VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Project Report
on

Hand Gesture Recognition System

Submitted in partial fulfillment of the requirements for the VIII Semester of degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

Submitted By

Arnav Kashyap
1RN18IS025

Under the Guidance of

Dr. Suresh L
Associate Professor
Department of ISE

ESTD:2001
An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098

2021-2022

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Internship work entitled ***Hand Sign Detection*** has been successfully completed by **Arnav Kashyap (1RN18IS025)** Bonafede students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements of 8th semester for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2021-2022**. The internship report has been approved as it satisfies the academic requirements in respect of internship work for the said degree.

Dr. Suresh L Internship Guide Professor and HoD Department of ISE RNSIT	**Dr. Suresh L** Professor and HoD Department of ISE RNSIT	**Dr. M K Venkatesha** Principal RNSIT

External Viva

Name of the Examiners	**Signature with Date**
1. _____	**1.** _____
2. _____	**2.** _____

DECLARATION

I, **Arnav Kashyap [USN: 1RN18IS025]** student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***HandSign Recognition System*** has been carried out by me and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2021-2022.

Place: Bengaluru

Date: 10-01-2022

Arnav Kashyap

(1RN18IS025)

ABSTRACT

Hand gesture recognition is of great importance for human computer interaction (HCI) because of its extensive applications in virtual reality and sign language recognition. Human hand is very smaller with very complex articulations comparing with the entire human body and therefore errors can be easily affected.

A Hand Gesture Recognition software reduces many efforts in the tasks which can be automated. Hand gesture recognition system received great attention in the recent few years because of its manifoldness applications and the ability to interact with machine efficiently through human computer interaction.

In our framework, the hand region is extracted from the background with the background subtraction method. Then, the palm and fingers are segmented so as to detect and recognize the fingers.

ACKNOWLEDGMENT

At the very onset I would like to place our gratefulness to all those people who helped me in making the Internship a successful one.

Coming up, this internship to be a success was not easy. Apart from the sheer effort, the enlightenment of the very experienced teachers also plays a paramount role because it is, they who guided me in the right direction.

First of all, I would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of internship work.

In this regard, I express sincere gratitude to our beloved Principal **Dr. M K Venkatesha,** for providing us all the facilities.

I am extremely grateful to our own and beloved Professor and Head of Department of Information science and Engineering, **Dr. Suresh L**, for guided internship and having accepted to patronize me in the right direction with all her wisdom.

I thank **Mr. Deepak Garg, CEO, NASTECH**, for providing the opportunity to be a part of the Internship program and having guided me to complete the same successfully.

I also thank our internship coordinator **Dr. R Rajkumar,** Associate Professor, Department of Information Science and Engineering. I would thank my friends for having supported me with all their strength and might. Last but not the least, I thank my parents for supporting and encouraging me throughout. I have made an honest effort in this assignment.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

Acronym	Description
ML	Machine Learning
CNN	Convolution Neural Network
MMDB	Multi Model Biometric Data Base
LBP	Local Binary Pattern
OpenCV	Open Source Computer Vision
API	Application Programming InterHand
IDE	Integrated Development Environment
GUI	Graphical User InterHand
ROI	Region of Interest

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Machine learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that will be provided. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

But, using the classic algorithms of machine learning, text is considered as a sequence of keywords; instead, an approach based on semantic analysis mimics the human ability to understand the meaning of a text.

Machine learning algorithms are often categorized as supervised or unsupervised.

- Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The

systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

- Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Image Processing

Image processing is a way to convert an image to a digital aspect and perform certain functions on it, in order to get an enhanced image or extract other useful information from it. It is a type of signal time when the input is an image, such as a video frame or image and output can be an image or features associated with that image. Usually, the Image Processing system includes treating images as two equal symbols while using the set methods used. Image processing basically involves the following three steps: i) Importing an image with an optical scanner or digital photography ii) Analysis and iii)image management including data compression and image enhancement and visual detection patterns such as satellite imagery. It produces the final stage where the result can be changed to an image or report based on image analysis. Image processing is a way by which an individual can enhance the quality of an image or gather alerting insights from an image and feed it to an algorithm to predict the later things.

The following libraries are involved in performing Image processing in python;

- Scikit-image
- OpenCV
- Mahotas
- SimplelTK
- SciPy
- Pillow
- Matplotlib

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, multidimensional 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.

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. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

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. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If a graph is drawn showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, this approach is called 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.

1.2 EXISTING SYSTEM

This system provide the following facilities in its application:

InterHand to Add Hand Data

HandSign Detection Platform that uses Artificial Network to recognize if a user is not wearing a mask. The app can be connected to any existing or new IP mask detection cameras to detect people hand sign. App users can also add Hands and phone numbers to send them an alertin case they are. If the camera captures an unrecognized hand, a notification can be sent out to the administrator.

Real-time Monitoring Dashboard

A user-friendly website allows the user to see who is saying what and see the photo orthe video captured by the camera. User can also generate reports to download and integrate withany other third-party integration.

Notification InterHand

If the hand Sign detector application identifies a user that he/she was conveying message, AI alerts are sent with the picture of the person. It allows the application to run automatically and enforces the wearing of the mask.

Features of hand Sign Detection System:

1. **Automatically Send Alert:** Send alert to the Hands which are recognized, also set the rateof sending the alerts and detection of hand.
2. **Multi-Channel Recognition:** Attach multiple cameras in a few minutes and enable all the cameras to access the AI capability of recognizing hand.

3. **No new hardware to install :** The system can work on any existing RTSP camera without the installation of any new cameras. Most of the hospitals and airports have IP cameras installed and RTSP-enabled

1.3 PROBLEM DOMAIN

Due to the COVID-19 pandemic, our daily habits have suddenly changed. Gatherings are forbidden and, even when it is possible to leave the home for health or work reasons, it is necessary to wear a Hand Sign to reduce the possibility of contagion. In this context, it is crucial to detect violations by people who do not wear a Hand Sign. Hand Sign detection hasturned up to be an astonishing problem in the domain of image processing and computer vision. Hand detection has various use cases ranging from Hand Sign Detection to capturing facial motions, where the latter calls for the Hand to be revealed with very high precision.

1.4 PROPOSED SYSTEM

The proposed system focuses on how to identify the person on image/video stream wearing Hand Sign with the help of computer vision and deep learning algorithm by using the OpenCV, Tensor flow and Keras library. The detection is carried out to see if particular person is wearing a mask or not using Convolution Neural Networks(CNN) algorithm that Sequence model uses. As the mask model that is trained using the Sequence model is included, it checks whether the Hand detected by has a mask or not. If there is no mask it draws a red box around the Hand with a label ‘No Mask’ and if there is a mask it draws a green mask with label ‘Mask’. The logic behind drawing these green and red boxes exactly around the Hand is when the model detects a Hand it considers it as a 2-D object on geometrical plane and it assumes four points around the Hand and considers then as the Hand borders. Now these points are joined and the RGB values of the lines are changed i.e., red, blue and green components of its color mixture to (0,255,0) I.e., Red value of color is set to zero, Green 255 which is maximum and Blue is also set to zero which results in a green color box around the Hand indicating that he or she is wearing mask for visualizing and when mask is not detected in a particular Hand, the RGB value can be changed i.e., Red, Green, Blue components to (255,0,0) indicating red value 255 maximum, Green zero and Blue is also set to zero results in a red color box around the Hand indicating that he or she is not wearing mask.

CHAPTER 2

LITERATURE SURVEY

In 2012, Hand Detection using Convolutional Networks and Gabor Filters [1] proposed by Bodan Kwolek used to detecting facial regions by composing a Gabor Filters and a convolutional neural network. Gabor Filter is concentrated on extract the intrinsic facial features. The main advantages of Gabor Filter are allows the signal analysis at different scales and resolution. The convolutional neural network layer consist one or more plane. Totally 6 convolutional neural networks used here. As a result it showed providing better recognition and high rate in Hand detection than the alone performance of CNN.

In 2015 intelligent Hand Sign detection system [2] proposed by N. Ozkaya, S. Sagiroglu used for the generation of Hand Signs from its finger print. To develop an intelligent system for obtains masked Hand from fingerprints without having any knowledge about their Hands. The multi model database contains 120 persons. The IFPSF contains 4 modules including Data Enrollment and MMDB module (Multi Model Biometric Data Base). The Hand Reconstruction Module consists a pre-processing and post processing steps. Here ANN (Artificial Neural Network) analyzes the existence of any relationship between Hand and fingerprint. As a result of achieve unknown biometric feature from unknown one, here unknown biometric is Hand Sign and unknown one is fingerprint.

In 2016, study of masked Hand detection approach in video analytics [3] proposed by Gayatri Deora and Ramakrishna, here video analytic approach is used for detection. When Hand detection can be triggered by calculating the distance between a person and camera. Viola Jones Algorithm used for facial part detection, such as detection of eyes, nose and mouth etc. This algorithm provides very high detection rates and low false positive rate. As a result poor image quality leads to high false detection rate.

In 2016, Hand Sign Detection and authentication using LBP and BSIF [4] proposed by Naveens, Dr. R.S Moni. Here introduce a Hand Sign Detection and authentication method for the detection and elimination of masks. The local and global facial features are used to realize a real Hand and masked Hand. A 3D mask data based 3DMAD used here by the combination of LBP (Local Binary Pattern) and BSIF (Binarized Statistical Image Features) extract textures for

Hand authentication. The steps are included here Hand detection, feature extraction, Hand Sign Detection and Hand authentication. Feature extraction find out the global and local features for Hand region. The nose and eye region features are included in local features. By the classification of these features, finds the real or masked Hand through Hand Sign Detection process.

In 2017, Hand detection and segmentation based on improved mask R-CNN [5] proposed by Kaihan Lin and Xiaoyong Liu, used a segmentation method is based on Mask R-CNN. The Convolutional Network Model ResNet101 architecture used for extracts feature. Popular Hand benchmark dataset, FDDB (Hand Detection Data Set and Benchmark) and AFW datasets are used. A fully convolutional layer network followed by a max pooling layer is used for creating a mask. As a result it gives high G-mask accuracy than normal mask accuracy.

In 2018, Detection of 3D mask in 2D Hand Sign Detection system by using DWT and LBP [7] proposed by Arti Mahore and Meenakshi Tripathi, here detection of 3Dmask is based on anti-spoofing. It follows the detection approaches categories such as hardware, software and user collaboration. In hardware method uses an external hardware for creating a mask. Software based method uses texture-base analysis. The input RGB image is covered luminance and chrominance parts, DWT is processed these channel efficiently. Feature extraction process is carried out by using a Local Binary Pattern (LBP). The SVM (Support Vector Machine) classifier is analyzed it is a real or fake image.

In 2020, Retinal Detector [6] proposed by Mingjie Jiang, Xinqi fan and Hong, here introduces a Retinal Hand Sign Detector. It is a One-stage object detector. The dataset contained 7959 images. The ResNet and mobile Net used as BACKBONE. But ResNet is considered as standard backbone. The detection network includes a backbone, a neck and head modules. As a result the ResNet accuracy is very much higher than the Mobile Net.

In 2020, “Real time data analysis of Hand Sign detection and social distance measurement using Matlab” proposed by S. Meivel, K. Indira Devi, S. Uma Maheswari, J. Vijaya Menaka[7]. In this paper they have introduced mask detection technique done using Matlab. Data set allocation and R-CNN algorithms work more efficiently according to author when integrated with Matlab. Complex pictures with large crowds and low light are dealt in this model of mask detection using Matlab. The Faster R-CNN algorithm is generally used in

complex security and medical systems. Lot of complexities like colour changes, contrast changes, brightness changes, balanced Hand restricting are dealt in this model using Matlab. The researchers who are involved in this work have mainly concentrated on the complexities in these Hand detection models generally and taken it as a challenge to improve the models efficiency instead of these challenges, hence this a good and unique attempt to take up the difficulties and sort out the challenges for future use and development of these mask detection models and they have chosen Matlab as the new inclusion in order to achieve this feat.

CHAPTER 3

REQUIREMENT ANALYSIS, TOOLS &TECHNOLOGIES

3.1 HARDWARE REQUIREMENTS

Processor	: Any Processor above 500 MHz
RAM	: 4 GB
Hard Disk	: 220 GB
Input device	: Standard Keyboard and Mouse
Output device	: VGA and High Resolution Monitor
Operating system	: Windows 10

3.2 SOFTWARE REQUIREMENTS

3.2.1 LANGUAGE USED

PYTHON

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects

Use of Python in Machine Learning and computer Vision

Easy and Fast Data Validation

The job of machine learning is to identify patterns in data. An ML engineer is answerable for harnessing, refining, processing, cleaning, sorting out, and deriving insights from data to create clever algorithms. Python is easy while the topics of linear algebra or calculus can be so perplexing, they require the maximum amount of effort. Python can be executed rapidly which allows ML engineers to approve an idea immediately.

Different Libraries and Frameworks

Python is already very well-known and thus, it has many various libraries and frameworks that can be utilized by engineers. These libraries and frameworks are truly valuable in saving time which makes Python significantly more well-known.

Code Readability

Since machine learning includes an authentic knot of math, now and then very troublesome and unobvious, the readability of the code (also outside libraries) is significant to succeed. Developers should think not about how to write, but rather what to write, all things considered.

This specific language is extremely strict about appropriate spaces. Another of Python's advantages is its multi-paradigm nature, which again empowers engineers to be more adaptable and approach issues utilizing the simplest way possible.

Low-entry Barrier

Python is not difficult to get familiar with a language. Hence, the entry barrier. is low. What's the significance here? That more data scientists can become experts rapidly and thus, they can engage in ML projects. Because of its easy phrase structure, you can unhesitatingly work with complex systems.

Portable and Extensible

This is a significant reason why Python is so mainstream in Machine Learning. So many cross-language tasks can be performed effectively on Python due to its portable and extensible nature. There are numerous data scientists who favor utilizing Graphics Processing Units (GPUs) for training their ML models on their own machines and the versatile idea of Python is appropriate for this.

3.2.2 PLATFORM

JUPYTER NOTEBOOK

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interHand allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

A Jupyter Notebook provides you with an easy-to-use, interactive data science environment that doesn't only work as an integrated development environment (IDE), but also as a presentation or educational tool. Jupyter is a way of working with Python inside a virtual "notebook" and is growing in popularity with data scientists in large part due to its flexibility. It gives you a way to combine code, images, plots, comments, etc., in alignment with the step of the "data science process." Further, it is a form of interactive

computing, an environment in which users execute code, see what happens, modify, and repeat in a kind of iterative conversation between the data scientist and data.

3.3 LIBRARIES/API USED

3.3.1 TENSOR FLOW

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

3.3.2 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.

Libraries of Keras used

1. **Keras ImageDataGenerator**

   Keras **ImageDataGenerator** class provides a quick and easy way to augment your images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more.

2. **Keras Models – Sequential**

   Keras model represents the actual neural network model. Keras provides a two mode to create the model, simple and easy to use *Sequential API* as well as more flexible and advanced *Functional API*

   A ANN model can be created by simply calling **Sequential()** API as specified below-from keras.models import Sequential

   model = Sequential()

3.3.3 MATPLOTLIB

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source

alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming InterHands) to embed plots in GUI applications.

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

3.3.4 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python. In Python there are lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

3.3.5 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 Hands, 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.

HaarCascasecade Classifier-

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones.

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

3.4 FUNCTIONAL REQUIREMENTS

The functional requirement define the system or the components of the system. A function is basically inputs, behaviors and outputs. Stuff that can be called functional requirements are: calculations, technical details, data manipulation and processing. It describes what a system is supposed to do. A Functional prerequisite is described as one portion or an element of a product , in the entire methodology of programming building. A capacity or part is also depicted as the lead of a section and its yields, given a great deal of data sources. A useful prerequisite may be the figuring identified with specialized and subtleties or data control and getting ready or whatever other express usefulness that describes the target of a particular structure uses the useful necessities are found being utilized cases.

- Accurate Hand detection
- Accurate mask position
- Perform mathematical computation
- Approximate percentage calculation
- Real-time monitoring
- Display of result

3.5 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a requirement that gives the criteria that can be used to judge how well a system can function. It comes under system/requirements engineering. It gives a judgement on the overall unlike functional requirements which define specific behavior or functions. Functional requirements are implemented by using the system design whereas system architecture is what is used for implementing the non-functional requirements. Non-functional requirements are also called constraints.

Some of the quality attributes are as follows:

Accessibility:

- Accessibility is a term that is used to describe if a product or software is accessible to the public and how easily it can be accessed.
- It is easy to access as the dataset is open source.

Maintainability:

- Maintainability describes how easily a software or tool or system can be modified in order to correct defects and meet new requirements.
- Different programming languages can be used to make the predictive model based on the programmer's wishes. The datasets can also be modified and new data can be added.
- Different ML algorithms can also be used to check which algorithm will give the best result.
- As python is a programming language that can adapt to new changes easily, it is easy to maintain this type of system.

Scalability:

- The system can work normally under situations such as low bandwidth and huge datasets.
- The Python IDE Jupyter Notebook can take care of these data and can perform the algorithms with ease.

Portability:

- Portability is a feature which tells about the ease at which one can reuse an existing piece of code when one moves from one location or environment to some other.
- This system uses python and R programming languages and they can be executed under different operation conditions provided it meets its minimum configurations. Only system files and dependent assemblies would have to be configured in such a case.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The following three steps are carried in the creation of a Hand Sign detector.

1. Dataset Collection: The dataset which is being used contains 1376 images with 690 images containing images of people wearing masks and 686 images with people without masks.
2. Training a model to detect Hand Signs: A default OpenCV module was used to obtain Hands followed by training a Keras model to identify Hand Sign.
3. Detection of mask on a person: A open CV model was trained to detect the names of the people who are not wearing masks by referring the database.

Before inputting, the following random image augmentation is performed, rotations up to 40 degrees , zoom in and zoom out up to 20%, offset width or height up to 20%, cutting angle up to 15 degrees counter clockwise, flip inputs horizontally, and points outside the boundaries of the inputs are padded from the closest available pixel of the input.

Sequence models are the machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used in sequence models.
In the proposed the system, the data set collected is augmented using Keras ImageGenerator Library. The transformations provided by ImageGenerator is applied to the data set and is stored as a Generator object. The transformed data is fed to the Sequence model for training. The neural network used for training is first built using the Keras Sequential model. 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 serving asto convert the data in 1D and ensure overfitting.

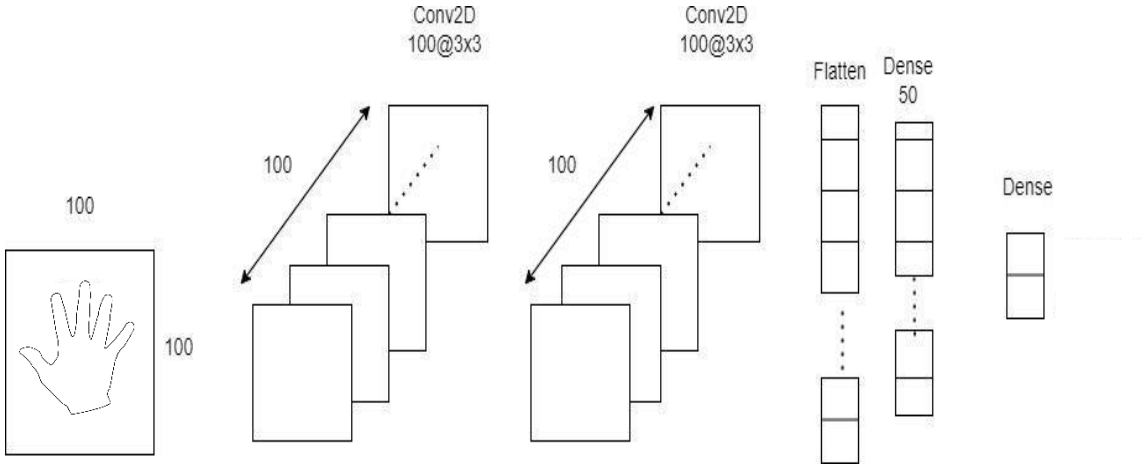

Figure 4.0.1 Convolution Neural Network Architecture

4.2 ARCHITECTURE

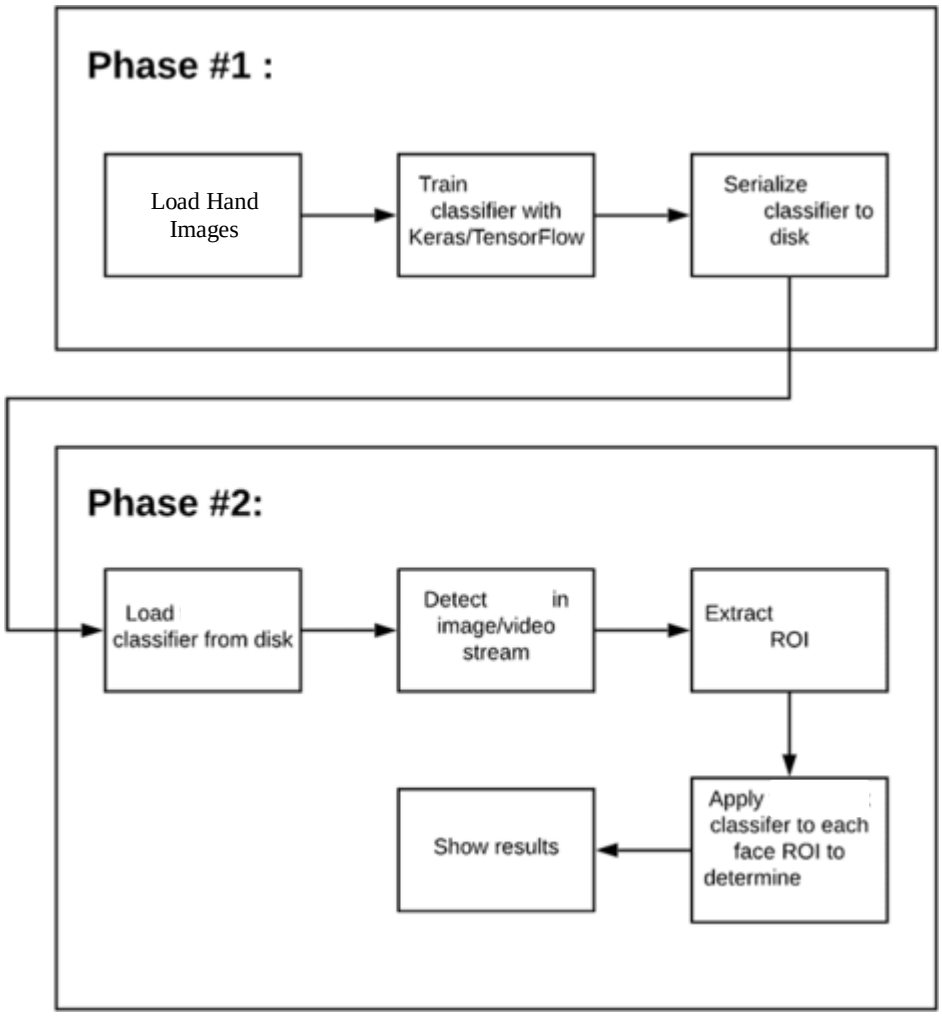

Figure 4.2 System Architecture

The above figure shows the architecture of the project. As depicted in the figure, the project is divided in two phases. Phase 1 deals with loading, preprocessing and training the data set. In this phase, the model that performs training on the data set the best, in other words, the model that provides highest accuracy, is saved onto the disk. Phase 2 deals with loading the model that was saved in the previous phase, to test it by feeding a real time video. The OpenCV library is used to detect Hands in the video stream that is fed continuously to the model. From each Hand that is detected, the region of interest (ROI) is extracted and the classifier is applied to the extracted image to detect whether the person in the image is wearing mask or not.

CHAPTER 5

DETAILED DESIGN AND IMPLEMENTATION

5.1 FUNCTIONAL MODULES

The entire project is divided into 3 modules:

1. Hand Detection
2. Hand Tracking
3. Hand Sign Detection

5.1.1 Hand Detection

This module takes input from the camera and tries to detect a Hand in the video input. The detection of the Hand is achieved through the Haar classifiers mainly, the Frontal Hand cascade classifier. The Hand is detected in a rectangle format. Hand detection is accomplished by the OpenCV algorithm proposed by Paul Viola and Michael Jones in 2001. Due to the complex background, it is not a good choice to locate or detect both the eyes in the original image, for this it will take much more time on searching the whole window with poor results. So firstly, the Hand will be located, and the range in which both the eyes are detected is reduced. After doing this the tracking speed can be improved and correct rate, reducing the effect of the complex background. Besides, this a very simple but powerful method is proposed to reduce the computing complexity.

Based on the Hand detection by the classifier model, each image array is appended into a numpy array along with the label to form the new data.

5.1.2 Cascade Classifier

The cascade classifier consists of number of stages, where each stage is a collection of weak learners. The weak learners are simple classifiers known as decision stumps. Boosting is used to train the classifiers. It provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier shows the region defined by the current location of the sliding window as either positive or negative. Positive indicates an object was found and negative

indicates no object. If the label is negative, the detector shifts the the classification of this region is complete, and window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive. It is used to eliminate less likely regions quickly so that no more processing is needed. Hence, the speed of overall algorithm is increased.

5.1.3 Detect Multiscale

detectMultiScale function is used to detect the Hands. This function will return a rectangle with coordinates(x,y,w,h) around the detected Hand. It takes 3 common arguments — the input image, scaleFactor, and minNeighbours. scaleFactor specifies how much the image size is reduced with each scale. In a group photo, there may be some Hands which are near the camera than others. Naturally, such Hands would appear more prominent than the ones behind. This factor compensates for that. minNeighbours specifies how many neighbours each candidate rectangle should have to retain it. You can read about it in detail here. You may have to tweak these values to get the best results. This parameter specifies the number of neighbours a rectangle should have to be called a Hand. These values are obtained after trail and test over a specific range.

5.1.5 Hand tracking

Due to the real-time nature of the project, the Hands need to be tracked continuously for any form of distraction. Hence the Hands are continuously detected during the entire time.. If Hands are found, it returns the positions of detected Hands as rect(x,y,w,h). Once these locations are returned, a rectangular box can be drawn around the location that would depict the region of interest(ROI) for the Hand .

5.2 DISCUSSION OF CODE

This section illustrates some of the code segments of the project.

HAND SIGN DETECTION – Processing and training.ipynb

```
from tensorflow.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.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
import matplotlib.pyplot as plt


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'])
model.summary()

TRAINING_DIR = "./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 = "./test"
validation_datagen = ImageDataGenerator(rescale=1.0/255)

validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                        batch_size=10,
                                        target_size=(150, 150))

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

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

accuracy = history.history['acc']
val_accuracy = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, "b", label="training accuracy")
plt.plot(epochs, val_accuracy, "r", label="validation accuracy")
plt.legend()
plt.show()

plt.plot(epochs, loss, "b", label="training loss")
plt.plot(epochs, val_loss, "r", label="validation loss")
plt.legend()
plt.show()
```

Working:

- The python notebook ‘Hand Sign Detection – Processing and training.ipynb’ deals with processing the acquired data set.
- Data augmentation is performed on the images present in both the test and train folders to increase the amount of images.
- Keras ImageGenerator library is used to augment the original data by applying random transformations to each image on the batch.
- The transformed images replace the original images and this batch of transformed images are then used for training the model.
- The model, Sequential neural networks, is built by specifying the shape of the input layer, the activation functions used in inout, hidden and output layers

- The function model.summary() provides the information of the Sequential model that is built.
- Once the model has been trained, the graphs of training and validation accuracies, and the training and validation losses are plotted to compare how well the model has performed.
- A check point is created that is used to save the model that performs the best in the training process.

HAND SIGN DETECTION – Testing.ipynb

```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("./model2-010.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(cv2.data.haarcascades +
'haarcascade_frontalHand_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))
    Hands = haarcascade.detectMultiScale(rerect_size)
    for f in Hands:
        (x, y, w, h) = [v * rect_size for v in f]

        Hand_img = im[y:y+h, x:x+w]
        rerect_sized=cv2.resize(Hand_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()
```

Working:

- The python notebook 'Hand Sign Detection – Testing.ipynb' deals with testing our pretrained model with live video streaming.
- The best pretrained that is saved onto the disk, is loaded using the keras load_model library.
- cv2.VideoCapture() is used to get a video capture object for the camera. The default number 0 is passed as the argument indicating the use of the internal camera of the device.
- An infinite while loop is set and the read() method is used to read the frames using the above created object.
- The images coming in from the video capture is resized before applying the haarcascade function on the same.
- Breaks the loop when the user clicks a specific key.

CHAPTER 6

TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and or/a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations does not fail in unacceptable manner.

6.1 Functional Testing:

Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black Box Testing tec logic of the system being tested is not known to the tester.Functional testing is normally performed during the levels of System Testing and Acceptance Testing. Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function's specifications.
- Determine the output based on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs

6.2 Non-Functional Testing

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. This testing has a greater impact on applications when it comes to the performance of the application under high user traffic. This testing ensures that your application is stable and is able to handle load under extreme conditions.

In this project, the testing of the live Hand Sign detection model is done as follows:

- The project is initialized by importing the necessary keras and OpenCV libraries.
- The paths to the test images are then read using the function imread() of cv2 library in OpenCV.
- The input test image is manipulated by flipping and resizing the image before feeding to the trained model.
- The HaarCascade frontal Hand algorithm is applied to the preprocessed frame orimage in order to detect the Hand in this frame.
- The extracted features from the image are fed to the pre-trained model to test the classification and accurate detection of a person wearing or not wearing a mask.

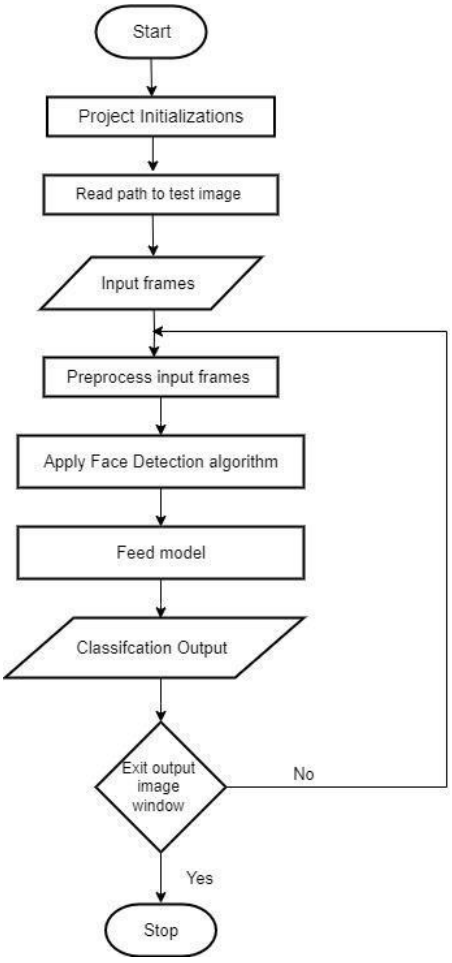

Fig 6.1 Testing Flowchart

The below images show testing of the trained Sequential model with two test images.

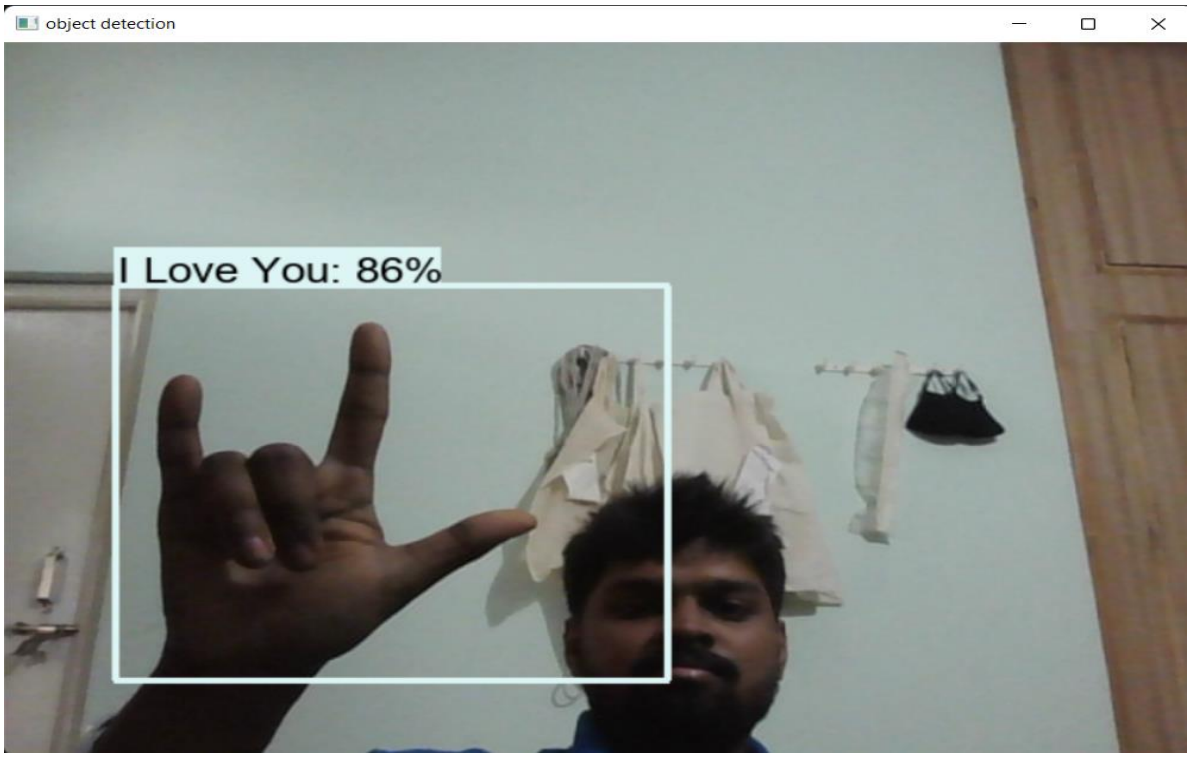

Figure 6.2 Testing with Image 1

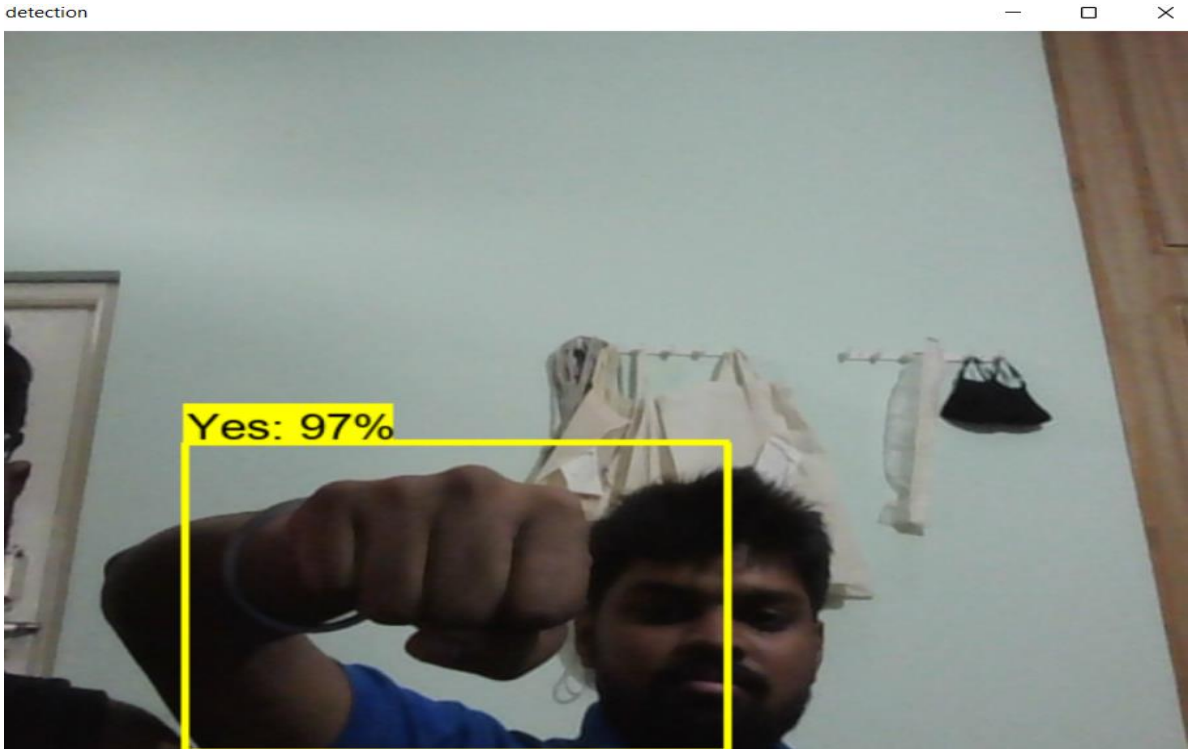

Figure 6.3 Testing with Image 2

CHAPTER 7

OBSERVATIONS AND RESULTS

7.1 RESULTS

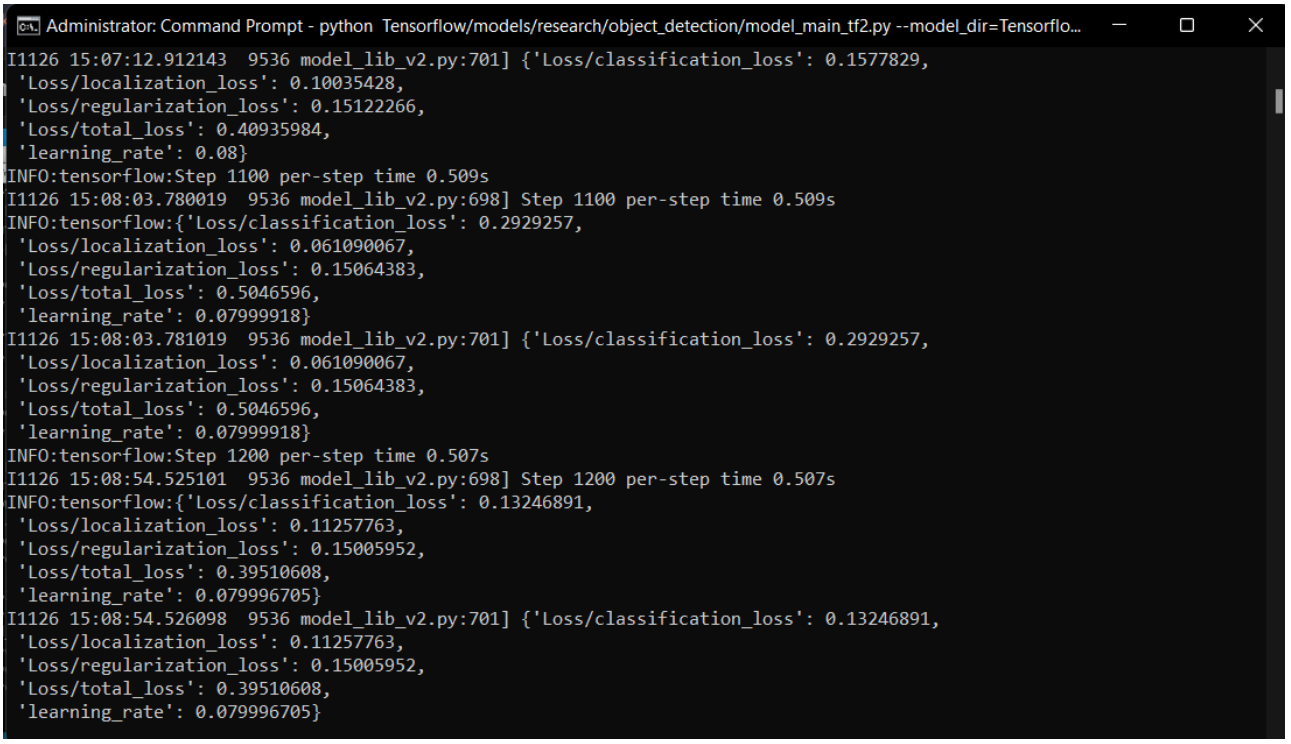

Figure 7.1 Sequential Model Creation

The above figure shows the code snippet for the creation of the training model. The activation function used in each layer is ReLu as it helps to prevent the exponential growth in the computation required to operate the neural network.

```
print("""python {}/research/object_detection/model_main_tf2.py --model_dir={}/{}
    --pipeline_config_path={}/{}/pipeline.config
    --num_train_steps=10000""".format(APIMODEL_PATH, MODEL_PATH,CUSTOM_MODEL_NAME,MODEL_PATH,CUSTOM_MODEL_NAME))
```

```
python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pi
peline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=10000
python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pi
peline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=10000
```

Figure 7.2 Image Augmentation

Data augmentation is performed to transform the train and test data.

```
In [6]: history = model.fit_generator(train_generator,
                                      epochs=10,
                                      validation_data=validation_generator,
                                      callbacks=[checkpoint])

<ipython-input-6-6272b23e5a0b>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Ple
ase use `Model.fit`, which supports generators.
  history = model.fit_generator(train_generator,

Epoch 1/10
109/109 [==============================] - ETA: 0s - loss: 0.7332 - acc: 0.5946INFO:tensorflow:Assets written to: model2-001.mo
del\assets
109/109 [==============================] - 159s 1s/step - loss: 0.7332 - acc: 0.5946 - val_loss: 0.6953 - val_acc: 0.5000
Epoch 2/10
109/109 [==============================] - 162s 1s/step - loss: 0.6764 - acc: 0.6066 - val_loss: 0.6996 - val_acc: 0.5000
Epoch 3/10
109/109 [==============================] - ETA: 0s - loss: 0.4933 - acc: 0.7424INFO:tensorflow:Assets written to: model2-003.mo
del\assets
109/109 [==============================] - 157s 1s/step - loss: 0.4933 - acc: 0.7424 - val_loss: 0.2326 - val_acc: 0.9072
Epoch 4/10
109/109 [==============================] - ETA: 0s - loss: 0.3268 - acc: 0.8587INFO:tensorflow:Assets written to: model2-004.mo
del\assets
109/109 [==============================] - 163s 1s/step - loss: 0.3268 - acc: 0.8587 - val_loss: 0.1211 - val_acc: 0.9485
Epoch 5/10
109/109 [==============================] - ETA: 0s - loss: 0.2719 - acc: 0.8920INFO:tensorflow:Assets written to: model2-005.mo
del\assets
109/109 [==============================] - 163s 1s/step - loss: 0.2719 - acc: 0.8920 - val_loss: 0.0894 - val_acc: 0.9691
Epoch 6/10
109/109 [==============================] - ETA: 0s - loss: 0.2335 - acc: 0.9160INFO:tensorflow:Assets written to: model2-006.mo
del\assets
109/109 [==============================] - 154s 1s/step - loss: 0.2335 - acc: 0.9160 - val_loss: 0.0853 - val_acc: 0.9742
Epoch 7/10
109/109 [==============================] - 153s 1s/step - loss: 0.2725 - acc: 0.8883 - val_loss: 0.0873 - val_acc: 0.9691
Epoch 8/10
109/109 [==============================] - ETA: 0s - loss: 0.2145 - acc: 0.9151INFO:tensorflow:Assets written to: model2-008.mo
del\assets
109/109 [==============================] - 165s 1s/step - loss: 0.2145 - acc: 0.9151 - val_loss: 0.0748 - val_acc: 0.9794
Epoch 9/10
109/109 [==============================] - 160s 1s/step - loss: 0.1758 - acc: 0.9418 - val_loss: 0.0950 - val_acc: 0.9536
Epoch 10/10
109/109 [==============================] - ETA: 0s - loss: 0.1860 - acc: 0.9298INFO:tensorflow:Assets written to: model2-010.mo
del\assets
```

Figure 7.3 Model Training

The above figure shows the training of Sequential model using the new augmented images.

7.2 GRAPHS

The accuracy of the model is 92%.

It can also be observed in the accuracy loss graph obtained below.

- The loss value is below 0.1.
- It can be observed that with every iteration of the model training, the loss is getting gradually decreased and the accuracy is getting increased

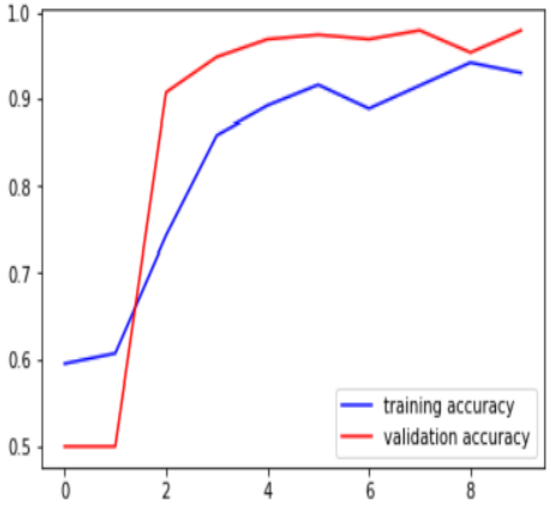

Fig 7.4 Training accuracy vs Validation accuracy

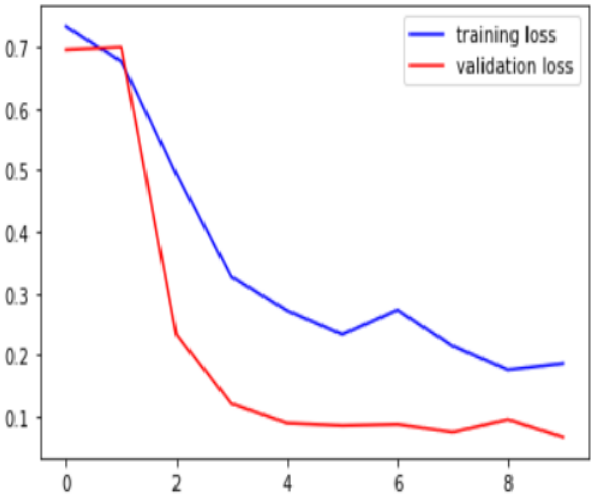

Fig 7.5 Training loss vs Validation loss

7.3 SNAPSHOTS

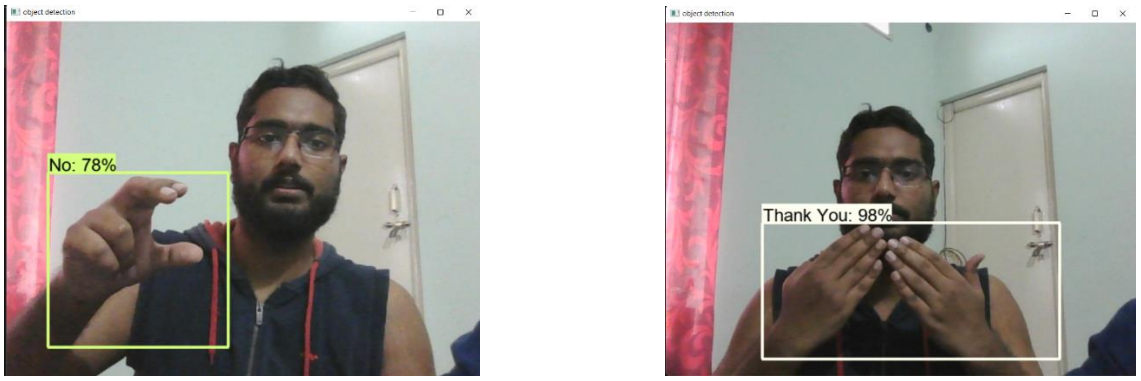

Figure 7.6 Live detection of person hand sign recognition

The fig 7.6 is output for Hand detection module. The input to this module is continuous stream ofvideo and output will be Hand detection within rectangular bounds. The Hand is detected by using haar cascade algorithm. It uses haar features through which the Hand is detected in a rectangular frames. The detection of the Hand is achieved through, the Frontal Hand cascade classifier.

The image from the video stream is resized and normalized before sending as input to the model prediction function. The above are the outputs for live detection of a person with mask and without mask.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

This research explores the opportunity and challenges in recognition of hand gestures. It also analyzes the effect of data augmentation in deep learning. We can say after this research that CNN is a data driven methodology and data augmentation has a huge effect in deep learning. The user can interact with the virtual environment by hand gestures. Although the system can recognize gestures successfully, some extension is still possible. For example, by applying knowledge driven methodology such as Belief Rule Base (BRB), which is widely used where uncertainty becomes an issue . Hence, the recognition of gesture can be accomplished more accurately.

8.2 LIMITATIONS

- A major challenge in gesture recognition is the proper segmentation of skin coloured objects (e.g. hands, face) against a complex static background. The accuracy of skin segmentation algorithms is limited because of objects in the background that are similar in colour to human skin.
- It could also be observed that the face recognition stage is not robust when the detected Gesture is at a certain angle of inclination.
- However, this is not a problem of great impact, as this application is oriented to access control and at this point, the person must maintain a firm and straight posture in front of the device that acquires the image.

8.2 FUTURE ENHANCEMENT

Gesture recognition algorithm is relatively robust and accurate. Convolution can be slow, so there is tradeoff between speed and accuracy. In the future, we will investigate other methods of extracting feature vectors, without performing expensive convolution operations. More gestures can be added to list of recognition. It was assumed that the background should be less complex. Therefore, recognition of gestures in complex background can be another extension. Recognition of gestures made with both hands is not possible by this system. Therefore, another future work can be the recognition of gestures.

REFERNCES

[1] Bogdan Kwolek, W-Pola 2 “Hand Detection Using Convolutional Networks andGabor Filters”. Rzeszao University of Technology, Poland.

[2] N. Ozkaya, Sagiroglu,S 2008, “Intelligent Hand Sign Prediction System”, IEEE Int. Joint Conf. on Neural Networks.

[3] Gayatri Deora, Ramakrishna Godhula and Dr. Vishwas Udpikar “Study of Masked Hand Detection Approach in Video Analytics”. 2016, IEEE Conference on Advances in Signal Processing.

[4] Naveen S, Shihana Fathima R, Dr. R.S Moni, 2016 International Conference on Communication Systems and Networks.

[5] Kaihan Lin, Xiaoyong Liu, Huimin “Hand Detection and Segmentation based on Improved Mask R-CNN”, 2017.

[6] Mingjie Jiang, Xinqi Fan and Hang Yan “Retinal Hand Sign Detector”. 9 June 2020, Hong Kong University

[7] S. Meivel, K. Indira Devi, S. Uma Maheswari , J. Vijaya Menaka “Real time data analysis of Hand Sign detection and social distance measurement using Matlab”, Dec 2020.

[8] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[9] https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d