable.

THE POOLING LAYER:

It moves in the same direction as the nonlinear layer. It works with the image's width and height, performing a down sampling procedure on them. As a result, the size of the image is lowered. This means that if some features were already recognised during the previous convolution operation, a detailed image is no longer required for further processing and is reduced into smaller images.

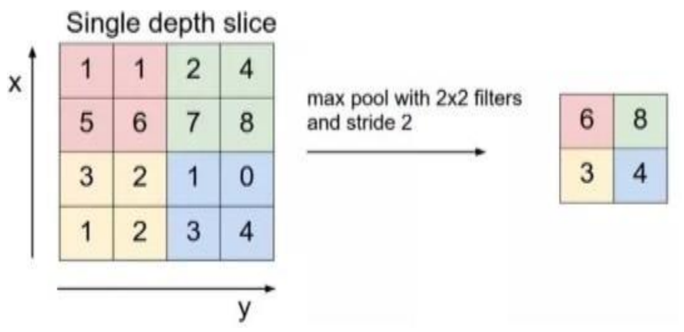


Fig 3:- Max Pooling Layer

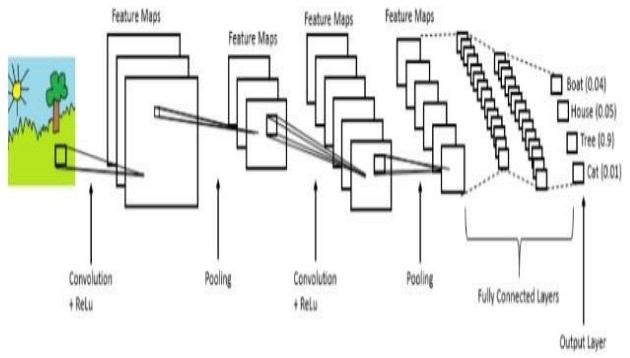


Fig 4 Overall structure of CNN

FULLY CONNECTED LAYER:

It's primary to link an overall linked layer after completing the succession of convolution, non-linear, and pooling layers. This layer receives the convolution network's output data. When a completely connected layer is attached to the network's end, it produces an N-dimensional vector, where Ni is the number of classes from which the model chooses the needed class.

CNN MODEL

1. The Tensorflow framework and the Opencv library were used to create this CNN model, which is widely utilised in real-time applications..

2. This concept can also be used to create a full-fledged software that scans everyone entering a public meeting.

LAYERS IN CNN MODEL

1. Conv2D Layer
2. MaxPooling2D Layer
3. Flatten () Layer
4. Dropout Layer
5. Dense Layer

1. Conv2D Layer:

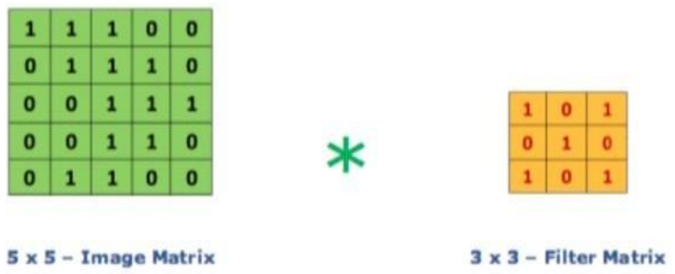


Fig 4:- 2d layer model

It has 100 filters and the activation function used is the 'ReLu'. The ReLu function stands for Rectified Linear Unit which will output the input directly if it is positive, otherwise it will output zero.

2. MaxPooling2D:
It is used with pool size or filter size of 2*2.

3. Flatten () Layer:
It is used to flatten all the layers into a single 1D layer.

4. Dropout Layer:
It is used to prevent the model from overfitting.

5. Dense Layer:
The activation function here is softmax which will output a vector with two probability distribution values.

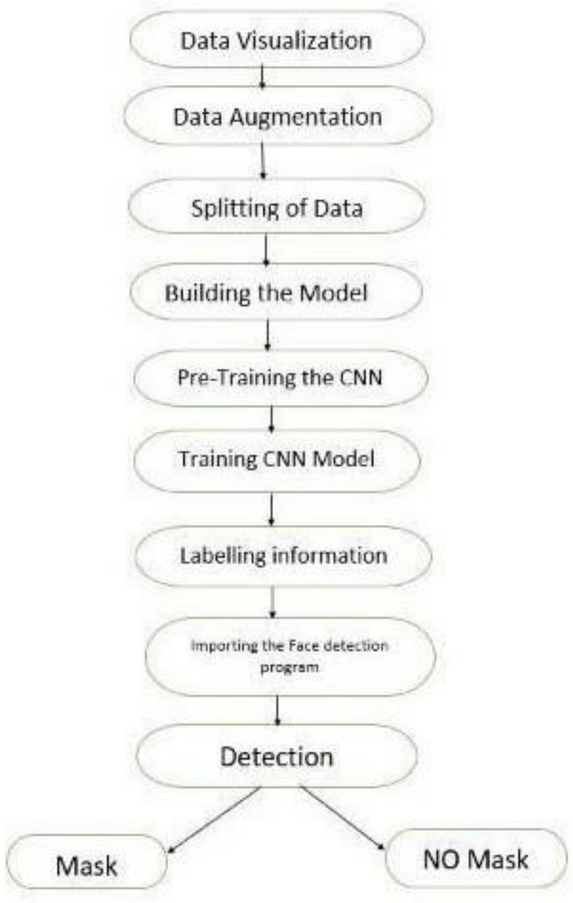


Fig 5 :- **PROPOSED SYSTEM ARCHITECTURE**

Steps

Data Visualization.

Data Augmentation.

Splitting the data.

Labeling the Information.

Importing the Face detection.

Detecting the Faces with and without Masks.

Data Visualization

Let's start by visualising the total number of photographs in both categories in our dataset. We can observe that the 'yes' class has 690 photographs while the 'no' class has 686 photos.

Data Augmentation

In the next step, we augment our dataset to include more number of images for our training. In this step of data augmentation, we rotate and flip each of the images in our dataset.

Splitting the data

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested.

Building the Model
In the next step, we build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense.

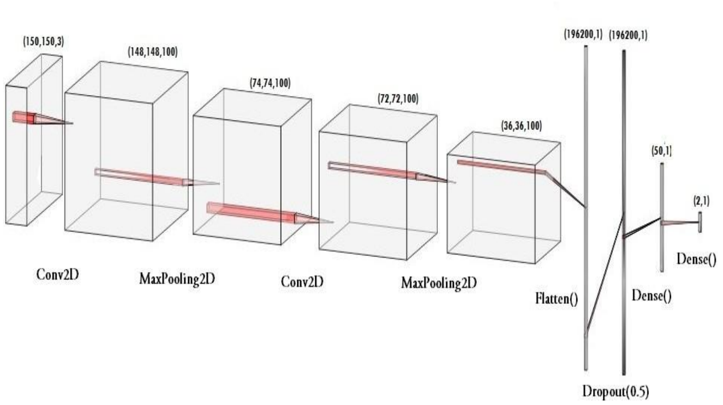


Fig 6 :-Pre-Training the CNN model

After building our model, let us create the 'train_generator' and 'validation_generator' to fit them to our model in the next step.

Training the CNN model
It is an important step where the images fit in the training set and to the test set for sequential model by using keras library.This model is trained for 30 epochs(iterations).For high accuracy we have to use more number of epochs in its training there it occurs over-fitting.

Labeling the Information
After building the model, we label two probabilities for our results. ['0' as 'without_mask' and '1' as 'with_mask']. I am also setting the boundary rectangle color using the RGB values.

Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the Haar Feature-based Cascade Classifiers for detecting the features of the face.

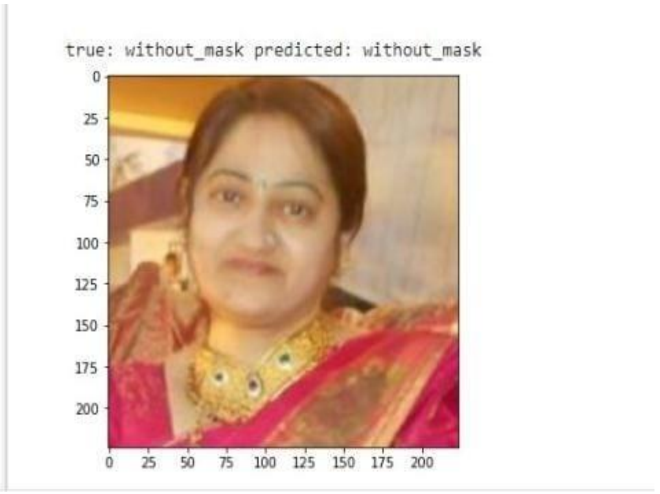
Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the Cascade Classifier.

INPUT AND OUTPUT
INPUT DATASET

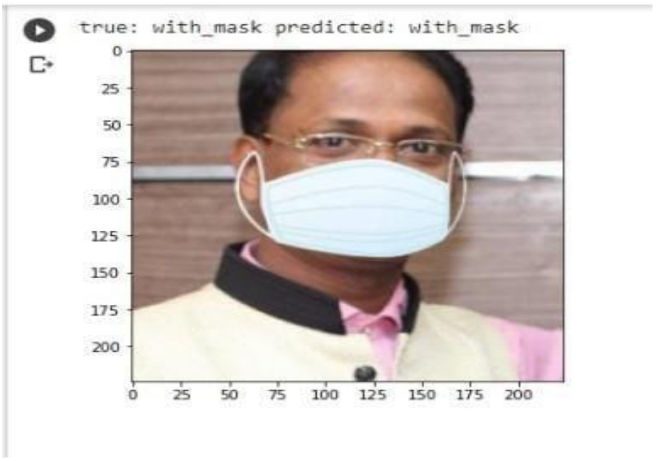
https://github.com/prajnasb/observations/tree/master/experiements/data

4 Experiential Results



```
true: without_mask predicted: without_mask
```

Fig 7 :- figure with no mask



```
true: with_mask predicted: with_mask
```

Fig 8 :- figure with mask

REAL TIME INPUT

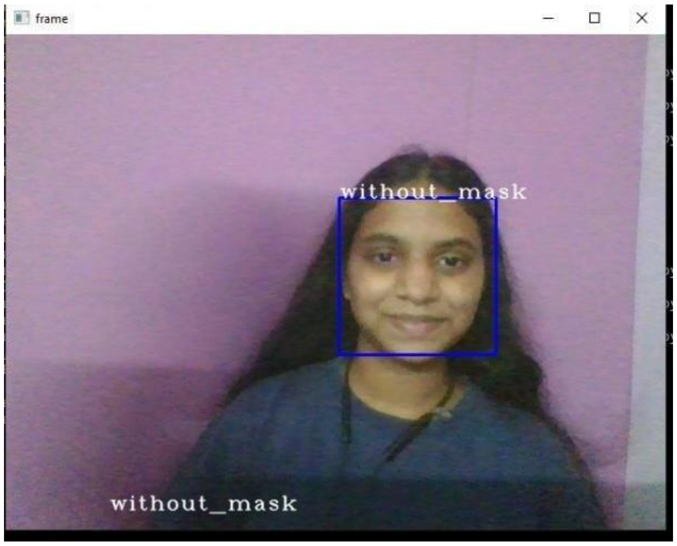


Fig 9 REAL TIME output detected as no mask

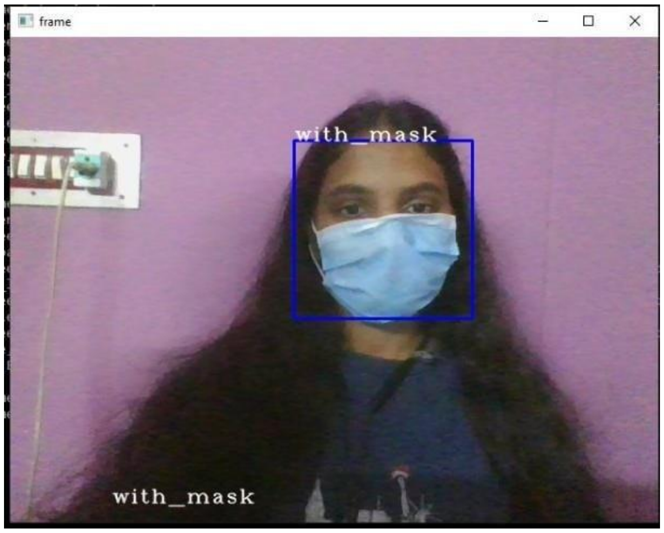


Fig 10 n REAL TIME output detected as with mask

5 CONCLUSION AND FUTURE ENHANCEMENT

The recognition accuracy of Jingdong exceeds 99 percent. The MFDD, RMFRD, and SMFRD datasets, as well as a cutting-edge algorithm based on them, were developed by us. In scenarios such as community access, campus governance, and enterprise resumption, the algorithm will provide contactless facial authentication. Our research has strengthened the scientific and technological capabilities of the planet.

6 REFERENCES

[1]M. S. Ejaz and M. R. Islam, "Masked Face Recognition Using Convolutional Neural Network," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, doi: 10.1109/STI47673.2019.9068044.

[2] M. R. Bhuiyan, S. A. Khushbu and M. S. Islam, "A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)

[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

[1] 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.

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

[6] 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.

[2] 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.

[3] 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), pp. 711– 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.ss

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