Hand Gesture Recognition Using Deep Learning

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science & Engineering

Submitted by

Kshitiz Bashyal (201306)

Under the guidance & supervision of

Dr. Diksha Hooda

Department of Computer Science & Engineering and

Information Technology

Jaypee University of Information Technology,

Waknaghat,

Solan - 173234 (India)

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **'Hand Gesture Recognition Using Deep Learning'** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Diksha Hooda** (Assistant Professor(SG) , Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Kshitiz Bashyal
201306

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Diksha Hooda
Assistant Professor (SG)
Computer Science Engineering and Information Technology
Dated:

CERTIFICATE

This is to certify that the work which is being presented in the project report **titled 'GESTURE RECOGNITION USING DEEP LEARNING'** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Kshitiz Bashyal during the period from August 2023 to May 2024 under the supervision of **Dr. Diksha Hooda** , Assistant Professor (SG), Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Kshitiz Bashyal
201306

The above statement made is correct to the best of our knowledge.

Dr. Diksha Hooda
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat
Dated:

Acknowledgement

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing which made it possible to complete the project work successfully.

We are grateful and wish our profound gratitude to Supervisor **Dr. Diksha Hooda, Assistant Professor**, Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervisor in the field of **"Artificial Intelligence"** to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, and valuable advice have made it possible to complete this project.

The in-time facilities provided by the Computer Science department throughout the project development are also equally acknowledgeable.

Last but not the least, our sincere thanks to all our teachers and friends who have helped us straightforwardly or in a roundabout way in making this project a win.

Kshitiz Bashyal
201306

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In the dynamic realm of technology-driven interactions, the demand for swift responses and streamlined operations has intensified. As digital experiences advance, focus shifts to innovative avenues of human-computer interaction beyond conventional methods. This project explores gesture recognition, leveraging the innate and instinctive nature of human gestures in day-to-day interactions. Through computer vision and deep learning, it transforms real-time hand movements into a sophisticated control mechanism for media players.

Gesture-based communication establishes a new standard of user interaction, freeing individuals from the constraints of physical input devices. By defining seven distinct gestures, the project empowers users to seamlessly leverage their local device cameras, eliminating the need for additional hardware. This innovative approach enhances operational efficiency and enables users to exert control over laptops or desktops from a distance. The amalgamation of cutting-edge technology and intuitive human gestures addresses the evolving demands of interaction, serving as a testament to the transformative potential of gesture recognition in reshaping our digital interfaces.

In this project, we focus on revolutionizing human-computer interaction through gesture recognition. The system, utilizing computer vision and deep learning, translates real-time hand movements into a sophisticated control mechanism for media players. With seven defined gestures, users can effortlessly leverage their local device cameras, eliminating the need for additional hardware. This approach augments operational efficiency and introduces a paradigm shift, allowing users to exert control over laptops or desktops from a distance. Adaptive learning mechanisms within the gesture recognition system ensure a personalized experience, continuously refining its understanding of individual gestures.

CHAPTER 1: INTRODUCTION

1.1 GENERAL INTRODUCTION

Advancements in technology have propelled society through a series of revolutions, particularly in the realm of computing. Human-Computer Interaction (HCI) stands as a crucial field, seamlessly integrating human abilities with the technical understanding of hardware and software technologies. This evolution has reshaped tasks, transformed complex calculations and wrote processes into effortless endeavors through the use of computers.

As HCI rapidly evolves, emerging concepts like Gesture Recognition take center stage. Gestures, inherent and natural to all, serve as a non-cognitive computing interface allowing devices to capture and interpret human gestures as commands. This paper explores the profound applications of gesture recognition technology, ranging from virtual reality environment control to drowsiness detection in drivers.

Despite the significant developments, challenges persist, such as the cost and inconvenience associated with glove-based systems. Neural networks, employing color segmentation and morphological operations, have proven efficient in addressing issues like noise in the region of interest and image background. Gestures have found applications across diverse fields, including entertainment, healthcare, and disaster relief.

This paper introduces a groundbreaking approach to HCI by focusing on controlling media players with hand gestures, utilizing Convolutional Neural Networks (CNN). The proposed system offers touch-free and remote-free control, enhancing user experience and providing a productive solution for scenarios like watching movies or tutorials.

1.2 PROBLEM STATEMENT

Introduction

In the evolving landscape of Human-Computer Interaction (HCI), the demand for rapid and intuitive interactions with complex systems is escalating. As technology progresses, achieving swift response times and user-friendly operations becomes imperative. This project delves into this necessity by harnessing computer vision and deep learning techniques to facilitate real-time control of media players using hand gestures.

Challenges in Gesture Recognition

While gesture recognition provides a natural and instinctive means of communication, existing challenges impede its seamless integration. Traditional methods, such as glove-based systems, prove to be expensive and cumbersome. Moreover, persistent issues like noise interference in the region of interest and slow computation limit the efficiency of gesture recognition technologies.

Proposed Solution

The proposed solution revolves around a web application that employs computer vision and Convolutional Neural Networks (CNN) to detect and interpret user hand movements for media player control. A dataset comprising seven defined gestures serves as the foundation for training the CNN model. The integration of PyAutoGUI ensures the seamless mapping of gestures to keyboard controls, empowering users to interact effortlessly with media players.

Further Innovations and User Experience

The integration of gesture recognition technology extends beyond traditional applications, promising innovative solutions for various sectors. This project not only addresses the current challenges in media player control but also opens avenues for broader applications enhancing user experiences across diverse domains.

1.3 OBJECTIVES`

The primary aim of this project is to develop an advanced system facilitating media player control through hand gestures, leveraging cutting-edge technologies in computer vision and deep learning. The overarching goal is to redefine user interaction with media players, offering a touch-free and remote-free control mechanism. The project focuses on the following key objectives:

1) Data Collection and Preprocessing

To perform real-world image data collection and implement data preprocessing techniques to enhance the accuracy of the Hand Gesture Recognition Model.

2) To design and create a Gesture Recognition Model

Design and create a sophisticated Convolutional Neural Network (CNN) model.
Utilize the custom-built dataset, encompassing seven precisely defined gestures and to train the model for real-time gesture recognition.

3) To perform Testing and Validation:

Conduct thorough testing and validation procedures on the developed gesture recognition model. Evaluate the model's accuracy, precision, and recall to ensure its reliability and effectiveness in diverse scenarios.

4) Develop an Interactive Web Application:

Create an intuitive and user-friendly web application using Stream lit to integrate the gesture recognition model. This application will include functionalities for starting the webcam, real-time gesture recognition, and controlling a media player based on recognized gestures, thereby providing an immersive and interactive user experience.

1.4 SIGNIFICANCE AND MOTIVATION

1.4.1 Significance of Gesture-Recognition:

Gesture-controlled media players offer a novel and intuitive way for users to interact with their devices, providing a more engaging and immersive experience.

This technology caters to users with physical limitations, providing an accessible means of controlling media players without the need for traditional input devices.

By eliminating the need for physical remotes or keyboards, gesture control streamlines the interaction process, offering users a more efficient and convenient way to manage media playback.

The versatility of gesture control extends its applications beyond media players, potentially influencing advancements in fields such as healthcare, gaming, and smart home automation.

1.4.2 Motivation for Gesture Recognition Research:

The motivation behind gesture recognition research lies in creating user-centric technologies that align with the evolving preferences and expectations of modern users.

Advancing gesture recognition technology contributes to the overall innovation landscape, fostering the development of interactive systems that go beyond traditional input methods.

Research in gesture recognition aims to future-proof human-computer interaction, anticipating and addressing the changing ways users engage with technology.

The motivation extends to exploring and expanding the application domains of gesture recognition, unlocking new possibilities for human-machine collaboration.

1.5 ORGANIZATION OF PROJECT REPORT

Chapter 1: Introduction

In the introduction chapter, focus on providing an overview of the project, its significance, and the problem it aims to address. It includes background, problem statement, objectives and significance and motivation of the project work.

Chapter 2: Literature Review

Gave a thorough examination of previous studies and publications that are relevant to the project in the literature review chapter. It contains critical analysis, a review of related work, and applicability to ongoing initiatives, key findings and gaps.

Chapter 3: System Development

The technical aspects of project are covered in detail in this chapter. It includes the system architecture, which highlights the key parts and how they work together, the methodology used to design the system, and technical information about how the process was done.

Chapter 4: Testing

This section pays close attention to the project's testing phase. It outlines the testing procedures used as well as the standards for judging a test's performance. The results of the testing phase are however shown in the results chapter. Continuous testing and adjustments are made to the system to guarantee optimal performance and minimize false positives.

Chapter 5: Result and Evaluation

The project's success is assessed and the overall outcomes are presented in this chapter. It talks about how well the project accomplishes its goals and shows the metrics that are used to assess the system's performance with the pre-processed data.

Chapter 6: Conclusion and Future Scope

Summarized the main conclusions of the paper and suggested possible directions for future research. Described the project's primary results and accomplishments. Provided a final assessment of the project's success. Stressed how the initiative advances the field. Made suggestions for future project enhancements or extensions. Talked about potential expansions and applications for the project in various settings.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

2.1.1 introduction

Deep learning-based hand gesture recognition is an exciting field in human-computer interaction that could have far reaching application areas spanning from virtual reality control through assistive systems. Deep learning approaches will be exploited in this project to further the field of hand gesture recognition. Hand gestures constitute an integral part of non-verbal communication. Therefore, accurate recognition of these movements is necessary for smooth interaction between man and machine.

It seeks to explore the literature on these efforts with different systems for recognizing hand gestures. The understanding of the historical background and development of these approaches provides a good base for grasping the issues and perspectives associated with the contemporary usage deep learning implementations in this realm. This encompasses feature extraction methods, factors regarding datasets, and the effectiveness of various deep learning networks including CNNs and RNNs});

This literature review aims at bringing together different conclusions made by previous researchers so as to reveal what remains unanswered regarding the main issue under study that creates the necessity for the proposed project. Overall, the purpose of this study is to provide an extensive overview of contemporary Hand Gesture Recognition through Deep Learning towards laying the way forward on approach and implementing phase.

2.1.1 Summary of relevant papers

S.No.	Paper Title	Journal/Conference	Tools/Techniques	Result	Limitations
1.	Accuracy Enhancement of Hand Gesture Recognition Using CNN	IEEE (2023)	(IR-UWB) RADAR, Double Parallel CNN model with 2D FFT. Custom Data Set of 5 gestures obtained from IR-UWB radar.	CASE 1-Accuracy: 95.20% CASE 2-Accuracy: 86.00%	Small data set. Does not address real time implementation.
2.	Hand Gesture Recognition Using Automatic Feature Extraction and Deep Learning Algorithms	MDPI [2023]	NumPy, Pandas, Sklearn CNN, Python. Leap Gesture DB dataset and the RIT dataset.	Average recognition rate: 90%	Limited gesture vocabulary. High computational cost
3.	Sign Language and Hand gesture recommended system	IJCRT [2023]	TensorFlow OpenCV Keras NumPy good quality Camera. American Sign Language (ASL) dataset.	Accurate sign language and hand gesture recognition. [Does not mentions numerical results]	Need for improvement in recognition accuracy, latency reduction for real-time applications.
4.	Real-Time Multiple Gesture Recognition :	IEEE (2022)	(EMG) signal acquisition device. 1D CNN model, Py Torch version	Accuracy: 89.96 Training Time: 14 min 13	Low accuracy of model to counter performance

	Application Of a Lightweight Individualize 1D CNN Model		1.4.0 SIAT database	seconds	overhead in wearable devices.
5.	Gesture Recognition Using Deep Learning Techniques	Research Square [2022]	Python, TensorFlow , Keras, OpenCV, Custom CNN model. Custom dataset of 2400 images of hand gestures.	Accuracy: 99%	Tiny amount of training dataset and limited number of trainable parameters.
6.	A Review of the Hand Gesture Recognition System: Current Progress and Future Directions	IEEE (2021)	3D CNN'S, PCA Convolutional self-organizing Map RWTH-PHOENIX-Weather-2012	CASE 1- Signer Dependent Accuracy: 92% CASE 2- Signer Independent Accuracy: 77%	The existing studies focused on recognizing isolated gestures, which have limited use.

7.	Deep learning based static hand gesture	Indonesian Journal of Electrical Engineering and Computer Science [2020]	Intel Real Sense D435 depth camera, OpenCV, TensorFlow & Keras Custom dataset of 2000 images of static hand gestures.	Average: Accuracy 95.5%	Limited to static hand gestures; does not address dynamic gestures. Dataset size may be relatively small.
8.	Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation	IEEE (2019)	OpenCV, Spyder IDE, PIL SK learn, CNN classifier. Hand Gesture Recognition Database.	With augmentation Accuracy: 97.12% Without augmentation Accuracy: 92.87%	Recognition of gestures made with both hands are not possible.

In [1] Gyutae Park, Jinhwan Koh, celebrating its one-year anniversary, delves into the enhancement of accuracy in hand gesture recognition algorithms for human-machine interactions. The innovative approach presented in the paper combines 2D-FFT and convolutional neural networks (CNN) to analyze image data obtained through Ultra-Wide Bandwidth (UWB) radar. Notably, the classification results of this proposed method demonstrated comparable accuracy to well-established models, all while requiring less time for training. The paper underscores the significance of preprocessing techniques, particularly 2D-FFT, in augmenting accuracy, although not all preprocessing methods proved equally effective. Furthermore, the study reveals that prominent models boasting a substantial number of layers incurred longer execution times when compared to the efficiency of the proposed model. The experimental setup involved the utilization of IR- UWB radar equipped with two Vivaldi antennas and a bandwidth spanning from 6.0 to 8.5 GHz. In addition to presenting classification results for prominent CNN models, the research paper includes a confusion matrix comparing the Double Parallel CNN model with ResNet- 50, which exhibited the highest accuracy among the models examined. As the paper

commemorates its one-year milestone, it stands as a testament to the ongoing pursuit of advancements in gesture recognition technologies.

In [2] Rubén E. Nogales * and Marco E. Benalcázar presents, The hand gesture recognition is considered as a highly dimensional pattern recognition problem attracting attention of researchers. It is employed to show feelings of communicate to other people/machines. Hand gesture recognition based on manual and computer-based feature extraction is proposed in this paper. It uses a set of statistical functions of central tendency, whereas automatic feature extraction is based on combining both CNN and BiLSTM. The classifiers that may be used include SoftMax, ANN, and SVM to evaluate these features.

The most effective model employed in the analysis is the combination of BiLSTM and ANN (BiLSTM-ANN) with an accuracy score of 0.999912

This work also considers the use of matrices to represent fingertip directions as well as positions in space. The study also outlines the assessment of feature extractions schemes with different classifiers (ANN, SVM and SoftMax).
Determining whether or not these feature extraction methods result in significant differences.

In [3] Mrinal M Prasad, Rejeesh R… presents special attention to the use of hand gestures, signs as well as in developing a real time visual based system for recognizing signs for human computer interface.
Deaf persons use sign language as a crucial mode of communication; however, they are usually inhibited when communicating with hearing people.
Gesture recognition has a significant impact on the speed of communication between individual and machine as well as system effectiveness in understanding users' intents. Natural interaction requires seamless and continuous dialogue between people, the system and the environment so as to support continuous gesture recognition process.
Gesture recognition systems can be used in robotics, virtual reality, games, smart homes and for IoT.
Two important tools involved in developing a system for hand gesture recognition are TensorFlow and OpenCV. TensorFlow is a dataflow and differentiable programming library, while Open-Source Computer Vision (OpenCV) provides a set of ready-to use functions for real-time applications in computer vision.
Steps of developing a hand gesture recognition system include training and testing a model. Finally, it reports the accuracy of the model which is obtained by using the training data and the expected output for the training.

In [4] distinguished 21 different types of movements; researchers create an intelligent, wearable device based on eight channels of electromyography (EMG) signals. To identify the EMG signals, an integrated EMG signal acquisition device with an elastic armband was made, and an analog front end (AFE) integrated chip (IC) was created. We built and trained a lightweight 1D CNN model with a database of 21 gestures that were gathered from 10 volunteers. The average model training duration was 14 minutes, and the maximum signal recognition accuracy was 89.96%. The gesture identification accuracy matrix revealed irregular data distribution among participants, with no discernible convergence in the accuracy of gesture type recognition among participants. Wrist movements (gestures 1-4) had greater accuracy because the associated channel had a significant signal response.

In [5] Shubham Shukla. presents, to reduce latency and increase performance, future research in gesture recognition systems should concentrate on optimizing the processing pipeline through edge computing, parallel processing, or hardware acceleration.

For natural and continuous interaction, continuous gesture recognition is essential. Therefore, future systems should try to recognize dynamic, continuous sign language motions while capturing temporal aspects and gesture variations.

Widespread accessibility and utilization can be ensured by using sign language and gesture detection systems in real-world contexts, especially for people with language or hearing impairments. Accessibility can be improved by integration with virtual reality systems, robots, assistive technology, and communication devices.

It is possible to enhance gesture recognition systems to provide complex gesture-based control in a number of interactive systems, such as virtual reality, robotics, gaming, smart homes, and the Internet.

In [6] Noraini Mohamed, Mumtaz Begum Mustafa. presents, Hardware acceleration, parallel processing, or edge computing should be the main goals of gesture recognition systems in order to increase processing speed and decrease latency.

Future systems should strive to identify dynamic, continuous sign language motions as well as temporal components and gesture alterations.

It is possible to improve the accessibility and usability of sign language and gesture detection systems through integration with assistive technology, communication devices, virtual reality systems, or robots.

Internet of Things (IoT) platforms, robots, virtual reality, gaming, and smart homes are among the industries where gesture recognition systems may find use.

Augmenting vision, depth, audio, and wearable sensors with other modalities can boost the precision and resilience of gesture recognition systems. Accurate identification can also be improved by user customization and adaptation strategies.

In [7] Md. Zahirul Islam, Mohammad Shahadat Hossain… presents Hardware acceleration, parallel processing, or edge computing should be the main goals of gesture recognition systems in order to increase processing speed and decrease latency. Future systems should strive to identify dynamic, continuous sign language motions as well as temporal components and gesture alterations. It is possible to improve the accessibility and usability of sign language and gesture detection systems through integration with assistive technology, communication devices, virtual reality systems, or robots. Internet of Things (IoT) platforms, robots, virtual reality, gaming, and smart homes are among the industries where gesture recognition systems may find use. Augmenting vision, depth, audio, and wearable sensors with other modalities might increase the precision and resilience of gesture recognition systems.

In [8] Dina Satybaldina, Gulzia Kalymova presents, Hardware acceleration, parallel processing, or edge computing should be the main goals of gesture recognition systems in order to increase processing speed and decrease latency. Future systems should strive to identify dynamic, continuous sign language motions as well as temporal components and gesture alterations. It is possible to improve the accessibility and usability of sign language and gesture detection systems through integration with assistive technology, communication devices, virtual reality systems, or robots. Internet of Things (IoT) platforms, robots, virtual reality, gaming, and smart homes are among the industries where gesture recognition systems may find use. Augmenting vision, depth, audio, and wearable sensors with other modalities can boost the precision and resilience of gesture recognition systems.

2.2 Key gaps of the literature

- There are issues with latency and processing speed in gesture recognition systems.
- Recognizing dynamic, continuous sign language motions requires improvement.
- Improved accessibility and user-friendliness are necessary for gesture detection systems.
- Further study is required on gesture detection systems across different fields.
- Boost the precision of systems that recognize gestures.
- Publications contain little information about dataset representation.
- The effectiveness of the assessment metrics for CNN model performance needs to be verified.
- The effectiveness of the assessment metrics for CNN model performance needs to be verified.
- Issues related to integration with current technologies and infrastructures across various application domains.
- Improving gesture recognition systems' energy efficiency, especially for mobile devices in environments with limited resources.
- It's possible that systems won't be able to customize gesture recognition according to unique user preferences and routines.
- Recognizing and appropriately interpreting gestures, particularly in situations that are subtle or complex.

Chapter 3: SYSTEM DEVELOPMENT

3.1 Requirements and Analysis

The proposed gesture recognition using deep learning project needs to be designed in such a way that it is accurate while controlling any media through gestures. So, we have to contemplate various requirements, and we have to make sure that these requirements are met completely, and apart from this we have to scan thoroughly each and every potential risks involved. Some key requirements and analysis that we must need to consider are: -

Functional Requirements:

Gesture Recognition:

The system must accurately recognize seven predefined hand gestures in real-time.

- It should be able to differentiate between gestures to perform specific actions. Media Player Control:
- The recognized gestures should be mapped to control commands for a media player.
- Controls include play/pause, mute/unmute, volume up/down, forward, and rewind.

Web Application:

- A web application must be provided for user interaction.
- The application should have options to start the webcam, display real-time predictions, and exit the system.

User Interface:

- The web application should provide a user-friendly interface to guide users on how to interact with the system.
- Clear instructions and feedback should be provided during gesture recognition.

Non-functional Requirements:

Accuracy:

- The system should achieve a high accuracy level, as mentioned in the experimental results (98%).

Latency:

- The system should exhibit low to negligible latency for real-time interaction.

- Response time for recognizing gestures and controlling the media player should be minimal.

Adaptability:

- The system should be adaptable to different environmental conditions, including lighting variations.

Hardware Requirements:

Webcam:

- A functional webcam on the user's device for capturing real-time hand gestures.

Software Requirements:

Programming Language:

- The system is implemented using a programming language, not explicitly mentioned. Further details are needed.

Libraries:

- OpenCV: Used for image capture and preprocessing.
- Keras: Utilized for building and training the Convolutional Neural Network (CNN).
- PyAutoGUI: Employed for integrating keyboard controls with predicted gestures.
- Streamlit: Used for creating the web application interface.

3.2 Project Design and Architecture

Gesture Recognition System involves data acquisition, data pre-processing, feature extraction and finally classification for the sake of having an immersive and interactive system.

1. **Data Acquisition:**

   Then it captures each frame which gets converted from continuous videos to individual ones. The ability to acquire real-time data is vital as this forms the foundation of gesture analysis and interpretation that will follow.

2. **Pre-processing:**

   The black and white conversion of the frames is done through using OpenCV as a tool for this process. It makes the visual data simpler by emphasizing on important aspects but also eliminating clutter and distraction.

3. **Feature Extraction:**

   Accurate gesture recognition requires extracting appropriate features from the processed frames. The third stage involves extracting vital features or characteristics from the generated black and white images, which will eventually be used as input into the classifier model.

4. **Classification:**

   A model made with the Keras library is crucial for identifying these extracted features with certain hand gestures. The evaluation of the test accuracy confirms the model's ability to give accurate predictions with the processed data.

5. **Integration with Control Functions:**

   PyAutoGUI enables the incorporation of the identified and classified gestures with the respective control functions. In this part, the identified gestures are transmitted into instructions that direct the multimedia player thus providing a smooth interaction experience.

6. Web App Integration using Streamlit: Set up Streamlit by installing it using pip and create the application with three pages: About, Project Demo Video, and Gesture Recognition. Develop the "About" page to describe the system, the "Project Demo Video" page to showcase a demonstration video, and the "Gesture Recognition" page to allow users to start the webcam and use the system. The main script navigates between pages and integrates the webcam feed and gesture recognition model. The application runs with a command to launch Streamlit, enabling an interactive web application for controlling a media player through hand gestures.

In summary, this Gesture Recognition System integrates different technologies and libraries to develop a sound and people-friendly way of playing music remotely without using hands.

There are two main steps to the suggested gesture detection method for controlling media players. Hand motions are detected in the first step, and in the second stage, each gesture is linked with the keyboard commands. Using a specially created dataset,

computer vision and deep learning techniques are used to implement the gesture recognition.

The image frames are fed to three convolution layers with a ReLU activation function in order to train the CNN model. Pooling layers are added to achieve max pooling, which is subsequently flattened and added to dense layers.

A. Proposed Workflow

The system is divided into five modules. Fig. 3.1 shows the overview of the system workflow, emphasizing the gestures and their corresponding controls.

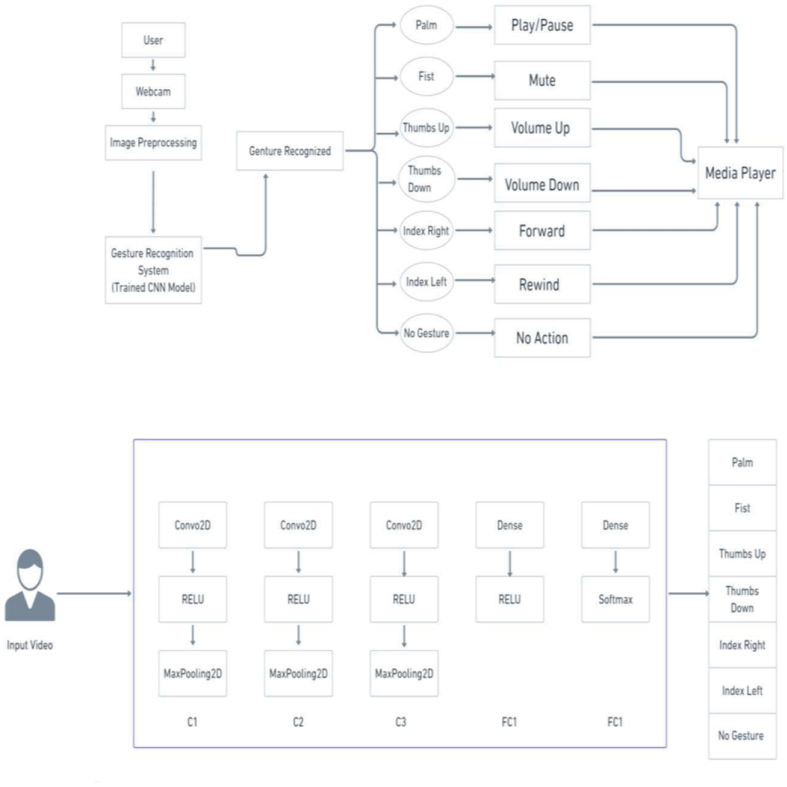

Fig 3.1[2] System design workflow

1) Obtaining and Pre-Processing images when a user makes hand motions in front of the camera. OpenCV is used to gather the picture frames from the live video. To increase the accuracy of gesture prediction, these images are transformed into black and white, as illustrated in Figure 5, and are subsequently saved in the appropriate directories. Three distinct people's gestures are gathered for this project. For every gesture, there are 150 images in the dataset. To store images, a directory structure is made. For user convenience, there are two modes available: train and test. The option

to select any one of the modes is provided to the user. The model is trained using the images acquired in train mode, and its accuracy is tested using the images obtained in test mode. The user-inputted directory determines where the images are kept. When the camera is turned on, two frames are shown on the screen, and the user can use the read function to take picture by picture while simulating the mirror image. The user must make the gestures while placing their hand inside the bounding box, or Region Of Interest (ROI). After being taken out of ROI, the frames are resized to 120x120x1.

2) The Extraction of Features Import the hidden layers and Keras models needed to construct convolutional networks. A hidden input layer and two convolution layers are used in the construction of the CNN model. After each convolution layer, a pooling layer called MaxPooling and an activation operation called ReLU are added. Two fully connected layers are added, one for classification using SoftMax activation and the other with RELU activation. A flattening layer is also added. The CNN model's architecture, which is utilized for gesture classification and feature extraction, is depicted in Fig. 6. Finally, the CNN model is compiled by using the Adam algorithm as an optimizer and categorical cross entropy as a loss function technique to determine accuracy and error as performance metrics.

3) Training the model.

The model is then trained and validated using batches of images generated by the ImageDataGenerator class. The model is trained using the fit function over a predetermined number of epochs. Using the save weights function, the weights are saved straight from the model after training, and the trained model is saved in JSON format.

Controlling the media with anticipated Hand motions

To predict hand gestures, the model weights and trained model in JSON format are loaded.

The keyboard key used with PyAutoGUI

Controlling the media with anticipated Hand motions

To predict hand gestures, the model weights and trained model in JSON format are loaded.

The keyboard key used with PyAutoGUI

B. Architecture

Any gesture recognition system's outline typically includes the following three elements:

1. Gathering and preprocessing data

2. Representing data or extracting features

3. Organizing or creating decisions

Fig. 3.2 shows the suggested system design. It has all three of the essential components needed for a gesture recognition system.

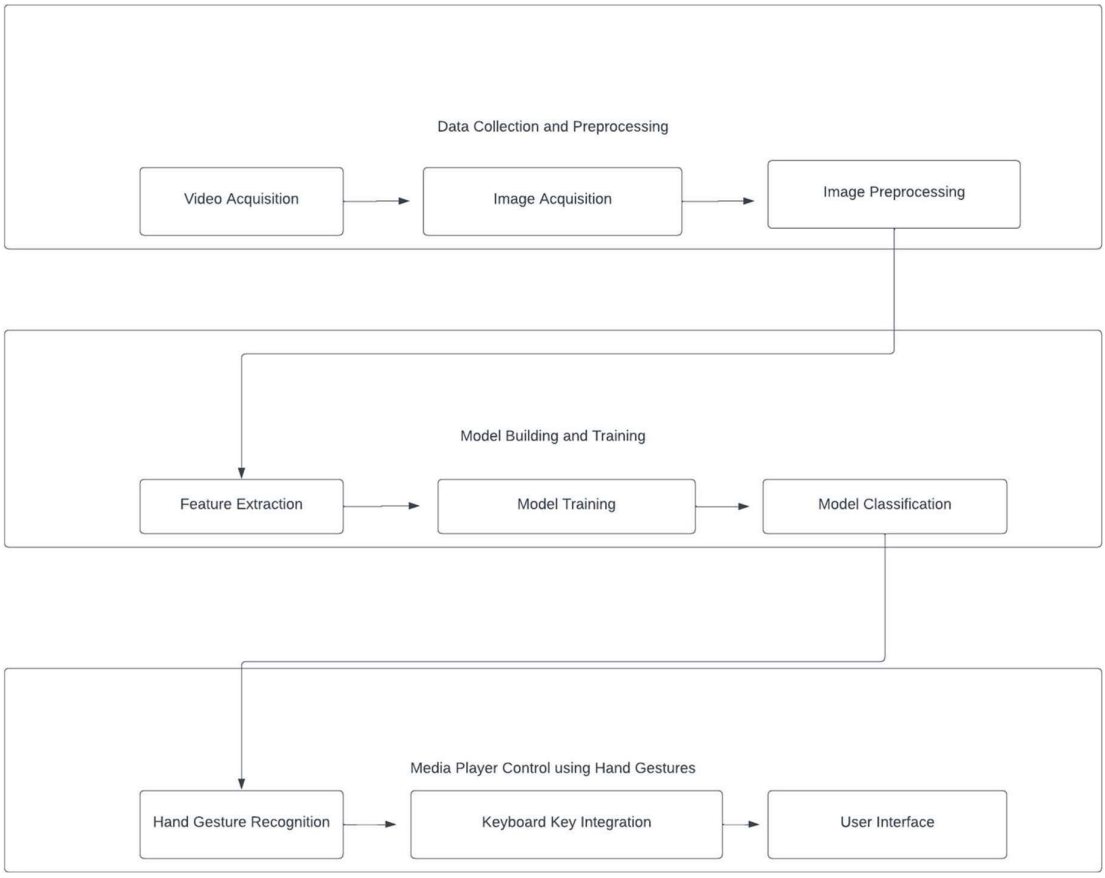

Fig 3.2 Proposed architecture diagram

Phase 1: Data Acquisition and Preprocessing: A webcam is used to record video footage, which is used as the image data input. There are more frame breaks in the video. OpenCV is used to transform these frames into black-and-white pictures, which are then saved in designated folders.

2) **Model Construction and Feature Extraction:** The Keras libraries are used to construct a CNN model. To extract only the necessary features from the images in the directories, pre-processing is done using the ImageDataGenerator class. Train dataset images are used to train the model.

3) **Classification and Prediction:** The model is assembled, and the test's accuracy is assessed. Keras libraries are used to classify gestures to a specific class and to save and load the model.

4) **Using the PyAutoGUI library**, integrate all gestures with control functions to integrate keyboard controls. The prediction is controlled and carried out. Create a web application and use streamlit sharing to deploy all project files.

3.3 Data Preparation

1. Four distinct people's gestures are gathered for this project. For every gesture, there are 200 images in the dataset. To store images, a directory structure is made. For user convenience, there are two modes available: train and test. The option to select any one of the modes is provided to the user. The model is trained using the images acquired in train mode, and its accuracy is tested using the images obtained in test mode. Depending on what the user enters, the images are saved in a particular directory.

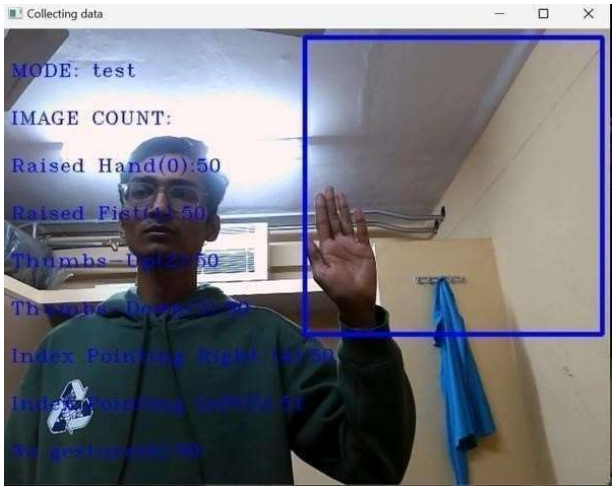

Fig 3.3 Data collection for training of model

2. When the camera is turned on, two frames are shown on the screen, and by using the read function and the mirror image, the user can record pictures frame by frame.

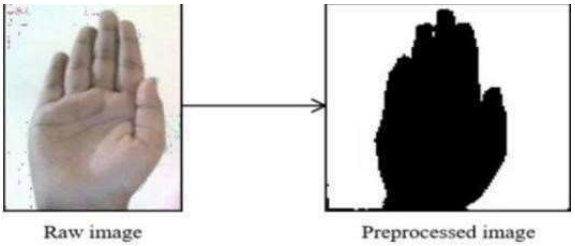

Fig 3.4 Data preprocessing

3. The user must make the gestures and place their hand inside the bounding box, or Region Of Interest (ROI). After being taken out of ROI, the frames are resized to 120x120x1. The screen displays the total number of images in every directory. Every time a user takes a picture by tapping one of the 0–6 number keys on the keyboard, the count is increased, and the pictures are stored in the appropriate class directory.[Fig 3.4]


Fig 3.5 Region of interest

4. During capturing, the preprocessed image was visible in the small frame, and these images will be kept in the dataset. After data collection, the user can log out by hitting the keyboard's escape key.

5. Data Augmentation is the technique of artificially creating new data from preexisting data, known as data augmentation, is mostly used to train new machine learning (ML) models. Large and diverse datasets are necessary for the initial training of machine learning models, however finding sufficiently diverse real-world datasets can be difficult due to data silos, legal restrictions, and other issues. Data augmentation involves making minor adjustments to the original data in order to artificially enlarge the dataset. Many sectors are increasingly using generative artificial intelligence (AI) technologies for quick and high-quality data augmentation

```
: from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=False)

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('data/train',
                                                 target_size=(120, 120),
                                                 batch_size=7,
                                                 color_mode='grayscale',
                                                 class_mode='categorical')

test_set = test_datagen.flow_from_directory('data/test',
                                            target_size=(120, 120),
                                            batch_size=7,
                                            color_mode='grayscale',
                                            class_mode='categorical')
```

```
Found 1004 images belonging to 7 classes.
Found 357 images belonging to 7 classes.
```

Fig-3.6 Data Augmentation

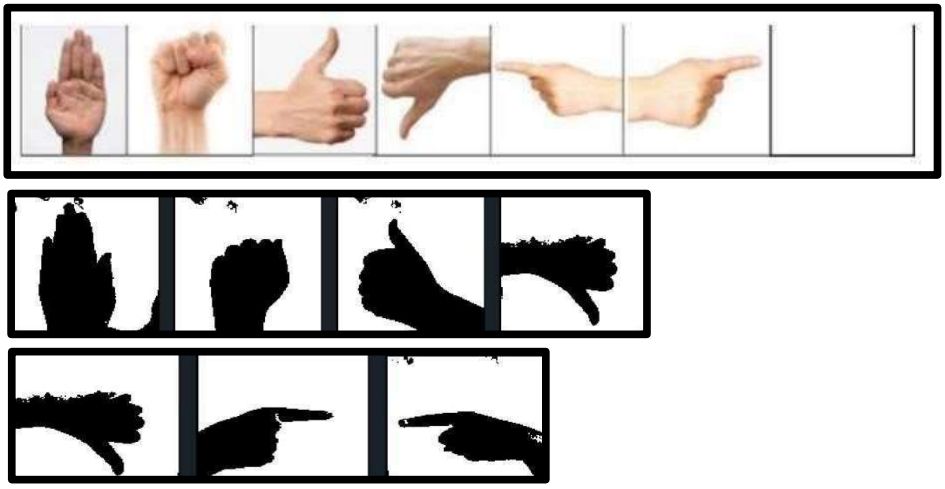

Fig 3.7 Gesture data after Pre-processing

3.4 Implementation (include code snippets, algorithms, tools and techniques, etc.)

The following is the step-by-step plan in which the Gesture Recognition Using Deep Learning is implemented:

3.4.1 SETTING UP PROJECT ENVIRONMENT

The implementation process begins with downloading and setting up essential software tools and libraries required for the project. The following steps outline the initial setup:

Python Installation:

Ensure Python is installed on your system. Otherwise install python from the Python website.

Setting Up Jupyter Lab: Use pip install Jupyter to install Jupyter Lab. Activate your virtual environment. Start using the Jupyter Lab command: Jupyter lab. Develop a project bygoing through Jupyter Lab via your web browser.

3.4.2 INSTALLING AND IMPORTING NECESSARY LIBRARIES

First, we need to install all the necessary libraries.

```
import os

# Install necessary libraries using pip
os.system("pip install pyautogui opencv-python numpy pillow matplotlib streamlit")
```

Fig:3.7 Installing necessary libraries

- **PyAutoGUI**: is a Python library, which enables carrying out automated mouse and key board activities. For example, it may serve as an automation tool for GUIs, as well as a snapping tool for displaying screenshots and simulating user interactions.
- **OpenCV (cv2)**: One of the open-source libraries currently available is OpenCV – Open- Source Computer Vision Library, which supports computer vision and machine learning. These include tools for image and video analysis like different image processing functions, object detection and feature extraction.
- **NumPy**: It contains a central package of scientific computing known as NumPy. It supports big multidimensional arrays and matrices in addition to all mathematical operations on array.
- **PIL (Python Imaging Library) / Pillow:** PIL (or Pillows in it more recent forms) is an open- source library that handles multiple types of image formats and operations on them including, but not limited too; opening, transforming, cropping, combining or resizing images, etc. However, it is a common choice for image processing.
- **Matplotlib:** Matplotlib – A 2D plotting library for Python. In Python it generates static, animated and interactive visualisations. Such a program may create different types of plots like plot, histogram, power spectrum among others, bar charts as well as error charts.
- **Streamlit**: is a Python library for creating web applications with minimal effort. It simplifies the process of turning data scripts into shareable web apps. It is often used for creating interactive and user-friendly dashboards.

Importing these libraries:

```
import cv2
import numpy as np
import os
from PIL import Image
from matplotlib import pyplot as plt
%matplotlib inline
```

Fig:3.8

3.4.3 Creating Directory Structure For Data Collection

```
if not os.path.exists("data"):
    os.makedirs("data/train")
    os.makedirs("data/train/01_palm")
    os.makedirs("data/train/02_fist")
    os.makedirs("data/train/03_thumbs-up")
    os.makedirs("data/train/04_thumbs-down")
    os.makedirs("data/train/05_index-right")
    os.makedirs("data/train/06_index-left")
    os.makedirs("data/train/07_no-gesture")
    os.makedirs("data/test")
    os.makedirs("data/test/01_palm")
    os.makedirs("data/test/02_fist")
    os.makedirs("data/test/03_thumbs-up")
    os.makedirs("data/test/04_thumbs-down")
    os.makedirs("data/test/05_index-right")
    os.makedirs("data/test/06_index-left")
    os.makedirs("data/test/07_no-gesture")
```

Fig:3.9 Directory structure for six gestures

OUTPUT:

Fig:3.10 Train test data split

Fig:3.11 Folders for six gestures

3.4.3 Capturing Images for Data Set

As mentioned in the data set description the data for the project is collected through the webcam to prepare the data set for TRAINING and TESTING. Here is the code snippet to capture images from the webcam.

```
cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    # Simulating mirror image
    frame = cv2.flip(frame, 1)

    # Getting count of existing images
    count = {'palm': len(os.listdir(directory+"/01_palm")),
             'fist': len(os.listdir(directory+"/02_fist")),
             'thumbs-up': len(os.listdir(directory+"/03_thumbs-up")),
             'thumbs-down': len(os.listdir(directory+"/04_thumbs-down")),
             'index-right': len(os.listdir(directory+"/05_index-right")),
             'index-left': len(os.listdir(directory+"/06_index-left")),
             'no-gesture': len(os.listdir(directory+"/07_no-gesture")),
            }
    # Printing the count in each set to the screen
    cv2.putText(frame, "MODE: "+mode, (10, 50),cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)
    cv2.putText(frame, "IMAGE COUNT:", (10, 100), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)
    cv2.putText(frame, "Raised Hand(0):"+str(count['palm']), (10, 150), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)
    cv2.putText(frame, "Raised Fist(1):"+str(count['fist']), (10, 200), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)
    cv2.putText(frame, "Thumbs-Up(2):"+str(count['thumbs-up']), (10, 250), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)
    cv2.putText(frame, "Thumbs-Down(3):"+str(count['thumbs-down']), (10, 300), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)
    cv2.putText(frame, "Index Pointing Right (4):"+str(count['index-right']), (10, 350), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,
    cv2.putText(frame, "Index Pointing Left(5):"+str(count['index-left']), (10, 400), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,
    cv2.putText(frame, "No gesture(6):"+str(count['no-gesture']), (10, 450), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,0,0), 1)

    x1 = int(0.5*frame.shape[1])
    y1 = 10
    x2 = frame.shape[1]-10
    y2 = int(0.5*frame.shape[1])
    # Drawing the ROI
```

Fig:3.12 Video Capturing code

3.4.4 Preprocessing The collected images

The preprocessing part of the provided code snippet involves extracting and processing the Region of Interest (ROI), which is a crucial step in preparing the captured hand gesture images for further analysis. Here's a breakdown of the relevant code:

```
    roi = cv2.resize(roi, (120, 120))

    cv2.imshow("Collecting data", frame)

    # do the processing after capturing the image!
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, roi = cv2.threshold(roi, 130, 255, cv2.THRESH_BINARY)
    cv2.imshow("ROI", roi)

    interrupt = cv2.waitKey(10)

    if interrupt & 0xFF == 27: # esc key
        break
    if interrupt & 0xFF == ord('0'):
        cv2.imwrite(directory+'01_palm/'+str(count['palm'])+'.jpg', roi)
    if interrupt & 0xFF == ord('1'):
        cv2.imwrite(directory+'02_fist/'+str(count['fist'])+'.jpg', roi)
    if interrupt & 0xFF == ord('2'):
        cv2.imwrite(directory+'03_thumbs-up/'+str(count['thumbs-up'])+'.jpg', roi)
    if interrupt & 0xFF == ord('3'):
        cv2.imwrite(directory+'04_thumbs-down/'+str(count['thumbs-down'])+'.jpg', roi)
    if interrupt & 0xFF == ord('4'):
        cv2.imwrite(directory+'05_index-right/'+str(count['index-right'])+'.jpg', roi)
    if interrupt & 0xFF == ord('5'):
        cv2.imwrite(directory+'06_index-left/'+str(count['index-left'])+'.jpg', roi)
    if interrupt & 0xFF == ord('6'):
        cv2.imwrite(directory+'07_no-gesture/'+str(count['no-gesture'])+'.jpg', roi)


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

Fig:3.13

This step converts the grayscale image into a binary image, where pixel values above a certain threshold become white (255), and those below become black (0). In this case, a threshold value of 130 is used. The thresholding helps in emphasizing the edges and important features of the hand gesture, making it easier for a machine learning model to learn the patterns.

cv2.imshow("ROI", roi): Displays the processed ROI in a separate window named "ROI." This visualization step can be useful for debugging and understanding how the preprocessing steps affect the image.

Normalization of Input Data: This step first converts the input data and labels into NumPy arrays, a standard format for handling numerical data in Python. Normalization is then performed by dividing the pixel values of the input data by 255.0. Since the original pixel values range from 0 to 255, this normalization scales the values to a range between 0 and 1. Normalizing the data helps in speeding up the training process and improving the model's performance by ensuring that each input feature contributes equally to the computation of gradients during the learning process.

One-Hot Encoding of Labels: This step converts the categorical labels into a binary matrix representation. The `two categorical` function from Keras is used for this purpose, were

num_classes=7 specifies that there are seven unique gestures in the dataset. One-hot encoding transforms each label into a vector with a length equal to the number of classes. In this vector, all elements are 0 except for the element corresponding to the class index, which is set to 1. This representation is essential for training a classification model, as it allows the model to output a probability distribution over the classes. The dtype parameter `dtype='i1'` ensures that the data type of the encoded labels is integer 1 byte, which is memory efficient.

3.4.4 Training CNN Model

Import of keras model and hidden layers for our convolutional network

```
In [17]: from keras.models import Sequential
         from keras.layers import Conv2D, MaxPooling2D
         from keras.layers import Dense, Flatten
```

Fig:3.14 Importing layers for CNN Model

Sequential: Sequential is a linear stack of layers in the neural network. It allows you to create models layer by layer in a step-by-step fashion.

Conv2D: This layer creates a convolutional layer for 2D spatial convolution. Convolutional layers are fundamental in CNNs for detecting patterns and features in the input data.

MaxPooling2D: Max pooling is a down sampling operation often used in CNNs. It reduces the spatial dimensions of the input volume, leading to a reduction in the number of parameters and computation in the network. It helps in retaining the most important information.

Dense: Dense, or fully connected layers, are used for classification in the neural network. They connect every neuron in one layer to every neuron in the next layer.

Flatten: This layer is used to flatten the input, which is typically the output of the convolutional and pooling layers. It transforms the data into a 1D array before passing it to the dense layers.

Building the CNN:

```
In [ ]: # Step 1 - Building the CNN

        # Initializing the CNN
        model = Sequential()

        # First convolution layer and pooling
        model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(120, 120, 1)))
        model.add(MaxPooling2D((2, 2)))

        # Second convolution layer and pooling
        model.add(Conv2D(64, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2, 2)))

        # Third convolution layer
        model.add(Conv2D(128, (3, 3), activation='relu'))

        # input_shape is going to be the pooled feature maps from the previous convolution layer
        model.add(MaxPooling2D((2, 2)))
        # Flattening the layers
        model.add(Flatten())

        # Adding a fully connected layer
        model.add(Dense(256, activation='relu'))
        model.add(Dense(7, activation='softmax'))

In [ ]: # Compiling the CNN
        model.compile(optimizer='adam', # Optimization routine, which tells the computer how to adjust the parameter values to minimize t
                      loss='categorical_crossentropy', # Loss function, which tells us how bad our predictions are.
                      metrics=['accuracy']) # List of metrics to be evaluated by the model during training and testing.
```

Fig:3.15 CNN Model

Model Initialization: Initialize a Sequential model using Keras.
Convolutional Layers: Add convolutional layers with ReLU activation and max-pooling to extract hierarchical features.

Flatten and Fully Connected Layer: Flatten the output from convolutional layers. Add a fully connected layer with ReLU activation.

Compile the Model: Compile the model using the Adam optimizer, categorical cross entropy loss function, and accuracy as the evaluation metric.

Preparing the train/test data and training the model:

```
In [ ]: from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=False)

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('data/train',
                                                 target_size=(120, 120),
                                                 batch_size=7,
                                                 color_mode='grayscale',
                                                 class_mode='categorical')

test_set = test_datagen.flow_from_directory('data/test',
                                            target_size=(120, 120),
                                            batch_size=7,
                                            color_mode='grayscale',
                                            class_mode='categorical')
```

```
In [ ]: history=model.fit(
        training_set,
        steps_per_epoch=125, # No of images in training set
        epochs=7,
        validation_data=test_set,
        validation_steps=50)# No of images in test set
```

Fig:3.16 preparing the train/test data and training the model

Data Generators: Create data generators for training and testing data using ImageDataGenerator with optional data augmentation.

Training: Use the fit method to train the model on the training set. Specify the number ofepochs, steps per epoch, validation data, and validation steps.

Saving entire model to a HDF5 file:

```
In [ ]: model.save('handrecognition_model.hdf5')
        model.summary()
```

Fig:3.17 Saving entire model

Saving the model weights:

```
In [ ]: model_json = model.to_json()
        with open("gesture-model.json", "w") as json_file:
            json_file.write(model_json)
        model.save_weights('gesture-model.h5')
```

Fig:3.18 Saving JSON file

3.4.5 Hand Gesture Prediction and Media player Control

Importing Libraries:

```
In [ ]: import numpy as np
        from keras.models import model_from_json
        import operator
        import cv2
        import sys, os
        import pyautogui
        import time
```

Fig:3.19 Importing libraries

Loading the model:

```
In [ ]: json_file = open("gesture-model.json", "r")
        model_json = json_file.read()
        json_file.close()
        loaded_model = model_from_json(model_json)
        # load weights into new model
        loaded_model.load_weights("gesture-model.h5")
        print("Loaded model from disk")

In [ ]: final_label = ""
        action=""
```

Fig:3.20 Loading the model

Opening JSON File:

json_file = open("gesture-model.json", "r"): Opens the JSON file containing the architecture (structure) of the neural network model in read-only mode.

Reading JSON Content:

model_json = json_file. read (): Reads the content of the opened JSON file and stores it in thevariable model_json.

Closing JSON File:

json_file.close(): Closes the opened JSON file since its content has been read.

Load Model Architecture:

loaded_model = model_from_json(model_json): Uses a function from Keras (model_from_json) to create a new neural network model based on the architecture described in the loaded JSON content. This model is initially untrained.

Loading Model Weights: loaded_model.load_weights("gesture-model.h5"): Loads the pre-trained weights of the neural network from a separate file (gesture-model.h5). These weights are the learned parameters that allow the model to make accurate predictions.

Printing Confirmation: print("Loaded model from disk"): Prints a message to indicate that the model has been successfully loaded.

Initialization of Labels:

final_label = "": Initializes an empty string variable final_label that will be used to store the final recognized gesture label.

action = "": Initializes an empty string variable action that will be used to store the corresponding action based on the recognized gesture.

Media Controlling Through Gesture:

The real-time working of the system when different gestures are shown in front of the webcam is displayed from Fig 3.21 to Fig 3.27 The images show how the media player is controlled using hand gestures. When the fist is shown the first time the volume is muted and when it is predicted again thevolume is unmuted. Similarly, the media player is controlled with the other six gestures.

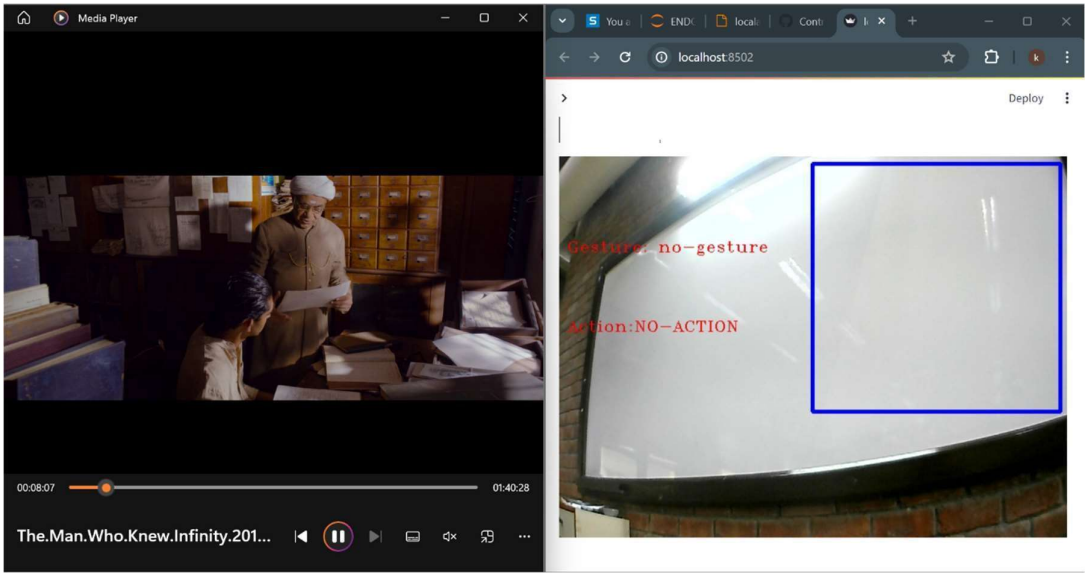

FIG 3.21 Media player continues in its state when no gesture is detected

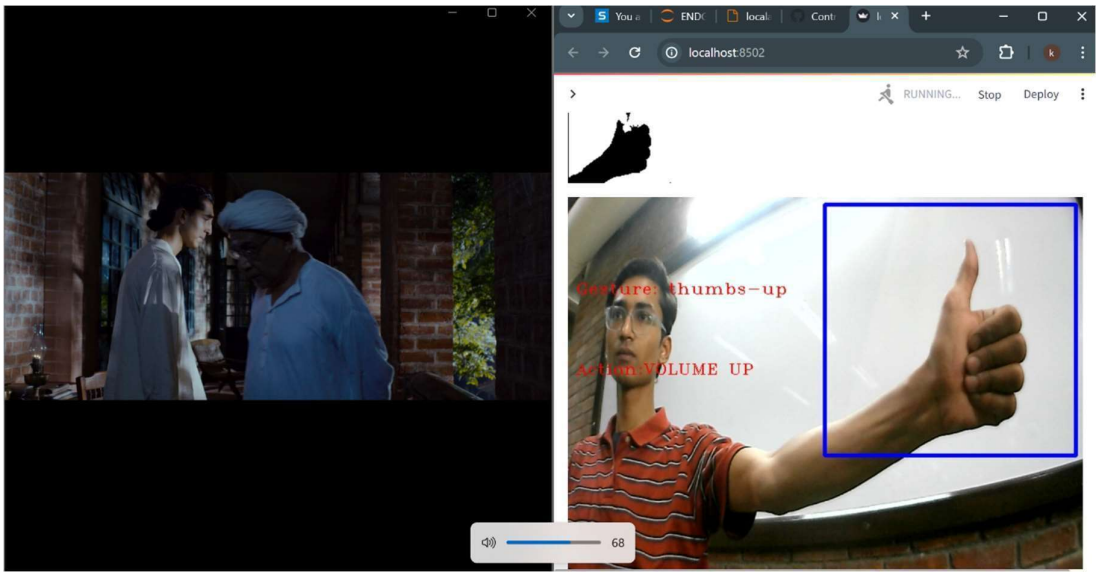

FIG 3.22 Volume is increased when thumbs up is detected

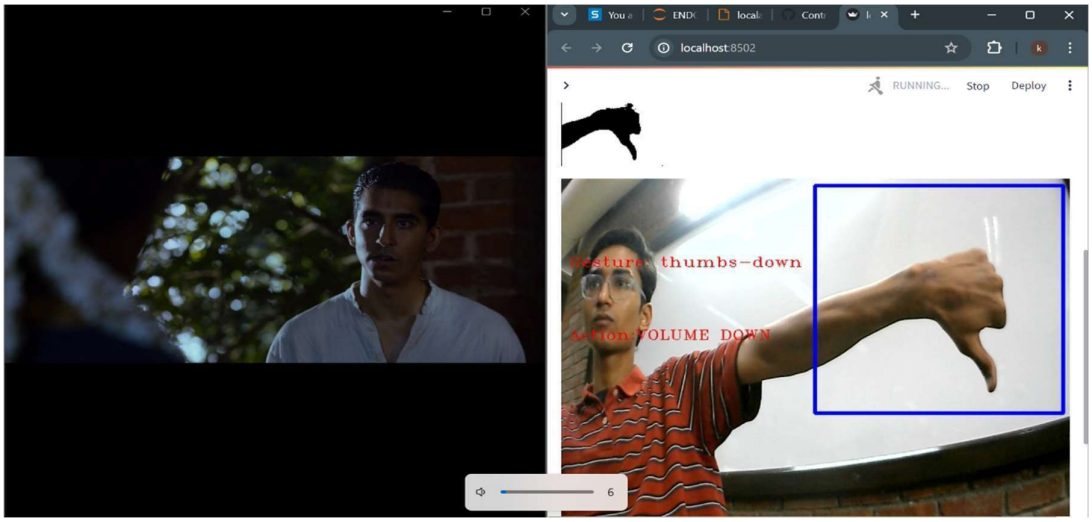

FIG 3.23. Volume is decreased when thumbs down is detected

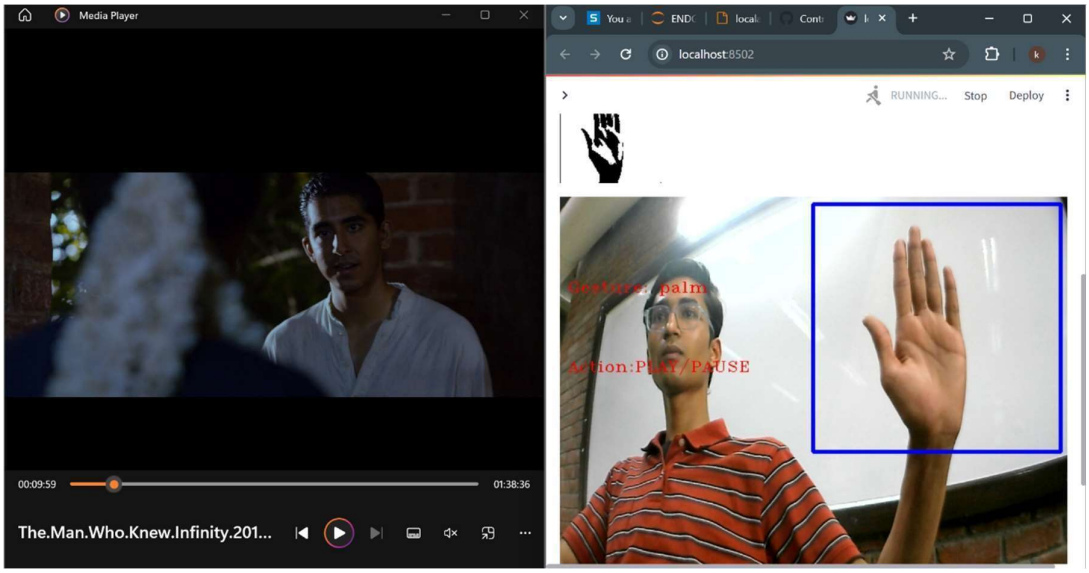

FIG 3.24 Video is paused while playing when the palm is detected

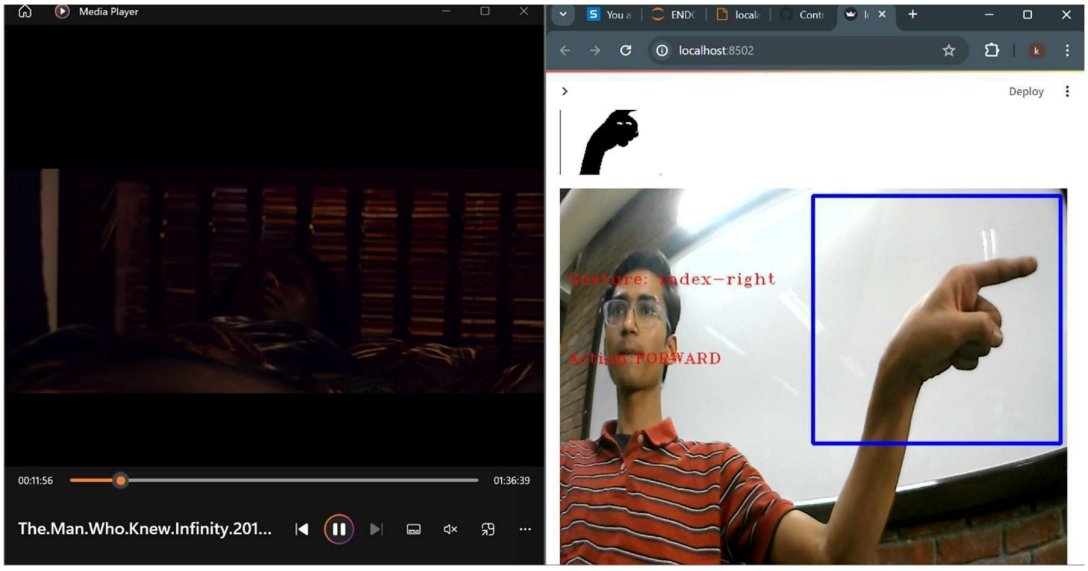

FIG 3.25 Video is forwarded when index right is detected

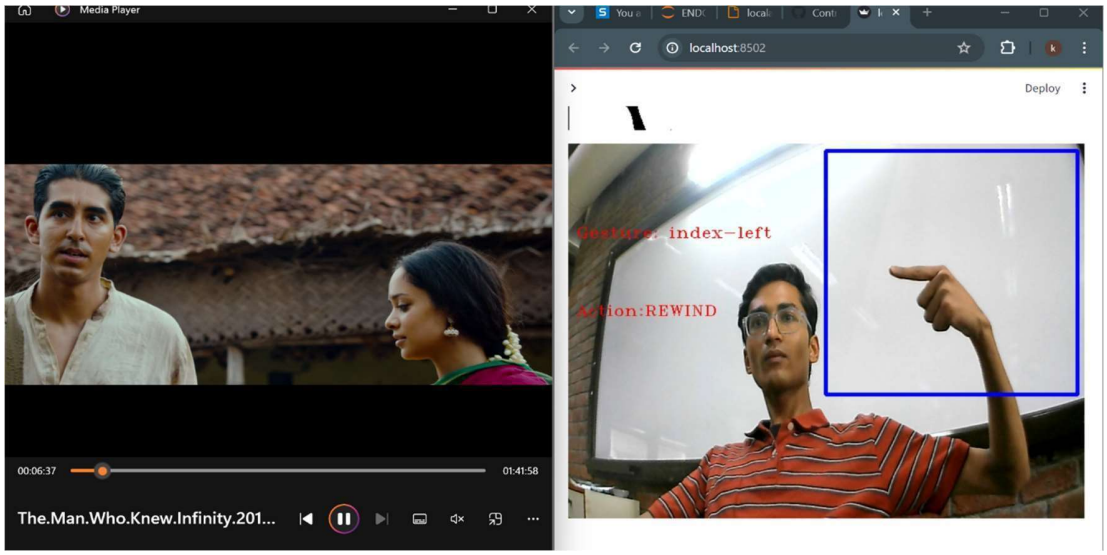

FIG 3.26. Video is rewound when index left is detected

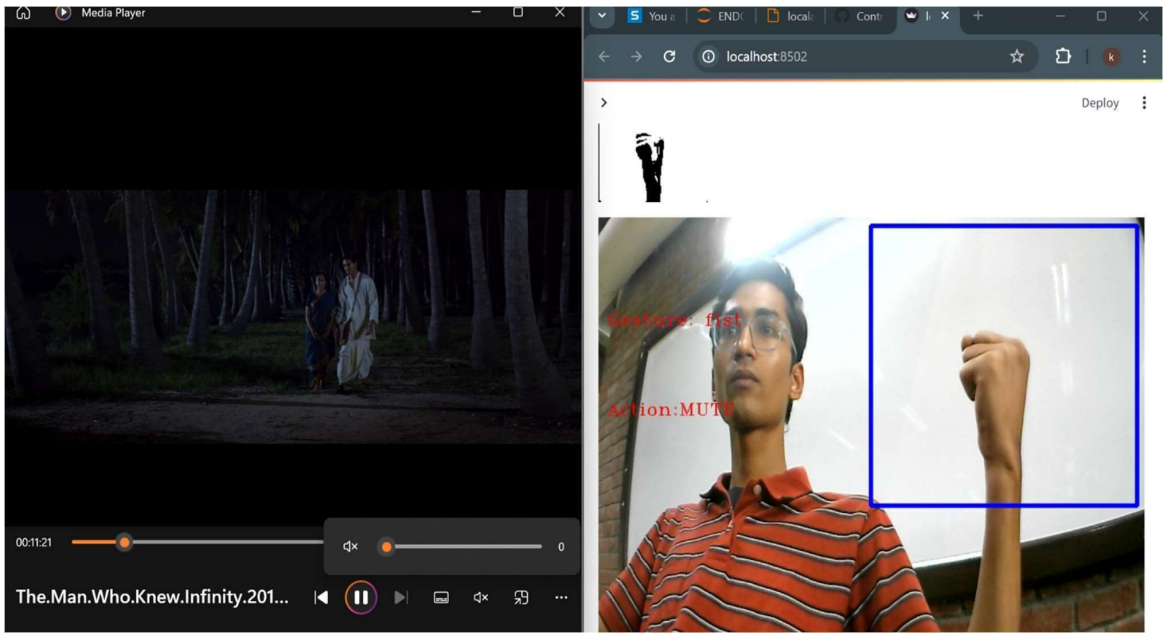

FIG 3.27 Media player is muted when the fist is detected.

The project sources files are hosted in a web app using streamlit.io sharing which enables the system to be used from their respective devices. It also provides a link to download the source files. Fig.3.28 to Fig. 3.29 shows different pages of the web app deployed on streamlit.

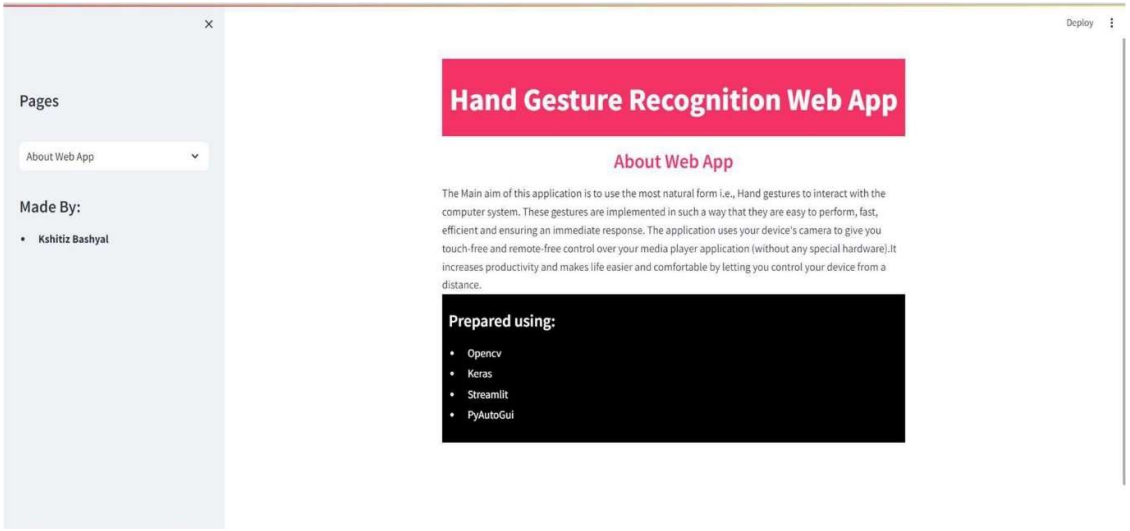

FIG 3.28 Homepage of the webapp

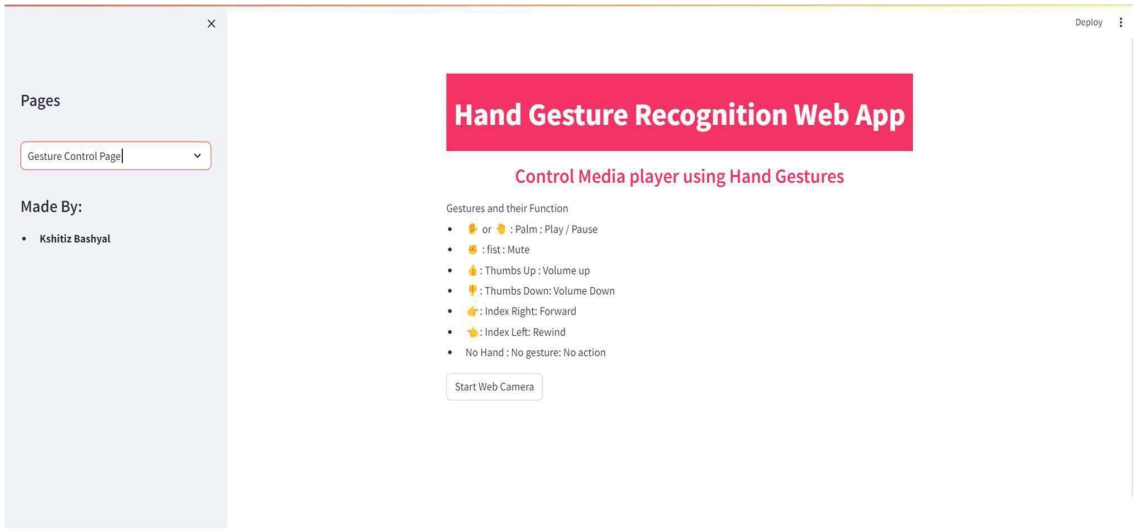

Fig 3.29 Gesture recognition page

3.5 Key Challenges

1. Camera-based gesture recognition poses certain difficulties, including lighting, background noise, and region of interest. Devices such as the Microsoft Kinect can help solve these problems partially.

2. One enduring issue in gesture recognition is slow and costly computation.

3. Sensor-equipped glove-based systems are costly and need constant glove wear from users.

4. Another challenge in gesture recognition is handling noise in the region of interest and image background. Through the use of morphological operations and color segmentation, neural networks can assist in the solution of these issues.

5. The field of data augmentation holds potential to enhance the effectiveness of gesture recognition, but it has not yet been thoroughly investigated.

Chapter 4 Testing

4.1 Testing Strategy

In order to evaluate the efficiency and portability of a Gesture Detection System in various cases of use, it is necessary to carry out robust testing procedures. Here's an elaborate explanation of the mentioned experiments:

Testing with Different Sets of People:

This comprised gathering different individuals to point using their hands towards the camera as they faced the machine. However, this variation in users' hand sizes, shape, gestures, makes it possible whereby the system will be able to recognize gestures under most conditions.

Testing with Different CNN Models:

Comparison study was done between the new proposed CNN model and other two other similar CNN models. The testing involved running hand gestures of different persons through these models and evaluating their performance in recognizing and categorizing the gestures correctly.

Real-time Prediction for Media Player Control:

To prove its predictive capacity in real time, it allowed users to operate with hand gestures of a media player. The real time gestural processing was triggered by a click onto the "Start Web Camera" button and served to illustrate the system's receptivity to user instigation in the course of live interaction.

High Accuracy Achieved by Proposed CNN Model:

The experiments showed that the proposed CNN model proved to be reliable for distinguishing the hand gestures of different people. The model's strong structure and learning with diverse sets of data led to its high quality of generalizability between different people that makes it a valuable option for real-life applications.

Evaluation Using Performance Metrics:

The evaluation of the CNN model performance also involved the use of precision, recall, and F1-score. Precision is used to determine the correctness of positive predictions, recall analyzes the ability of capturing all occurrences with positive predictions, and F1-score evaluates these two measurements to come up with an estimate that combines precision and recall. These metrics provide comprehensively for strength or weakness of model.

4.2 Tools Used

Neural Network Model:

- The implementation involves a CNN architecture with multiple layers. Specific details of the CNN model architecture are provided in the experimental results.

Web Application Deployment:

- Streamlit.io: The web application is deployed using Streamlit sharing for accessibility. Data Collection:
- The system involves a custom dataset with images of hand gestures. The data collection process includes capturing images using the webcam.

Chapter-5 Results and Evaluation

5.1 Comparison with existing solutions

Seven gestures are chosen based on how practical they are for the applications and many tests are carried out to evaluate the resilience of the model in order to test the proposed system in a real-world setting. For a total of 1400 photos, the custom dataset included 200 images each class from seven separate classes. As seen in Fig. -, the photos are taken in optimal lighting conditions with a static background and no background noise. Good results are obtained when testing the suggested method in real- time with various groups of people in appropriate lighting and noise-free environments. As indicated in Table 5.1, three distinct CNN models are used to test the system using hand gestures from various users. By changing the quantity of pooling and convolutional layers

Depth	CNN Model 1	CNN Model 2	CNN Model 3
1	Convolutional (3x3)	Convolutional (5x5)	Convolutional (5x5)
2	Convolutional (3x3)	Max Pooling (2x2)	Max Pooling (2x2)
3	Max Pooling (2x2)	Convolutional (7x7)	Convolutional (7x7)
4	Convolutional (3x3)	Max Pooling (2x2)	Max Pooling (2x2)
5	Max Pooling (2x2)		Convolutional (5x5)
6			Max Pooling (9x9)

Table 5.1 CNN Models Architecture

The training time and testing time obtained from each CNN model are shown in Table 5.2

Time taken	Training (in milliseconds/step)	Testing (in seconds)
CNN Model 1	460	0.136736869812
CNN Model 2	694	0.0553321838378
CNN Model 3	828	0.1521420478820

Table 5.2 Training & Testing Time

The performance metrics of the 3 CNN models are compared and the system showed a highest of 98% accuracy for model 3 as shown in Table IV.

	CNN Model 1	CNN Model 2	CNN Model 3
Accuracy	0.9492	0.9746	0.9778
Precision	0.95	0.97	0.98
Recall	0.95	0.97	0.98
F1 Score	0.95	0.97	0.98

Table 5.3 Comparison of accuracy of proposed model

The accuracy of the proposed model is compared with different models as shown in Table 5.3.

	precision	recall	f1-score	support
01_palm	0.96	1.00	0.98	22
02_fist	1.00	0.91	0.95	23
03_thumbs-up	0.96	1.00	0.98	23
04_thumbs-down	1.00	1.00	1.00	24
05_index-right	1.00	1.00	1.00	26
06_index-left	1.00	1.00	1.00	18
07_no-gesture	1.00	1.00	1.00	32
accuracy			0.99	168
macro avg	0.99	0.99	0.99	168
weighted avg	0.99	0.99	0.99	168

Table 5.4 Precision, recall and F1-score of different signs by our CNN model 3

5.2 Results

The user must choose the kernel and click "run all" to initiate real-time prediction to control the media player. The OpenCV frames will then be displayed to begin the prediction, the system operates in real time when various motions are made in front of the webcam. The pictures demonstrate how to use the media player with hand gestures. The loudness is muted when the fist is displayed for the first time; it is unmuted when it is projected to appear again. In a similar manner, the other six motions are used to manage the media player. Other media players have also responded to it in a similar manner.

For the hand gesture recognition system, the suggested CNN model achieved a high level of accuracy. Figure. compares the accuracy of the third CNN model's train and test sets, whereas Figure. compares the loss of the third CNN model's train and test sets. The graphs' convergence indicates that there is no overfitting or underfitting problems with the model, making it suitable for prediction.

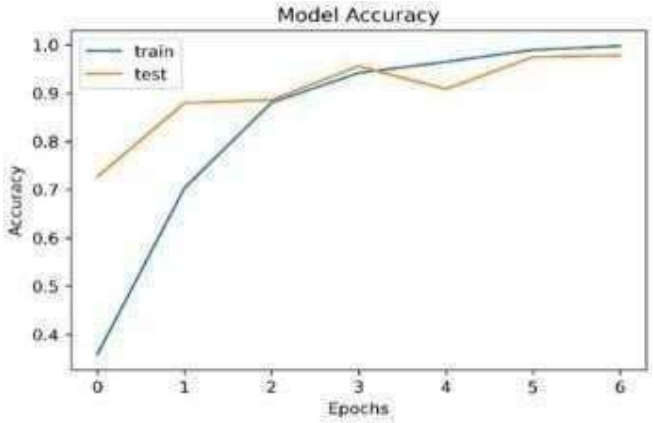

Fig 5.5 Graph of variations in accuracy with Epoch in CNN model.

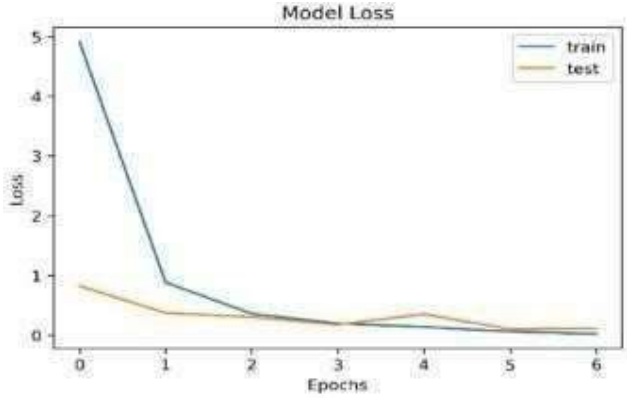

Fig 5.6 Graph of variations in loss with Epoch in CNN model

Additionally, the CNN model's performance is assessed using various performance measures, including recall, precision, and f1-score, and the results are presented through the classification report. The model 3 categorization report is displayed in Figure A fully linked SoftMax layer serves as the last layer in the implementation of the CNN model. With a 97.7% testing accuracy and a just 2% misclassification on the customized dataset, Figure showcases the optimal CNN model.

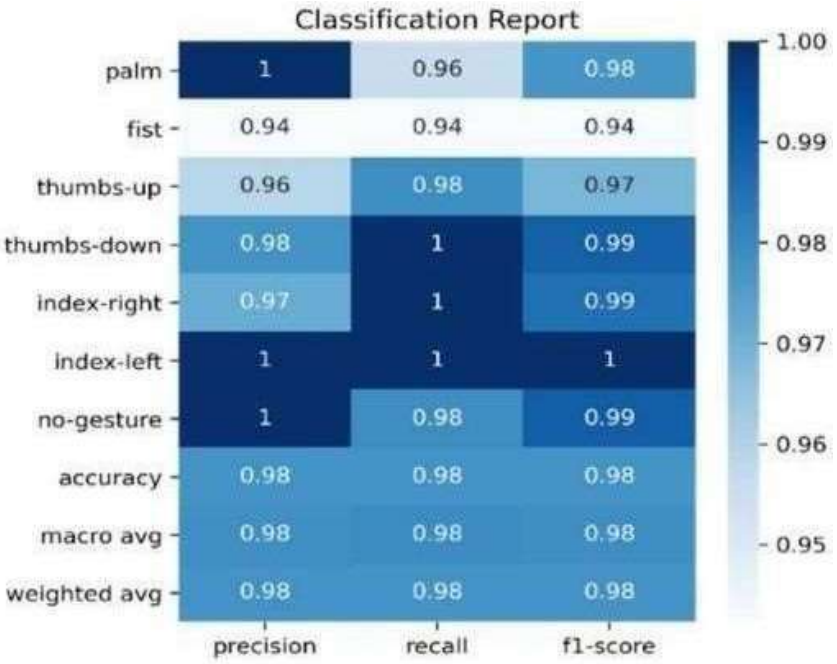

Fig 5.7 Classification report of CNN model

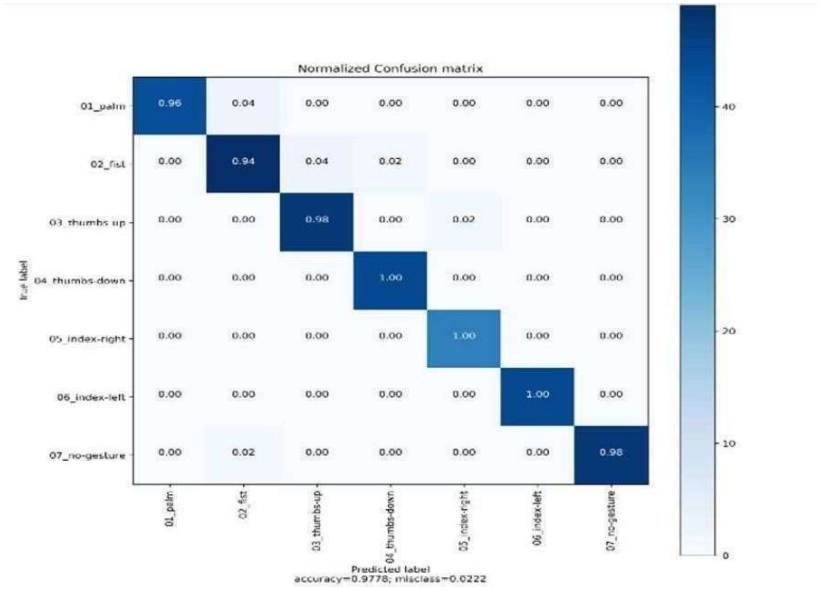

Fig 5.8 Confusion matrix of the CNN model.

Chapter-6 Conclusions and Future Scope

6.1 Conclusion

1. Based on computer vision and deep learning techniques, the suggested method of interacting with media players through hand gestures via an online application has demonstrated encouraging outcomes in terms of effectiveness and user-friendliness.

2. It has been discovered that Convolutional Neural Networks (CNN) perform better for hand gesture recognition than earlier algorithms, offering superior evaluation metrics.

3. Neural network-based gesture recognition has been investigated through a variety of approaches, such as image processing methods like color segmentation, contour extraction, and morphological filters.

4. CNN models with fully connected SoftMax layers have been used, and the results show that they can recognize gestures with up to 96.83% accuracy.

5. Although issues with background noise, illumination, and processing speed persist, technological developments and methods such as data augmentation can help get around these problems and enhance the effectiveness of gesture recognition even more.

6.2 Future Scope

1. To make gesture recognition systems more accurate and resilient in difficult situations, like cluttered areas and situations with people in the background, more research can be done.

2. Investigating data augmentation methods can improve gesture recognition performance by expanding the variety and volume of training data.

3. Other sensors, like infrared cameras or depth sensors, can be integrated to provide more data and increase the precision of gesture recognition systems.

4. To make gesture-controlled systems more responsive and user-friendly, real-time gesture recognition algorithms that can handle dynamic gestures and continuous movements should be investigated.

5. Increasing the number of gestures that are recognized and creating customizable gesture mapping options can provide users greater freedom and control over how they interact with media players and related apps.

6. Investigating gesture recognition's potential in other fields, like robotics, augmented reality, and virtual reality, may create new opportunities for natural and engaging human-computer interaction.

References

[1] Li, S., Chan, A. B., & Chung, A. C. S. (2018). "Hand Gesture Recognition Using Convolutional Neural Networks for Real-Time Applications." IEEE Transactions on Multimedia, vol. 20, no. 8, pp. 2081-2090, August 2018.

[2] Wang, J., Chen, Y., Hao, S., & Pan, W. (2016). "Deep Learning for Hand Gesture Recognition on Skeletal Data." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 189-196, June 2016.

[3] Cao, Q., Pan, J., & Ngo, C. W. (2017). "Hybrid CNN and Dictionary-Based Models for Hand Gesture Recognition." IEEE Transactions on Image Processing, vol. 26, no. 5, pp. 2265-2278, May 2017.

[4] Wu, Y., Lin, Z., Wang, X., & Gao, W. (2018). "MV-CNNs for Real-Time Hand Gesture Recognition." IEEE Transactions on Cybernetics, vol. 48, no. 4, pp. 1166-1178, April 2018.

[5] Li, X., Yang, Y., Zhou, F., & Li, Y. (2019). "Real-Time Hand Gesture Recognition Using Depth and RGB Information." IEEE Access, vol. 7, pp. 135631-135641, September 2019.

[6] Yun, K., Honorio, J., Chattopadhyay, D., Berg, T. L., & Samaras, D. (2012). "Two-Handed Gesture Recognition in Ego-Centric View." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1919-1926, June 2012.

[7] Zhao L. Gesture Control Technology: An investigation on the potential use in Higher Education. University of Birmingham, IT Innovation Centre: Birmingham, UK. 2016 Mar.

[8] Harshada Naroliya, Tanvi Desai, Shreeya Acharya, Varsha Sakpal Enhanced Look Based Media Player with Hand Gesture Recognition. International Research Journal of Engineering and Technology, (ISSN: 2395-0072 (P), 2395- 0056 (O)).2018;5(3): 2032-35.

[9] Wachs JP, Kölsch M, Stern H, Edan Y. Vision-based handgesture applications.

[10] Yashas J, Shivakumar G. Hand Gesture Recognition: A Survey. In2019 International Conference on Applied Machine Learning (ICAML) 2019 May 25 (pp. 3- 8). IEEE.

[11] Sharma P, Sharma N. Gesture Recognition System. In2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU) 2019 Apr 18 (pp. 1-3). IEEE.

[12] Hakim NL, Shih TK, Kasthuri Arachchi SP, Aditya W, Chen YC, Lin CY. Dynamic hand gesture recognition using 3DCNN and LSTM with FSM context-aware model. Sensors. 2019 Jan;19(24):5429.

[13] Cardenas EJ, Chavez GC. Multimodal hand gesture recognition combining temporal and pose information based on CNN descriptors and histogram of cumulative magnitudes. Journal of Visual Communication and Image Representation. 2020 Aug 1; 71:102772.

[14] Adithya V, Rajesh R. A deep convolutional neural network approach for static hand gesture recognition. Procedia Computer Science. 2020 Jan 1;171:2353-61. [15] Oudah M, Al-Naji A, Chahl J. Hand gesture recognition based on computer vision: a review of techniques. journal of Imaging. 2020 Aug;6(8):73.

[15] Mohammed AA, Lv J, Islam MD. A deep learning-based Endto-End composite system for hand detection and gesture recognition. Sensors. 2019 Jan;19(23):5282

[16] Pinto RF, Borges CD, Almeida A, Paula IC. Static hand gesture recognition based on convolutional neural networks. Journal of Electrical and Computer Engineering. 2019 Oct 10;2019

[17] Cao Z, Hidalgo G, Simon T, Wei SE, Sheikh Y. OpenPose: realtime multi- person 2D pose estimation using Part Affinity Fields. IEEE transactions on pattern analysis and machine intelligence. 2019 Jul 17;43(1):172-86.

[18] Zhuang H, Yang M, Cui Z, Zheng Q. A method for static hand gesture recognition based on non-negative matrix factorization and compressive sensing. IAENG International Journal of Computer Science. 2017 Mar 1;44(1):52-9.

[19] Paul PK, Kumar A, Ghosh M. Human-Computer Interaction and its Types: A Types. International Conference on Advancements in Computer Applications and Software Engineering (CASE 2012), At Chittorgarh, India. –2012 2012 Dec 21.

[20] Ritupriya G.Andurkar, Human–computer interaction. International Research Journal of Engineering and Technology, (ISSN: 2395-0072 (P), 2395-0056 (O)). 2015; 2(6):744-72.

[21] Shanthakumar VA, Peng C, Hansberger J, Cao L, Meacham S, Blakely V. Design and evaluation of a hand gesture recognition approach for real-time interactions. Multimedia Tools and Applications. 2020 Feb 21:1-24.

[22] Bashir A, Malik F, Haider F, Ehatisham-ul-Haq M, Raheel A, Arsalan A. A smart sensor-based gesture recognition system for media player control. In2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) 2020 Jan 29 (pp. 1-6). IEEE