FACE MASK DETECTION USING CNN

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering in Computer Engineering

By

Sarvesh Guda (Roll No. 19102A0007)

Japleen Singh (Roll No. 19102A0002)

Harsh Pandita (Roll No. 19102A0040)

Under the Guidance of

Prof. Snehal Andhare

Department of Computer Engineering

Vidyalankar Institute of Technology
Wadala(E), Mumbai-400437

University of Mumbai

2021-22

CERTIFICATE OF APPROVAL

This is to certify that the project entitled

CHARITY DONATION SYSTEM USING BLOCKCHAIN

is a bonafide work of

Sarvesh Guda **Roll No. 19102A0007**

Japleen Singh **Roll No. 19102A0002**

Harsh Pandita **Roll No. 19102A0040**

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the

degree of

Undergraduate in **Computer Engineering**

Guide	**Head of Department**	**Principal**
(Prof. Snehal Andhare)	(Dr. Sachin Bojewar)	(Prof. Varsha Bhosale)

Project Report Approval for B. E.

This project report entitled ***Charity Donation System using Blockchain*** by

1. ***Sarvesh Guda 19102A0007***
2. ***Japleen Singh 19102A0002***
3. ***Harsh Pandita 19102A0040***

is approved for the degree of ***Bachelor of Engineering in Computer Engineering***

Examiners

1._____

2._____

Date:

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified anyidea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has notbeen taken when needed.

Sr. No.	Name of Student	Roll No.	Signature
1.	Sarvesh Guda	19102A0007	
2.	Japleen Singh	19102A0002	
3.	Harsh Pandita	19102A0040	

Date:

Acknowledgements

The completion of any project brings with satisfaction, but it is never complete without thinking those people who made it possible and whose constant support has crowned our efforts with success. One cannot even imagine the power of the force that guides us all andneither can we “succeed without acknowledging it”. We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our guide Prof. Snehal Andhare, in bringing this report to a successful completion. We also thank her for showing her keen interest, continuous encouragement, support and providing necessary facilities as and when possible towards this completion. It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our esteemed guide Prof. Snehal Andhare and our college staff of Computer Engineering Department for their revered guidance throughout our dissertationwork, which make this task a pleasant job. It was a real pleasure to work under their guidance. We are thankful to all teaching and non-teaching staff member in the departments well as in collage for their assistance in direct or indirect way.

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.

Therefore it is essential to develop a system that detects those citizens who wear a face mask and who do not.

We develop this system using Deep Learning Techniques and built a CNN model to detect people wearing masks using live web cameras input. Face masks are part of an infection control strategy to eliminate cross contamination.

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

Table of Contents

Sr.no.	Page Title	Page no.
01	**INTRODUCTION**	**01**
	1.1 Motivation	**03**
	1.2 Problem Statement	**03**
02	**LITERATURE REVIEW**	**04**
03	**SYSTEM ANALYSIS**	**06**
	3.1 Functional Requirements	**07**
	3.2 Non-functional Requirements	**07**
	3.3 Hardware Requirements	**08**
	3.4 Software Requirements	**08**
04	**SYSTEM DESIGN**	**09**
	4.1 Use Case Diagram	**12**
	4.2 Flow Chart Diagram	**13**
	4.3 Architecture Diagram	**14**
	4.4 Sequence Diagram	**15**
05	**METHODOLOGY**	**15**
	5.1 Proposed system	**16**
	5.2 Opencv	**16**
	5.3 Keras	**16**
	5.4 Machine learning approaches	**17**
	5.5 Deep Learning	**18**
	5.6 Neural Networks Vs Conventional computers	**19**
	5.7 Architectures of Neural Network	**19**
	5.8 Convoltuional Neural Network	**20**
	5.9 CNN model	**21**

Sr.no.	Page Title	Page no.
06	CODE IMPLEMENTATION	23
07	TESTING	26
08	RESULTS	28
09	CONCLUSION	30
10	FUTURE SCOPE	32
11	REFERENCES	34

List of Figures

Fig no.	Description	Page no.
4.1	Use Case Diagram	11
4.2	Flowchart	12
4.3	System Architecture	13
4.4	Sequence Diagram	14
5.7	Layers in NN	20
5.8	CNN	21
8.1	Without Mask	28
8.2	With Mask	28

List of Abbrevations

MTCNN	Multi-Task Cascaded Convolutions Neural Network
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

Introduction

INTRODUCTION

The world is still grappling with the aftermath of the COVID-19 pandemic, and a cure for the virus is yet to be discovered. To ease the economic strain caused by the outbreak, several countries have allowed limited economic activities to resume once the number of new cases has decreased below a certain threshold. However, there are concerns about worker safety as these countries cautiously reopen their economies in the new post-COVID-19 climate. To limit the risk of infection, people are advised to wear masks and maintain a distance of at least one meter from others.

Deep learning has gained significant attention in the field of object detection and has been used to develop a face mask identification tool that can determine whether someone is wearing a mask or not. This can be achieved by analyzing realtime streaming from the camera and evaluating the categorization results. A training dataset is required for deep learning applications, and this dataset was used to train the model to perform various tasks. However, detecting face masks has become a challenging and crucial task. While face recognition without a mask is simpler, recognizing a normal face is more efficient for feature extraction than a masked face. A covered face lacks many facial characteristics, such as the nose, lips, and chin.

Wearing a mask reduces the risk of exposure to an infected person in the medical industry, regardless of whether they exhibit symptoms. Face mask detection is performed in two stages: face recognition and feature extraction. The first stage involves detecting a person's face from an image, and the most common issue is detecting multiple masks and uncovered faces in a single image. Different algorithms like Adaptive Boost, Viola-Jones Algorithm, HOG Algorithm, and Histogram of Gradient are used for face recognition. In this case, the object detection method is categorized as multi-stage, single short detectors, and detectors. Multi-stage detectors use a faster RCNN, while single-stage detectors use Haar cascade and single-short detection (SSD).

Several studies have been conducted on face mask detection using various approaches, such as video analytics, picture semantic segmentation, fingerprints, DWT (Discreet Wavelet transform), and LBP (Local Binary Pattern). All these methods are examined to determine whether a person is wearing a mask and whether their face can be recognized.

1.1 MOTIVATION

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

1.2 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.

Literature Review

LITERATURE REVIEW

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.

System Analysis

SYSTEM ANALYSIS

This chapter describes the system analysis, which is conducted to study a system and its parts to identify its objectives.

3.1 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.

3.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements are as follows:

1. **Usability Requirements**

   The system should be user-friendly and does not require any guidance. In otherwords, the application has to be as simple as possible, so its users shall use it easily.

2. **Reliability Requirements**

   The system should not have any unexpected failure and must be reliable at least 98% of the time. In order to avoid any failure's occurrence, the specifications havebeen respected and followed correctly.

3.3 Hardware Requirements

The following are the details of the minimum requirements of the workstation needed for implementation and evaluation:

- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

3.4 Software Requirements

The following are the software requirements of the system:

- Python 3.9 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

SYSTEM DESIGN

SYSTEM DESIGN

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 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

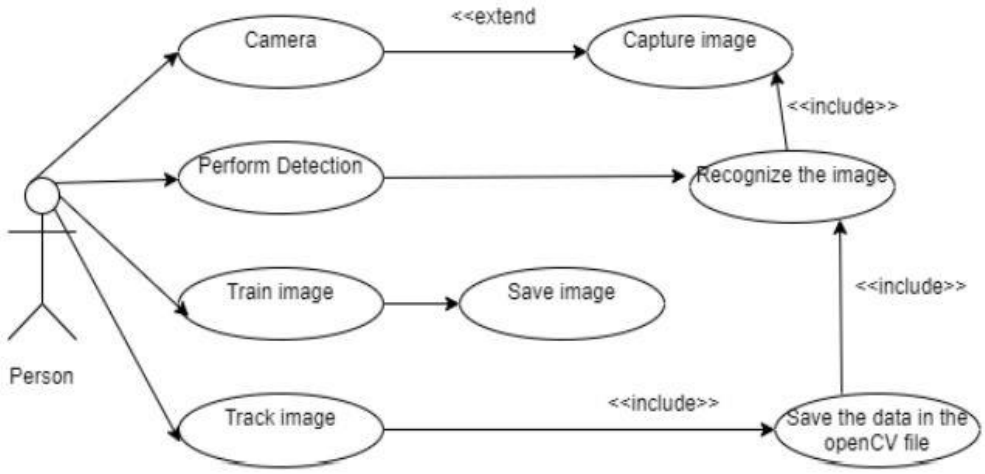


Fig 4.1: Use Case Diagram

4.2 FLOWCHART DIAGRAM

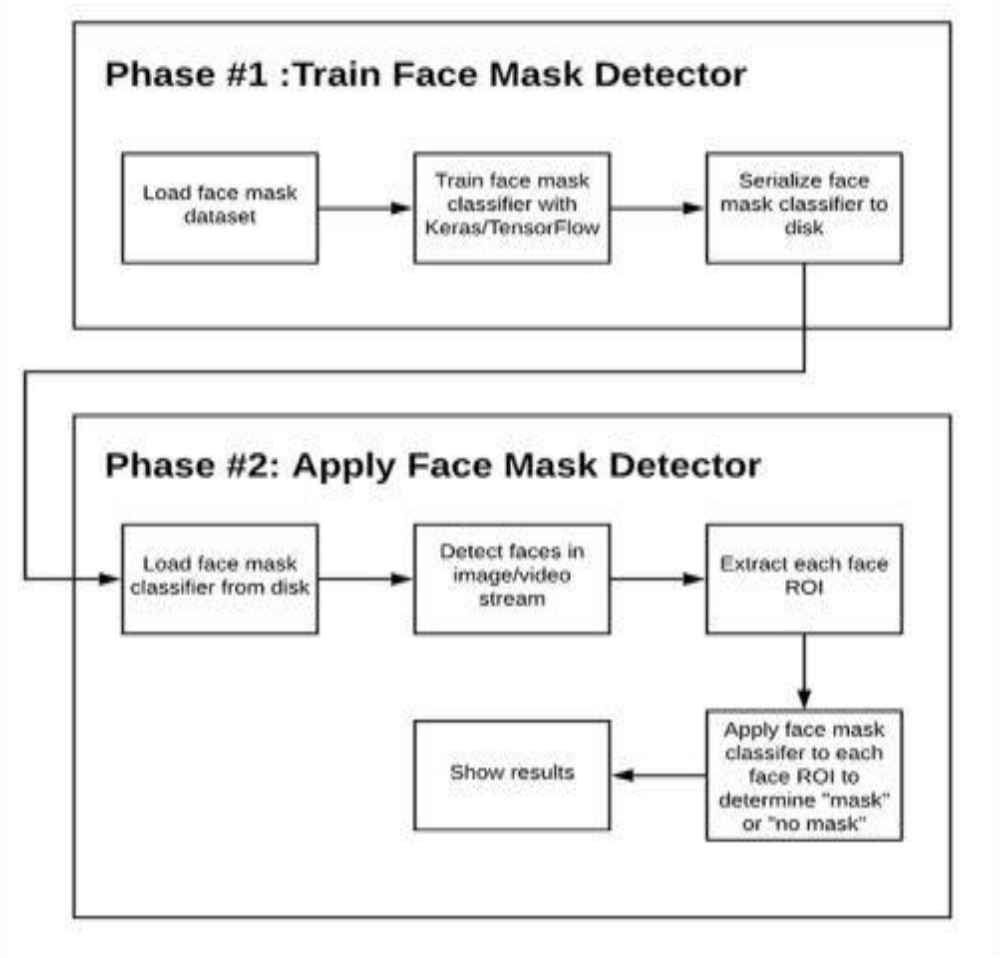


Fig 4.2: Flowchart

4.3 SYSTEM ARCHITECTURE

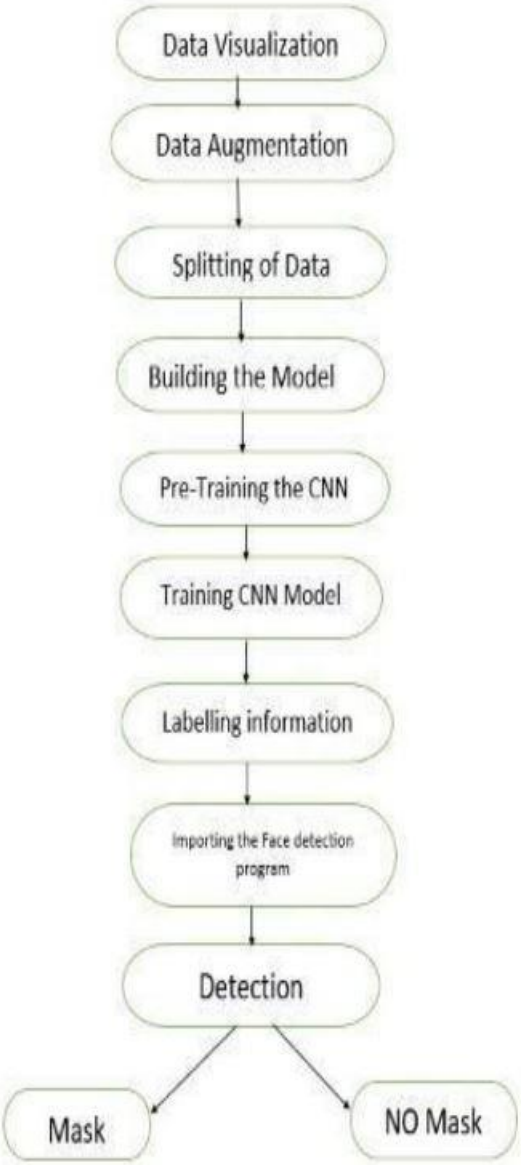


Fig 4.3 System Architecture

4.4 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.

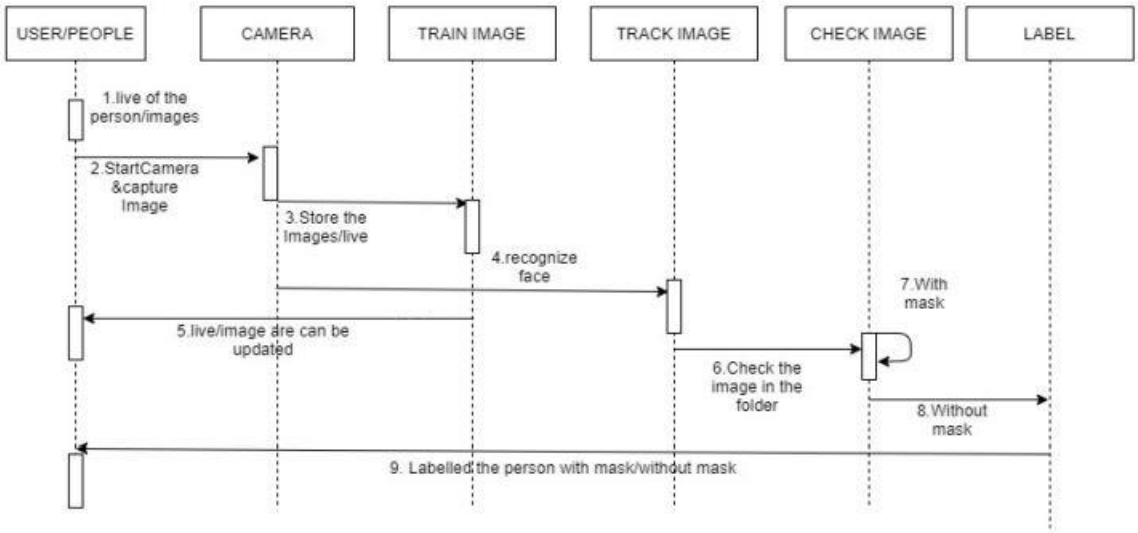


Fig 4.4: Sequence Diagram

METHODOLOGY

5.1 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.

5.2 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.

5.3 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.

5.4 MACHINE LEARNING APPROACHES:

1. Viola–Jones object detection framework based on 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%.

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 14 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.

Histogram of oriented gradients (HOG) features

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

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.

5.5 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).

5.6 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'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.

5.7 ARCHITECTURES OF NEURAL NETWORK:

5.7.1) 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.

5.7.2) 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

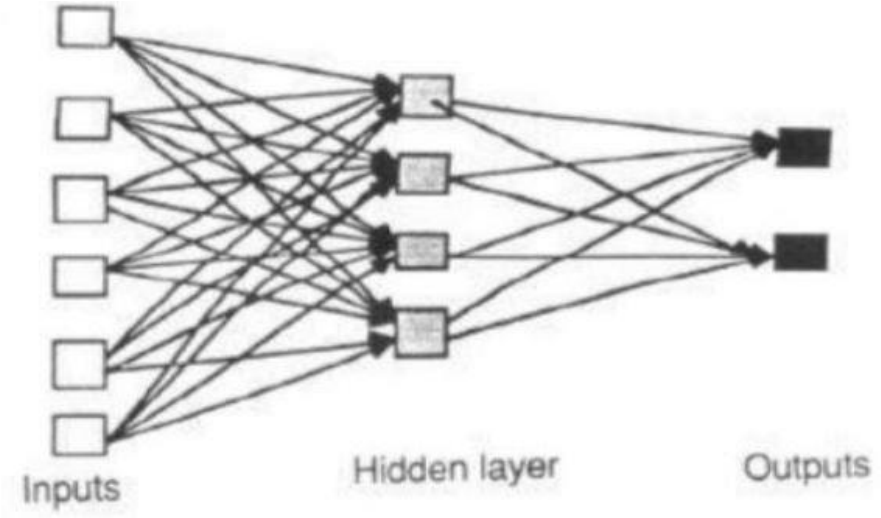


Fig 5.7: Layers in NN

5.7.3) 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.

5.8 CONVOLUTIONAL 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!

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.

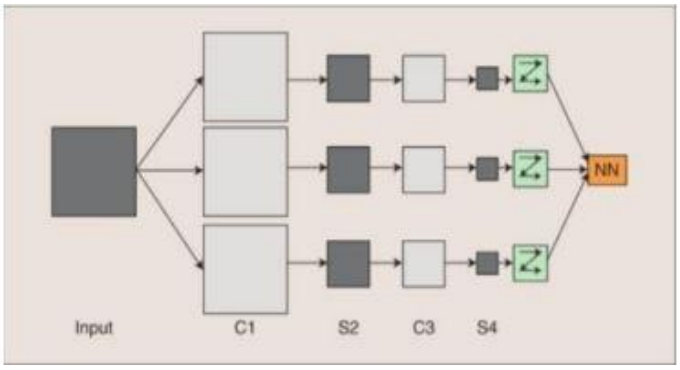


Fig 5.8 CNN

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.

5.9 CNN MODEL

1. This CNN model is built using the Keras 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.

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. **Max Pooling2D:** It is used with pool size or filter size of 2*2.
3. **Flatten ():** It is used to flatten all the layers into a single 1D layer.
4. **Dropout:** It is used to prevent the model from overfitting.
5. **Dense:** The activation function here is soft max which will output a vector with two probability distribution values.

CODE Implementation

CODE IMPLEMENTATION

Import all the libraries and modules required.

In [1]:
```
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
```

2. Build the neural network:

This convolution network consists of two pairs of Conv and MaxPool layers to extract features from the dataset. Which is then followed by a Flatten and Dropout layer to convert the data in 1D and ensure overfitting.

And then two Dense layers for classification.

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

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

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

3. Image Data Generation/Augmentation:

In [3]:
```
TRAINING_DIR = "C:\\Users\\Lenovo\\Desktop\\Face Mask Dataset (PRO)\\Train"
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=10,
                                                    target_size=(150, 150))
VALIDATION_DIR = "C:\\Users\\Lenovo\\Desktop\\Face Mask Dataset (PRO)\\Validation"
validation_datagen = ImageDataGenerator(rescale=1.0/255)
validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                              batch_size=10,
                                                              target_size=(150, 150))
```

```
Found 21198 images belonging to 2 classes.
Found 1606 images belonging to 2 classes.
```

4. Initialize a callback checkpoint to keep saving best model after each epoch while training:

In [4]:
```
checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
```

5. Train the model:

In [5]:
```
history = model.fit(train_generator,
                                epochs=10,
                                validation_data=validation_generator,
                                callbacks=[checkpoint])
```

Testing

In [ ]:
```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("C:\\Users\\Lenovo\\model2-008.model")
results={0:'without mask',1:'mask'}
GR_dict={0:(0,0,255),1:(0,255,0)}
rect_size = 4
cap = cv2.VideoCapture(0)
haarcascade = cv2.CascadeClassifier('C:\\Users\\Lenovo\\Desktop\\Face Mask Dataset (PRO)\\haarcascade_frontalface_default.xml')
while True:
    (rval, im) = cap.read()
    im=cv2.flip(im,1,1)

    rerect_size = cv2.resize(im, (im.shape[1] // rect_size, im.shape[0] // rect_size))
    faces = haarcascade.detectMultiScale(rerect_size)
    for f in faces:
        (x, y, w, h) = [v * rect_size for v in f]

        face_img = im[y:y+h, x:x+w]
        rerect_sized=cv2.resize(face_img,(150,150))
        normalized=rerect_sized/255.0
        reshaped=np.reshape(normalized,(1,150,150,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),GR_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),GR_dict[label],-1)
        cv2.putText(im, results[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
    cv2.imshow('LIVE', im)
    key = cv2.waitKey(10)

    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Result

RESULTS

This chapter contains the results achieved while implementing the system.

Fig 8.1: Without Mask

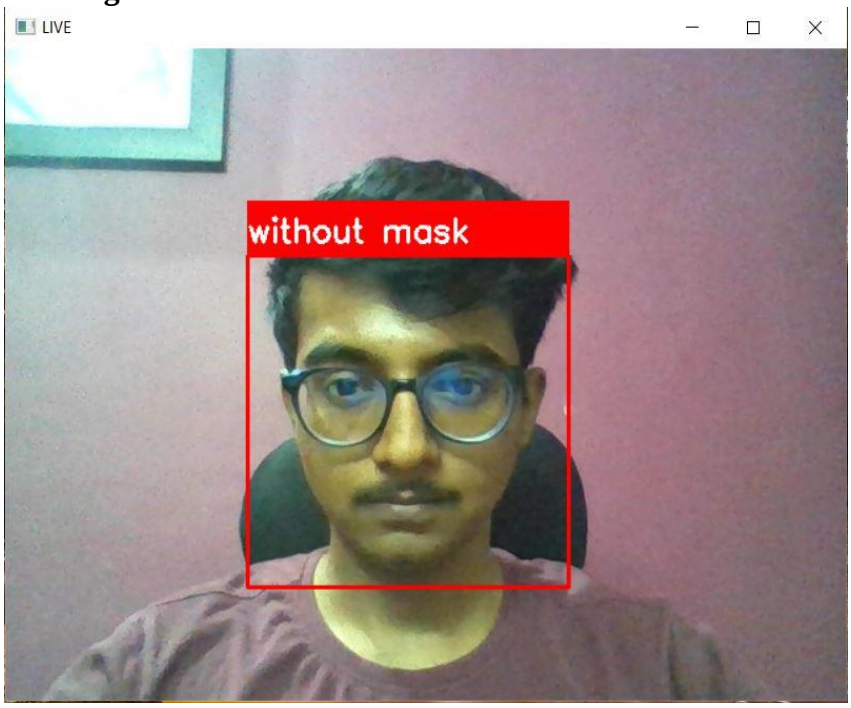
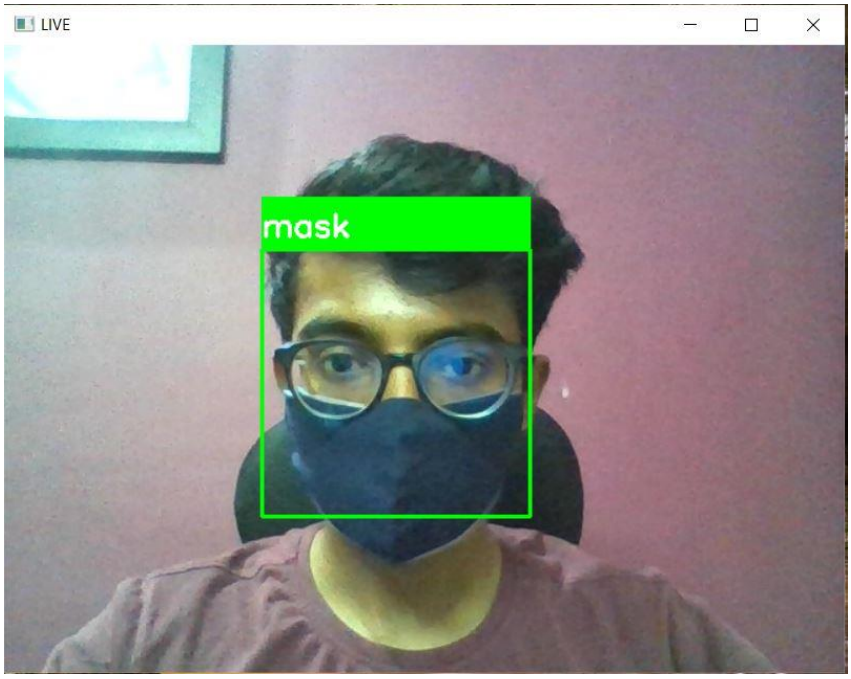


Fig 8.2: With Mask



Conclusion

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, Keras, 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 Scope

FUTURE SCOPE

- 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.
- Integrate the person identification model and face mask detection model into a single detection algorithm.
- Integrating them with automated thermal detection systems.
- Integrate it with Social Distancing systems.

References

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)

[3] 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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.