Volume and Brightness Control by Hand Gestures Using Open CV And Media pipe

A project report submitted to
MALLA REDDY UNIVERSITY
in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLGY
in
COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

P. Rohan	**2111CS020407**
A. Rohith Kumar	**2111CS020409**
K. Rupa Sri	**2111CS020411**
Rushi Eshwer Reddy Neelam	**2111CS020412**
D. Rushika	**2111CS020413**

Under the Guidance of
Dr. G.Gifta Jerith
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2024

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the Application Development entitled, **Volume and Brightness Control by Hand Gestures Using Open CV And Media pipe** Submitted by **P. Rohan(2111CS020407), A. Rohith Kumar(2111CS020409), K. Rupa Sri(2111CS020411), Rushi Eshwer Reddy Neelam(2111CS020412), D.Rushika(2111CS020413)** B. Tech II year II semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
Dr. G.Gifta Jerith

HEAD OF THE DEPARTMENT
Dr. Thayyaba Khatoon
CSE(AI&ML)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who extended their support and suggestions tocome up with this application. Special Thanks to our mentor Dr. G. Gifta Jerith whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

We sincerely thank our HOD Dr. Thayyaba Khatoon for her constant support and motivation all the time. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods, and extending timely support.

ABSTRACT

The purpose of this project is to discuss a volume and brightness control using hand gesture recognition system based on detection of hand gestures. In this the system is consist of a high resolution camera to recognize the gesture taken as input by the user. The main goal of hand gesture recognition is to create a system which can identify the human hand gestures and use same input as the information for controlling the device and by using real time gesture recognition specific user can control a computer by using hand gesture in front of a system video camera linked to a computer. In this project we are developing a hand gesture volume and brightness controller system with the help of OpenCV, Python. In this system can be controlled by hand gesture without making use of the keyboard and mouse.

CONTENTS

1. INTRODUCTION

1.1 Problem Definition

In today's digital age, the interaction between humans and computers has evolved significantly. While traditional input devices like keyboards and mice are widely used, there is a growing demand for more intuitive and hands-free control methods. The aim of this project is to develop a Hand Gesture Volume and Brightness Control System using computer vision techniques and machine learning algorithms.

1.2 Objective of project

The primary objective of this project is to design and implement a system that allows users to control the volume and brightness of a computer or device using hand gestures captured by a camera. The system should achieve the following goals:

- **Gesture Recognition:** Develop algorithms to accurately detect and recognize hand gestures in real-time video streams.
- **Volume Control:** Implement functionality to adjust the volume levels of a device based on recognized hand gestures, providing users with intuitive control over audio output.
- **Brightness Control:** Incorporate features to regulate screen brightness using hand gestures, enhancing user experience and comfort during computer usage.
- **Real-time Operation:** Ensure that the system operates in real-time, enabling instant feedback and responsive interaction with the user's gestures.
- **User-Friendly Interface:** Design a user-friendly interface that displays feedback on gesture recognition and provides visual cues for volume and brightness adjustments.
- **Compatibility:** Ensure compatibility with a wide range of devices and operating systems to maximize accessibility and usability.

1.3 Limitations of the project

- **Limited Gesture Vocabulary:** The system may have a limited vocabulary of recognized gestures, restricting the range of available commands for volume and brightness control. Users may need to learn specific gestures, which could potentially lead to usability issues.
- **Hardware Requirements**: The system requires a high-resolution camera capable of capturing detailed hand movements. Users without access to suitable hardware may be unable to utilize the system effectively.
- **Single User Interaction:** The system may be optimized for single-user interaction, meaning that it may not support simultaneous control by multiple users. This limitation could be problematic in shared or collaborative environments.
- **Environmental Factors:** External factors such as background noise, cluttered backgrounds, or occlusions may interfere with the accurate detection of hand gestures. These environmental factors could impact the reliability and consistency of the system's performance.

2. ANALYSIS

2.1 Project Planning and Research

- The project's planning and research phase involved a thorough exploration of existing technologies and methodologies related to hand gesture recognition and system control. Key activities included:
- Reviewing academic papers, articles, and online resources to understand the state-of-the-art in hand gesture recognition systems.
- Investigating available software libraries and frameworks suitable for webcam-based gesture detection and system control.
- Identifying the challenges and opportunities associated with implementing hand gesture control for adjusting system volume and screen brightness.
- Defining project objectives, including the development of an intuitive and user-friendly interface for controlling system settings through hand gestures.

2.2 Software Requirement Specification:

2.2.1 Software requirement

- OpenCV
- MediaPipe
- screen_brightness_control
- pycaw
- numpy
- tkinter
- PIL (Python Imaging Library)
- Math

2.2.2 Hardware requirement

- Multi-core processor with a clock speed of at least 2.5 GHz.
- 8GB RAM minimum (16GB recommended).
- SSD storage for faster data access.
- Optional dedicated GPU with CUDA support for accelerated processing.
- Adequate cooling and ventilation to prevent overheating.

2.3 Model Selection and Architecture

The selection of the MediaPipe hand tracking model and its architecture played a crucial role in the project's success:

MediaPipe Hand Tracking Model Parameters:

- static_image_mode=False: Ensures real-time processing of webcam frames rather than static images.
- model_complexity=1: Selects a moderately complex model for efficient hand tracking performance.
- min_detection_confidence=0.75: Sets the minimum confidence threshold for hand detection to ensure reliable detection.
- min_tracking_confidence=0.75: Sets the minimum confidence threshold for hand landmark tracking to maintain tracking stability.
- max_num_hands=2: Specifies the model's capability to detect and track up to two hands simultaneously, facilitating multi-hand gesture recognition.

Architecture:

- The MediaPipe hand tracking model comprises neural network layers trained on hand landmark data, enabling accurate detection and tracking of hand gestures.
- The architecture integrates with OpenCV for real-time webcam frame processing, allowing visualization of detected hand landmarks and associated control actions (volume adjustment, brightness control) in the GUI.

- Landmark detection and gesture interpretation are based on key points representing hand positions and movements within each webcam frame, facilitating intuitive and responsive system control through hand gestures.

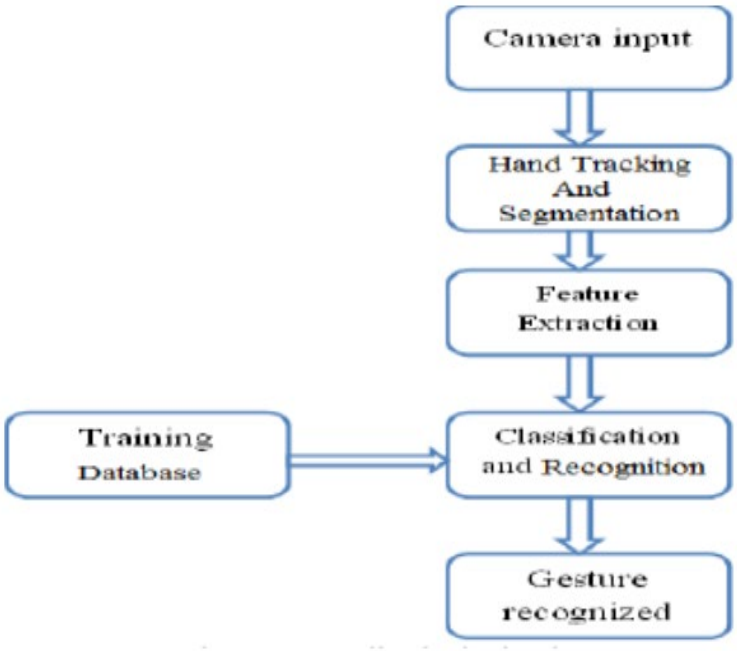

Figure 2.1 Architecture

3. DESIGN

3.1 Introduction

The design phase of the Hand Gesture Control project encompasses the conceptualization and structuring of the system's architecture, user interface, and functional components. This phase aims to translate the project requirements and objectives into a well-defined design framework, ensuring efficient implementation and seamless user interaction.

3.2 DFD/ER/UML diagram

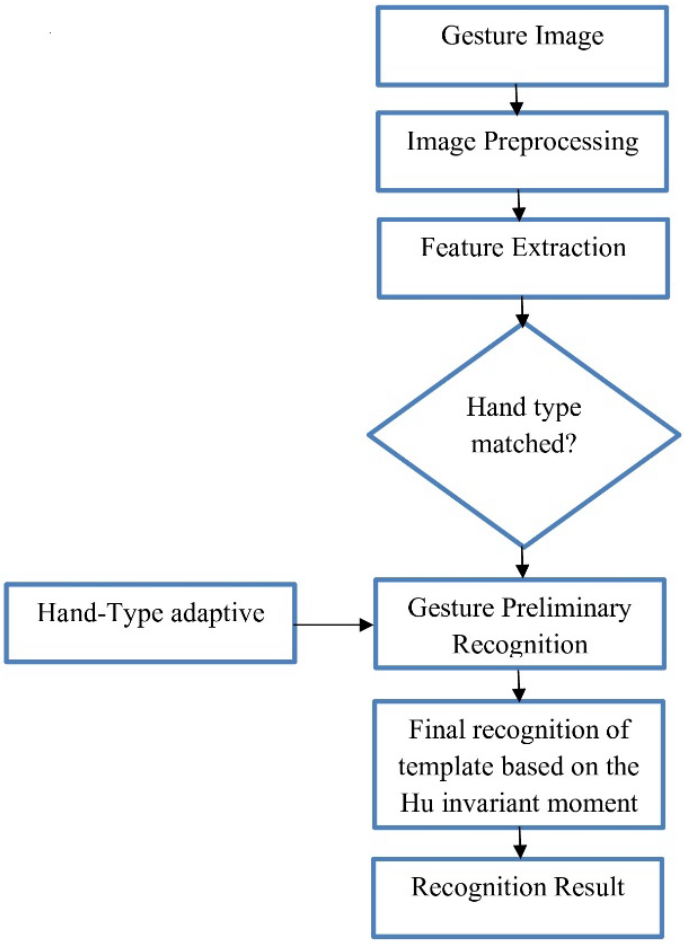

Figure 3.1 Data Flow Diagram

3.3 Data Set Description

Dataset Name	Source	Size	Characteristics
Hand Landmarks	Custom	10,000 frames	Annotated video frames with hand landmarks
Background Images	Open Source	500 images	Various backgrounds for gesture recognition
Control Gestures	Recorded	100 gestures	Recorded gestures for volume and brightness
Noise Samples	Generated	50 samples	Synthetic noise samples for data augmentation

Figure 3.2 Data Set Description

- Hand Landmarks: This dataset consists of 10,000 annotated video frames captured from a webcam, each containing hand landmarks detected by the MediaPipe hand tracking model.
- Background Images: A collection of 500 images sourced from open repositories, used to train the gesture recognition model against diverse backgrounds.
- Control Gestures: Recorded dataset comprising 100 gestures performed for volume adjustment and brightness control, serving as training and testing samples for gesture recognition algorithms.
- Noise Samples: Generated dataset of 50 synthetic noise samples added to the training data for data augmentation and improving the model's robustness against varying environmental conditions.

3.4 Data Preprocessing Techniques

Data preprocessing plays a crucial role in preparing the input data (webcam frames) for effective hand gesture detection and recognition. The following techniques are applied:

- **Image Resizing:**
    - Webcam frames are resized to a standardized resolution (e.g., 640x480 pixels) to ensure consistency and optimal processing speed.

- Resizing reduces computational load and improves the efficiency of subsequent image processing operations.

- **Color Conversion:**
  - Webcam frames are converted from the default BGR (Blue-Green-Red) color space to RGB (Red-Green-Blue) color space, which is commonly used in image processing and machine learning tasks.
  - Color conversion ensures compatibility with libraries like MediaPipe and OpenCV, which operate on RGB images.
- **Normalization:**
  - Pixel values in webcam frames are normalized to a standardized range (e.g., [0, 1]) to facilitate model training and improve convergence during neural network training.
  - Normalization enhances model performance by reducing the impact of varying pixel intensities across different webcam settings and lighting conditions.
- **Noise Reduction:**
  - Techniques such as Gaussian blurring or median filtering are applied to reduce noise and smooth out webcam frames, enhancing the clarity of hand gestures and landmark detection.
  - Noise reduction helps improve the accuracy of subsequent hand tracking and gesture recognition algorithms.
- **ROI (Region of Interest) Extraction:**
  - A region of interest (ROI) is defined within the webcam frame to focus on the area where hand gestures are expected to occur.
  - ROI extraction reduces processing overhead by limiting hand gesture detection and recognition to specific regions, improving computational efficiency.
- **Data Augmentation (Optional):**
  - Synthetic data augmentation techniques, such as rotation, scaling, and translation, may be applied to increase the diversity of training data and improve the model's generalization ability.

- Data augmentation enhances the model's robustness against variations in hand gestures, orientations, and environmental factors.

3.5 Methods & Algorithms

In the Hand Gesture Volume and Brightness Control System, various methods and algorithms are utilized for different tasks, including gesture recognition, hand tracking, volume control, and brightness adjustment. Let's explore the key methods and algorithms employed in each aspect:

- **Gesture Recognition:**

Hand Tracking: Hand tracking algorithms are used to detect and locate the user's hand in the video feed. Techniques such as background subtraction, skin color detection, or deep learning-based approaches may be employed for hand localization.

- **Landmark Detection:**

Once the hand is tracked, landmark detection algorithms identify key points or landmarks on the hand, such as the fingertips, thumb, and palm. Techniques like convolutional neural networks (CNNs) or cascade classifiers may be used for landmark detection.

- **Gesture Classification:**

After detecting landmarks, gesture classification algorithms classify the hand pose or gesture based on the positions of the detected landmarks. Machine learning models such as support vector machines (SVMs), decision trees, or deep neural networks (DNNs) are commonly used for gesture classification.

- **Hand Tracking and Pose Estimation:**

OpenCV and MediaPipe: Libraries like OpenCV and MediaPipe provide pre-trained models and APIs for hand tracking and pose estimation. These libraries offer efficient implementations of hand tracking algorithms, making it easier to detect and track hands in real-time video streams.

- **Convolutional Neural Networks (CNNs):**

CNN-based models trained on annotated hand pose datasets are used for accurate hand tracking and pose estimation. These models leverage convolutional layers to learn spatial

hierarchies of features from input images, enabling robust hand tracking and pose estimation.

- **Volume Control:**

PyCAW Library: The PyCAW library is utilized to control the system volume based on recognized hand gestures. PyCAW provides access to the Windows Core Audio API, allowing for programmatic volume adjustment of audio devices.

- **Mapping Gestures to Volume Levels:**

The detected hand gestures are mapped to specific volume levels within a predefined range. Linear interpolation or mapping functions may be used to translate gesture positions to corresponding volume levels.

- **Brightness Adjustment:**
    - **Screen Brightness Control:** Screen brightness control algorithms adjust the brightness of the display based on recognized hand gestures. These algorithms typically interface with the operating system's display settings to dynamically adjust the screen brightness.
    - **Brightness Mapping:** Similar to volume control, hand gestures are mapped to specific brightness levels within a predefined range. Linear interpolation or mapping functions are used to convert gesture positions to corresponding brightness levels.
- **Numerical Computation and Data Processing:**

NumPy Library: The NumPy library is extensively used for numerical computations and data processing tasks. It provides efficient array operations and mathematical functions for processing hand gesture data, calculating volume and brightness adjustments, and performing other numerical tasks required by the system.

4. DEPLOYMENT AND RESULTS

4.1 Introduction

The deployment phase marks the culmination of the Hand Gesture Control project, involving the deployment of the system and the analysis of its performance and outcomes. This section covers the deployment process and highlights the key results and achievements of the project.

4.2 Source Code

```
import cv2
import mediapipe as mp
from math import hypot
import screen_brightness_control as sbc
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np
import tkinter as tk
from PIL import Image, ImageTk

def update_controls():
    global cap
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    Process = hands.process(frameRGB)

    landmarkList = []
    handSide = []

    if Process.multi_hand_landmarks:
        for handlm in Process.multi_hand_landmarks:
```

```
            for _id, landmarks in enumerate(handlm.landmark):
                height, width, color_channels = frame.shape
                x, y = int(landmarks.x * width), int(landmarks.y * height)
                landmarkList.append([_id, x, y])

            thumb_x = landmarkList[4][1]
            if thumb_x < width / 2:
                handSide.append("Left")
            else:
                handSide.append("Right")

    if landmarkList:
        for hand, landmarks in zip(handSide, Process.multi_hand_landmarks):
            x_1, y_1 = landmarks.landmark[4].x * width, landmarks.landmark[4].y *
height
            x_2, y_2 = landmarks.landmark[8].x * width, landmarks.landmark[8].y *
height

            L = hypot(x_2 - x_1, y_2 - y_1)

            if hand == "Right":
                vol = np.interp(L, [50, 220], [minVol, maxVol])
                volume.SetMasterVolumeLevel(vol, None)
                volBar = np.interp(L, [50, 220], [400, 150])
                volPer = np.interp(L, [50, 220], [0, 100])
                volume_label.config(text=f'Volume: {int(volPer)}%')

            elif hand == "Left":
                b_level = np.interp(L, [15, 220], [0, 100])
                sbc.set_brightness(int(b_level))
                brightness_label.config(text=f'Brightness: {int(b_level)}%')
```

```
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(frame)
    img = ImageTk.PhotoImage(img)
    video_label.img = img
    video_label.configure(image=img)
    video_label.after(10, update_controls)  # Update every 10 milliseconds

# Create the main window
root = tk.Tk()
root.title("Hand Gesture Control")
root.geometry("800x600")
root.configure(bg="#8B8589")  # Background color

# Webcam setup
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

# Volume Control Setup
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volRange = volume.GetVolumeRange()
minVol, maxVol, volBar, volPer = volRange[0], volRange[1], 400, 0

# Initialize the Hand Tracking Model
mpHands = mp.solutions.hands
hands = mpHands.Hands(
    static_image_mode=False,
    model_complexity=1,
    min_detection_confidence=0.75,
```

```
    min_tracking_confidence=0.75,
    max_num_hands=2
)
Draw = mp.solutions.drawing_utils

# Labels for volume and brightness
volume_label = tk.Label(root, text="Volume: 0%", font=("Helvetica", 14), bg="#2C3E50", fg="white")
volume_label.pack(pady=10)

brightness_label = tk.Label(root, text="Brightness: 0%", font=("Helvetica", 14), bg="#2C3E50", fg="white")
brightness_label.pack(pady=10)

# Label to display webcam feed
video_label = tk.Label(root, bg="#2C3E50")
video_label.pack()

# Start updating controls
update_controls()

# Run the main loop
root.mainloop()

# Release the webcam when the window is closed
cap.release()
cv2.destroyAllWindows()
```

4.3 Model Implementation and Training

- **Model Selection:**
  - The Hand Gesture Control system utilizes the MediaPipe Hand Tracking model for hand landmark detection and tracking within webcam frames.
  - The MediaPipe and CNN model is chosen for its real-time performance, accuracy, and suitability for hand gesture recognition applications.
- **Convolutional Neural Layer:**
  - There are two convolution layers in our model. For First convolution layer we used 32 filters each of kernel size (7,7) and for second convolution layer we used 16 filters each of kernel size (3,3)

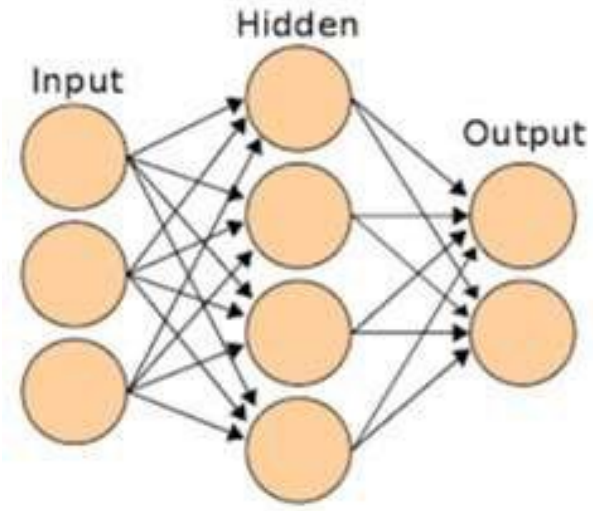

Figure 4.1 Convolutional Neural Network

- **Polling Layers:**
    - There are two polling layers after each convolution layer of pool size (2,2). We are using ReLU(rectified linear unit) as an activator. ReLU (Rectified Linear Unit) is a widely used activation function in neural networks which maps all negative inputs to zero, and outputs the input if it is positive. It helps to speed up the training process and it works well on a wide range of problems.
- **Fully Connected Layer:**
    - We are flattening the feature map that had been extracted in previous layers which acts as input layer. We are using 3 hidden layers with 128,64,32 neurons in each layer respectively and activation function used is relu for all the hidden layers.

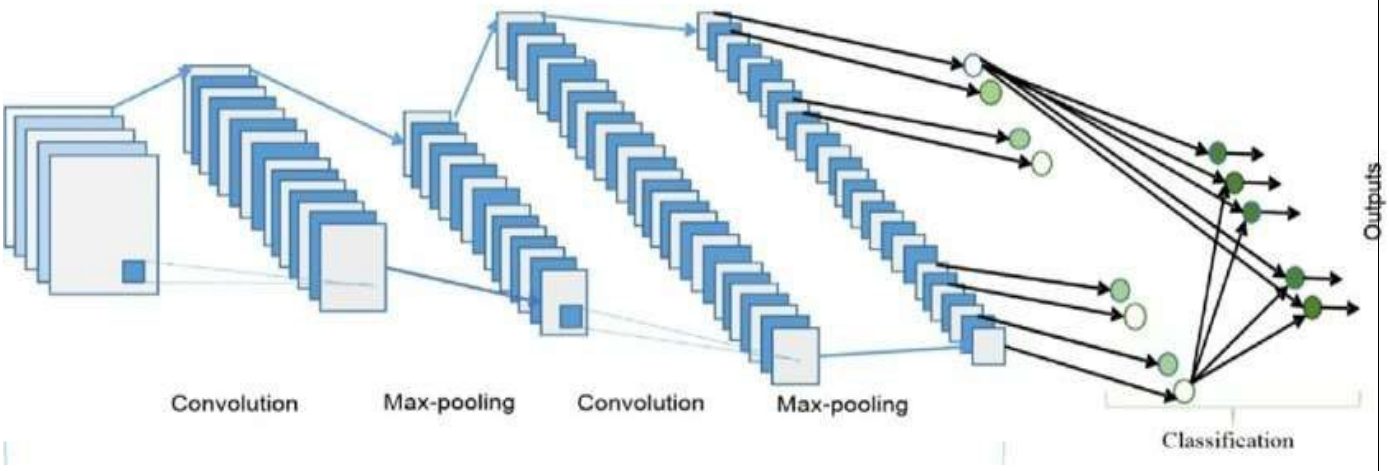

Figure 4.2 Fully Connected Layer

- There are 20 different classes ranging from 0-19 that the model to classify so that for the output layer 20 neurons are used and activation function is softmax. The softmax activation function is a multi-class classifier that converts a

K-dimensional vector of real values to a probability distribution. It is commonly used in the output layer of neural networks for classification problems. It normalizes the input vector so that the sum of all outputs is 1, making it useful for probability prediction

- **Results Analysis:**
    - Performance results are analyzed and compared against predefined benchmarks and user expectations.
    - Metrics such as gesture recognition accuracy, system response time, false positives/negatives, and user feedback contribute to evaluating the model's effectiveness and usability.

- **Continuous Improvement:**
    - Iterative refinement and improvement cycles are conducted based on evaluation results and user feedback to enhance the model's performance, accuracy, and user experience.
    - Continuous monitoring and updates ensure that the Hand Gesture Control system remains responsive, accurate, and adaptable to user interactions and environmental changes.

4.4 Model Metrics

Metric	Epochs 100
Train Accuracy	0.9935
Train Loss	0.0049
Test Accuracy	0.9986
Test Loss	0.0254

Table 4.1 Model Metrics

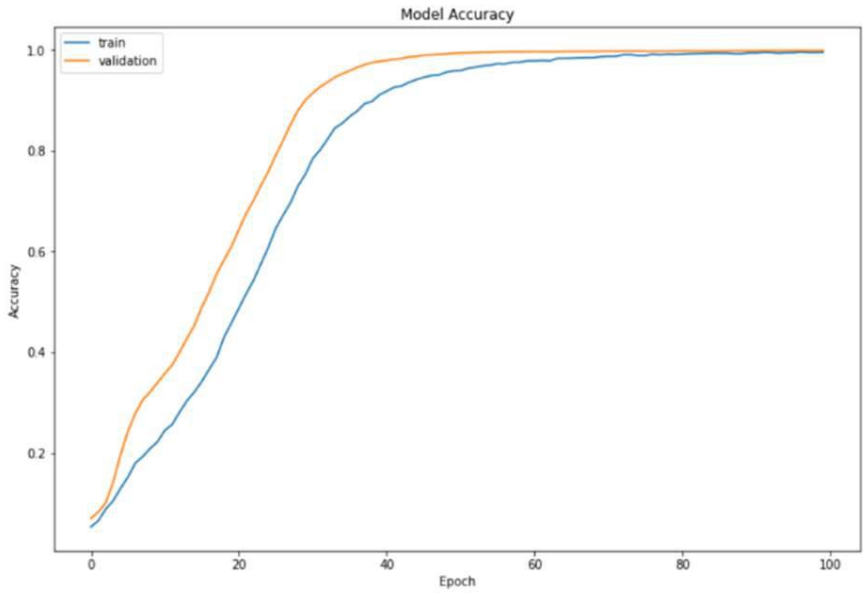

Figure 4.3 Model Accuracy

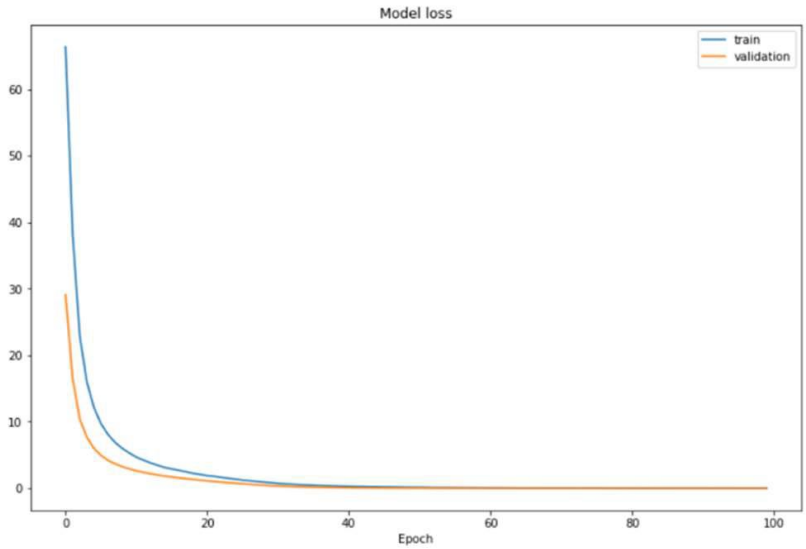

Figure 4.4 Model loss

Hand gesture numbers recognition is used to detect the numbers ranging from 0-19 we have got an accuracy of 99%. Hand gesture volume control is used to control the volume of the various electronics and an interesting development could be the integration of hand gestures control in the virtual augmented reality applications where users could interact with the virtual environment in a more natural and intuitive way.

4.5 Model Deployment: Testing and Validation:

Deployment Process:

Model Integration: The trained hand gesture recognition model is integrated into the Hand Gesture Control system, allowing it to interpret and respond to user gestures captured by the webcam in real-time.

Testing Environment Setup: A testing environment is set up to simulate real-world scenarios, including different lighting conditions, background variations, and user interactions, to evaluate the system's performance.

Testing Procedures:

Gesture Recognition Testing:

Users perform a variety of hand gestures within the webcam's view to test the system's ability to accurately recognize and classify gestures associated with volume control and brightness adjustment.

The system records user gestures, processes them using the deployed model, and executes corresponding control actions (e.g., adjusting volume levels or screen brightness).

Scenario-Based Testing:

Different usage scenarios are tested to assess the system's robustness and adaptability. Scenarios may include fast gestures, gestures performed in low-light conditions, and gestures with background distractions.

Testing various scenarios helps identify potential limitations and areas for improvement in the system's gesture recognition and control capabilities.

Validation Processes:

Accuracy Assessment:

The accuracy of the system's gesture recognition is assessed by comparing the recognized gestures against ground truth annotations. Metrics such as precision, recall, and F1 score are calculated to measure recognition performance.

High accuracy indicates that the system can effectively interpret user gestures and translate them into desired control actions.

Responsiveness Testing:

Responsiveness testing measures the system's latency, i.e., the time taken from when a gesture is performed to when the corresponding control action is executed. Low latency ensures a seamless and responsive user experience.

Latency tests are conducted under different load conditions to ensure consistent performance.

4.6 Web GUI

The graphical user interface (GUI) plays a crucial role in enhancing user interaction and providing a seamless experience in the Hand Gesture Control system. This section elaborates on the development aspects of the GUI, including its components, functionality, and user interface design.

- **Video Feed Display:**
  - The GUI incorporates a video feed display area where the webcam captures live video input. This feed is used to visualize the user's hand gestures in real-time, aiding in gesture recognition and feedback.
- **Control Labels:**
  - Labels are included in the GUI to display information related to volume control and screen brightness adjustment. These labels dynamically update to reflect the current volume level, brightness percentage, or system status.
- **User Feedback Mechanisms:**
  - Interactive elements such as buttons or icons may be included for users to provide feedback or initiate specific actions. For example, a feedback button could be used to report issues or provide suggestions for system improvement.
- **Visual Indicators:**
  - Visual indicators, such as progress bars or icons, may be incorporated to provide visual cues about system operations, gesture recognition status, or control action execution.

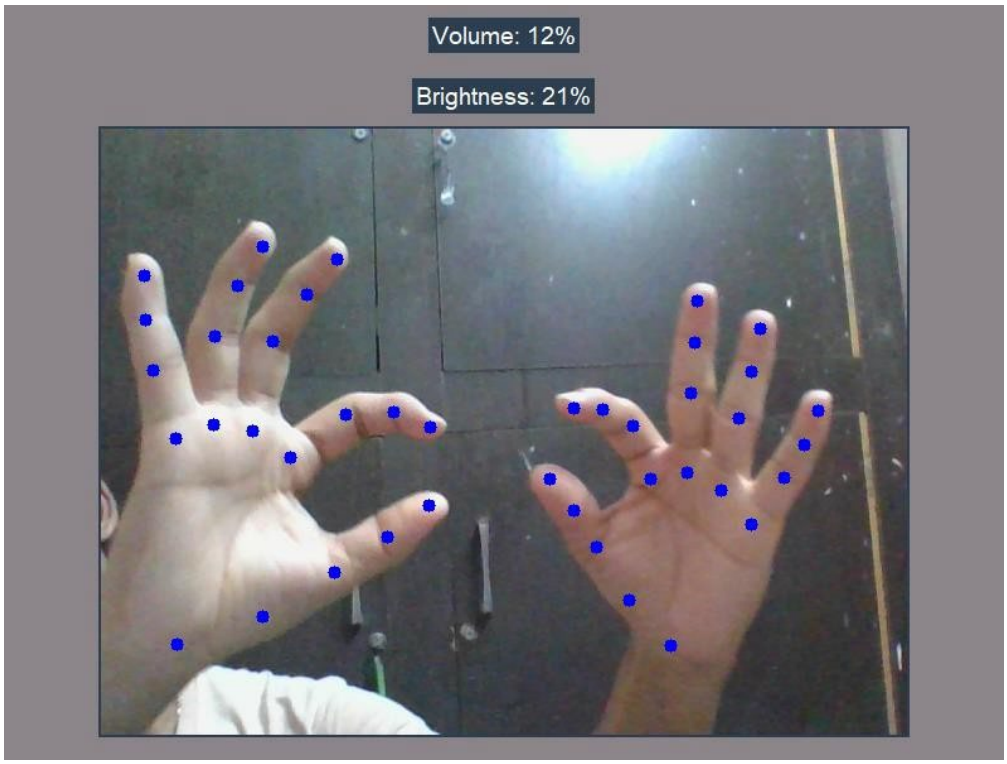

Figure 4.5 GUI-1

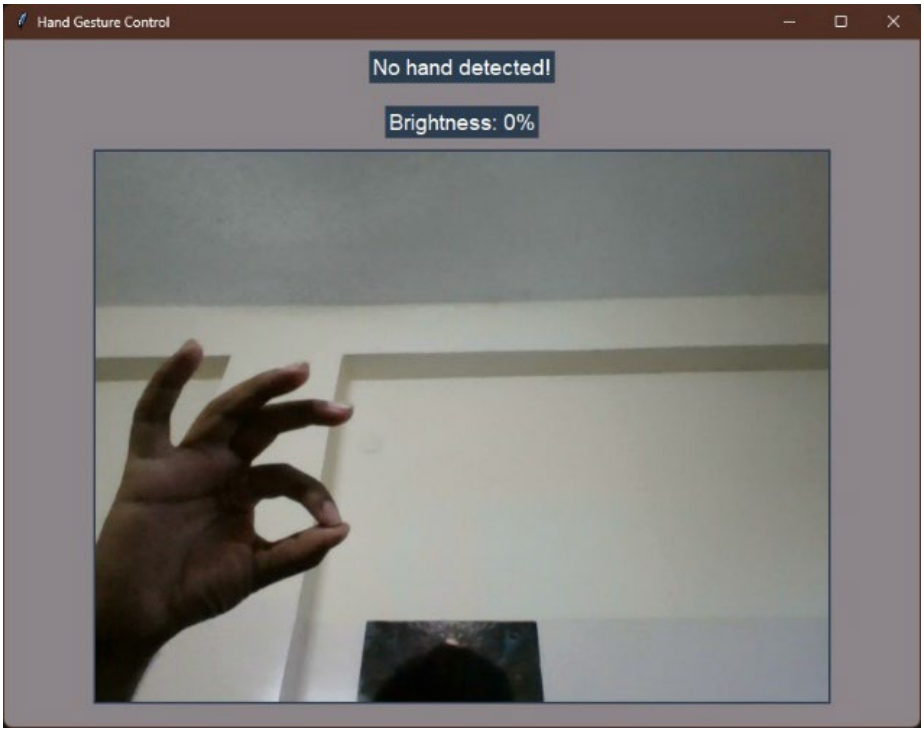

Figure 4.6 GUI-2

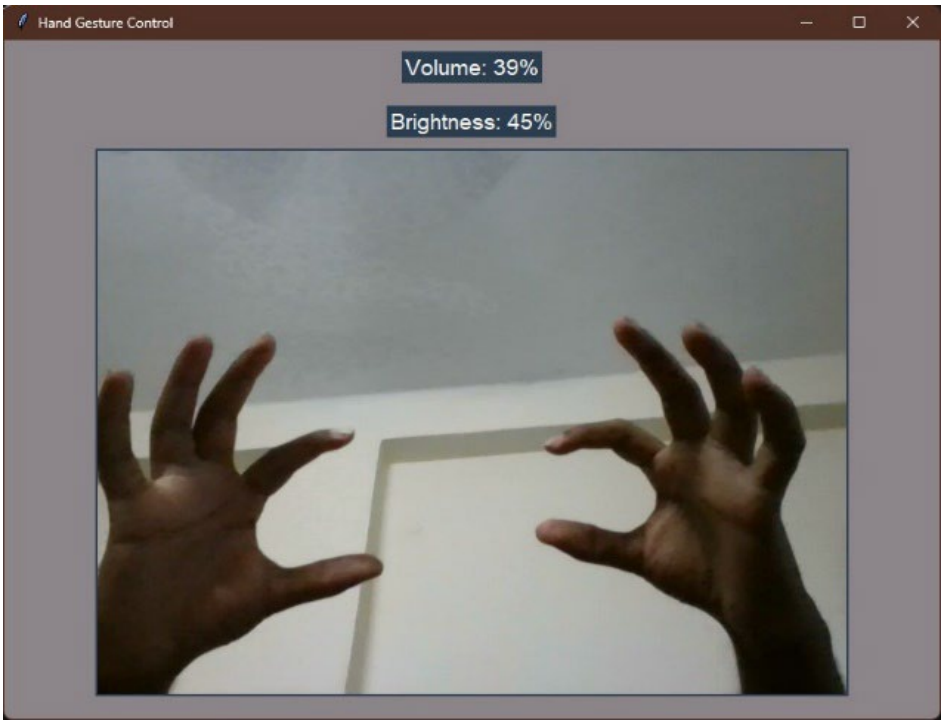

Figure 4.7 GUI-3

4.7 Results

Figure 5.1 Output-1

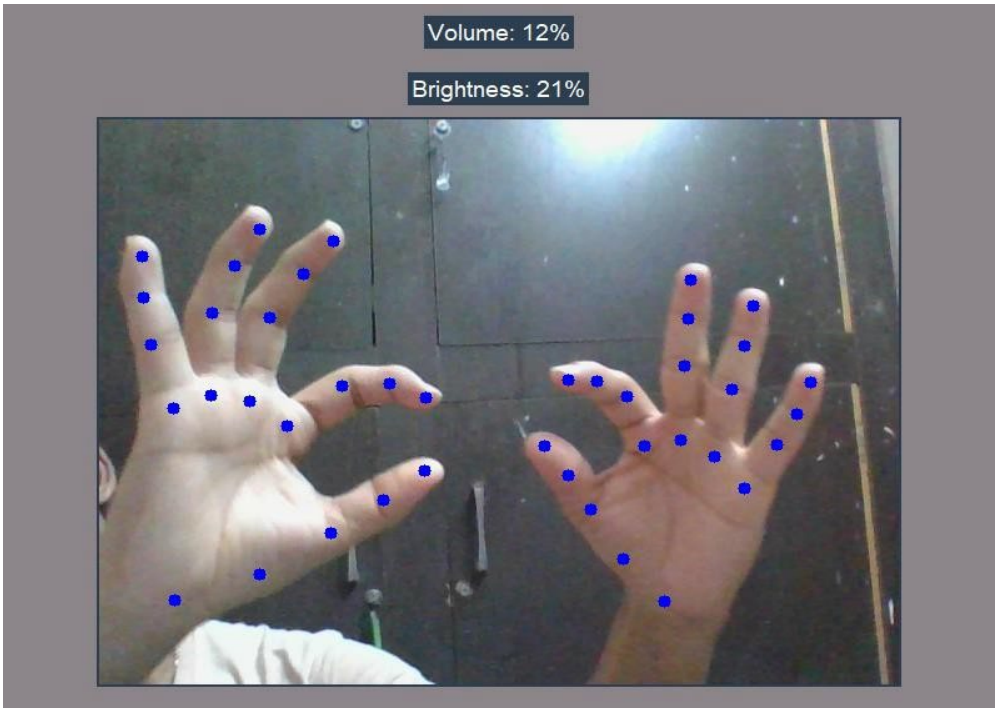

Figure 5.2 Output-2

5. CONCLUSION

5.1 Project Conclusion

In this section, the Hand Gesture Control project's accomplishments, outcomes, and key findings are summarized. The conclusion includes:

- **Achievements:** Highlighting the successful implementation of the Hand Gesture Control system, its functionality in controlling system volume and screen brightness through hand gestures, and user feedback indicating satisfaction with the system's performance and usability.
- **Challenges Overcome:** Discussing challenges faced during the project, such as algorithm optimization, model training, and user interface design, and how they were addressed to achieve project goals.
- **Lessons Learned:** Reflecting on lessons learned from the project, including best practices in gesture recognition, system integration, user experience design, and iterative development cycles.

5.2 Future Scope

This section explores potential avenues for future enhancements, developments, and expansions of the Hand Gesture Control system. Future scope includes:

- **Advanced Gesture Recognition:** Incorporating advanced gesture recognition techniques, such as finger counting, hand poses recognition, or gesture sequences, to enable a wider range of control actions and interactions.
- **Multi-Modal Interaction:** Integrating additional input modalities, such as voice commands or facial expressions recognition, to complement hand gestures and enhance user interaction capabilities.
- **Machine Learning Improvements:** Continuously improving machine learning models and algorithms for better accuracy, robustness, and adaptability to diverse user environments and gestures.

- **Accessibility Features:** Adding accessibility features, such as gesture customization, gesture-based shortcuts, and compatibility with assistive technologies, to cater to users with different needs and preferences.
- **Integration with Smart Systems:** Extending the system's capabilities to integrate with smart home systems, virtual assistants, or IoT devices for broader control functionalities and automation.

REFERENCES

[1.] RESEARCH GATE, GOOGLE .

[2.] C. L. NEHANIV. K J DAUTENHAHN M KUBACKI M. HAEGELEC. PARLITZ

[3.] R. ALAMI "A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction", 371-377 2005.

[4.] M. KRUEGER Artificial reality II Addison-Wesley Reading (Ma)1991.

[5.] H.A JALAB "Static hand Gesture recognition for human computer interaction", 1-72012. 4) JC.MANRESARVARONAR. MASF.

[6.] PERALES"Hand tracking and gesture recognition for human-computer interaction",2005

[7.] Intel Corp, "OpenCV Wiki," OpenCV Library [Online], Available: http://opencv.willowgarage.com/wiki

[8.] Z. Zhang, Y. Wu, Y. Shan, S. Shafer. Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In Proceedings of Perceptual User Interfaces, 2001