ISSN: 2634 - 8853

Journal of Engineering and Applied Sciences Technology

SCIENTIFIC Research and Community

Review Article

Open Access

Design and Development of Body Language Analysis System Based on Motion and Gesture Recognition

Md Shahriar Kabir Sajib[1]* and Khaled Mahmed Shawon[2]

[1]M.Sc. in Software Engineering, School of Software Engineering, University of Science and Technology of China, Hefei, Anhui, China

[2]Bachelor in Software Engineer, School of Computer Science and Artificial Intelligence, Zhengzhou University, Henan, China

ABSTRACT

This study presents the development of the Body Gesture Decoder System (BGDS), a real-time body gesture recognition system designed to improve human-computer interaction (HCI) by enabling natural, gesture-based communication. The purpose of this research is to create an accessible, lightweight system that recognizes human gestures through standard webcams, making it applicable in a variety of real-time, low-cost environments such as remote education, telemedicine, and security. The BGDS utilizes the MediaPipe Holistic framework for pose estimation, which extracts high-precision landmarks from body, face, and hand movements. A Random Forest classifier is trained on these landmarks to classify predefined gestures like- Happy, Sad, Thumbs Up, Victory. The system architecture is designed for modularity and real-time performance, achieving an accuracy of over 85% across various test conditions. It is implemented as a web-based application, making it scalable and easy to integrate with different platforms. The significance of this study lies in its demonstration of a practical, real-time solution for gesture recognition using only standard hardware, eliminating the need for expensive sensors or specialized equipment. The findings indicate that the BGDS can effectively recognize and classify a range of gestures with minimal latency, even in challenging lighting conditions. The implications of this research suggest broad potential applications in diverse sectors, including online education, telemedicine, assistive technologies, and interactive gaming. Furthermore, ethical considerations such as user privacy and data protection are central to the system's design, ensuring that no raw video or biometric data is stored. This work paves the way for future developments in the field of body language analysis, with opportunities for expanding gesture recognition capabilities and improving system accuracy and scalability.

***Corresponding author**

MD Shahriar Kabir Sajib, M.Sc. in Software Engineering, School of Software Engineering, University of Science and Technology of China, Hefei, Anhui, China.

Received: January 08, 2026; **Accepted:** January 12, 2026; **Published:** January 21, 2026

Keywords: Gesture Recognition, Real-Time Detection, MediaPipe Holistic, Random Forest, Human-Computer Interaction, Privacy, Body Language Analysis

Introduction

Over the last several decades, human-computer interaction (HCI) has changed immensely, moving from simple command-based systems to more and more complex multi-modal gestural interfaces which try to recreate how humans generally interact with one another [1,2]. Advances in speech recognition, natural language, and touch interfaces are the engines for these changes. The vast majority of HCI systems still depend heavily on explicit, manual input devices - like typing, clicking, or pressing buttons - which do not necessarily grasp the rich, non-verbal nature of human communication [3-5]. In most of today's digital systems, human gestures, body language, and facial expressions, which are important aspects of non-verbal communication, are mostly ignored. So many advantages could be gained from our human ability to communicate using non-verbal cues most appropriately when developing systems that allow for more natural human-like interaction between the user and the computer, where body language can be shown to communicate more nuanced or abstract concepts than spoken communication [6-8].

Gesture recognition ranks among the highly promising means of infusing body language into human-computer interaction (HCI). Gesture recognition is defined as the interpretation of human motions, usually through visual information, to the end of interaction with a computer [9-11]. Gesture recognition aims then at the accurate recognition and classification of a variety of bodily gestures, including hand movements, postures, or facial expressions. Gesture recognition holds huge promise in areas where a natural and intuitive mode of communication can enhance user experience, such as virtual reality, healthcare, distance education, assistive technology, and security systems. Several issues, however, impede the design of reliant real-time gesture detection systems [12,13]. A huge concern is the need for an accurate and effective gesture recognition system that can operate in a variety of environmental settings without relying on expensive or specialized hardware. The current systems are therefore not widely applicable in real-world scenarios as they typically involve expensive depth sensors, 3D cameras, or complicated wearables. Variations in user attributes, such as body sizes and hand shapes, and gesture-drawing techniques may also present challenges to gesture detections supervised by conventional 2D cameras. Another major limitation of existing gesture recognition systems is that they seldom allow real-life applications, with users in various locations and doing gestures at different speeds and under

a variety of lighting conditions, because they tend to focus on the recognition of rather simple types of motions or necessitate a highly controlled environment [14,15].

There is a lack of research that has established a flexible, scalable, inexpensive system to detect gestures in real time using only commercially available consumer hardware, such as browser-based learning algorithms like webcams. All existing systems are either too complicated to use for qualify for the casual users or typically consider only a limited number of promotions of gestures overtime without any consideration to the context in which the gesture is displayed-reducing the relevance and utility of the technology overall. The limitations this imposes on the use of gesture-based technologies in real world applications severely limits usability. This work attempts to address this gap by presenting the Body Gesture Decoder System (BGDS), a real-time gesture detection system designed for simple, gesture-based human computer interaction (HCI) that makes available consumer hardware like webcams useful by utilizing low power consume processes like browsers' machine learning algorithms. BGDS uses a powerful posture detection software to capture high fidelity landmark data of the drivetrain (facial/hand/body) by appending the MediaPipe Holistic framework into a stitched function that streamlines BGDS information capture. BGDS uses significantly less processing requirements to☆®n only from landmark data instead of framerate specific raw image data from stream-based so BGDS achieves good accuracy at minimal computing complexity. Once landmark data is captured consistent with the event to be detected, BGDS collects the GPS information is stored for input to a Random Forest classifier, which it trained to detect pre-defined gestures such as "Happy", "Sad", "Thumbs Influencers", "Thumbs Up", and finally detecting and displaying users use of "Victory" and others, a simple but very powerful way of real-time gestures.

The BGDS system is suitable for many different applications due to its scalability and flexibility. The cost of hardware is minimized because BGDS uses already commercially available devices, like cell phones and desktop computers, and does not require special technology to enable real-time gesture detection. As opposed to other strategies that require expensive or specialized hardware [16], this strategy will enable more practical, affordable, and scalable solutions across many industries, such as interactive entertainment, telemedicine, assistive technologies, and distance education. BGDS is modular, which makes it possible to add additional gestures, support multiple users, and, in theory, improve the performance of gesture classification accuracy.

This study has made two key contributions. The first is that we have demonstrated how accurate and efficient real time gesture recognition can be achieved with simple machine learning algorithms and common webcams. The second contribution is that we have expanded HCI to include non-verbal communication as we present a workable method of using gesture-based interaction in countless real-world applications. BGDS is not only technologically capable, we also want to make sure we are putting ethical and privacy considerations in the forefront of the design. This system only keeps the landmark coordinates needed for gesture classification and not the raw video or biometric data, thus following the guidelines established for data protection policies. The design, implementation and evaluation of BGDS is presented in this paper and we will also discuss technical details about the system and suggest potential and practical uses. We will highlight the various issues we experienced as we processed through the development, discuss how the system performed in various environmental circumstances and suggest avenues for improvement and efficiency for accurate gesture detection use in real time applications. BGDS is also a significant leap forward in body language analysis and human-computer interaction as it remedied the issues embroiled in the development of the gesture detection system and provides a low-cost solution for the production of a scalable gesture system.

Literature Review

Gesture Recognition Systems

The area of gesture recognition has undergone major transformations in recent times, especially when deep learning has been adapted along with the development of potent computing tools. Earlier systems mainly relied on hardware-dependent techniques like depth cameras, infrared sensors, and accelerometers, which could achieve high accuracy but were beyond the price range of an average user. Now, standard 2D cameras could be used for recognizing gestures as technology came up with computer vision-based techniques while computing power increased; however, under different lighting and background interference, these systems suffered inaccuracy [17-19]. Recent trends have gone towards more easy-to-use, cheaper, and scalable solutions that rely on software and algorithms based on computer vision and machine learning. Increasing accuracy through deep learning methods, especially convolutional neural networks (CNNs), has been the major trend in gesture identification in the past five years. Such techniques, though promising in achieving high performance in controlled environments, are seldom inexpensive in computational resources, such as the best GPUs, and cannot easily process in real-time [20,21]. Moreover, a wide variety of lighting conditions, occlusions, and inter-user variations are just a few of the challenges faced by deep learning-based methods in becoming widely applicable. Because of these constraints, researchers have been looking for new and better techniques that do not compromise speed and efficacy while running on more mainstream hardware [22-24].

Pose Estimation Techniques

It is critical now for gesture-recognition systems to include pose estimation, which helps in detecting significant points on the human body, face, and hands, so that spatial references can be given during the interpretation process. In recent years, pose estimation has been revolutionized with the implementation of frameworks like OpenPose and MediaPipe, which provides reliable real-time accurate solutions. OpenPose was one of the first frameworks to develop highly accurate multi-person posture estimation but has a drawback since it is computationally intensive for deployments on consumer-grade devices [25,26]. In contrast, Google's MediaPipe has gained rapid traction in the last five years since it can offer real-time performance on low-end devices. MediaPipe Holistic is the industry standard for real-time pose estimation, where more than 500 hand, facial, and body landmarks can be estimated. It has changed the game in mobile devices, and web-based systems with a low-cost hardware setup, as it does not need depth sensors or GPUs [27]. Recent contributions toward posture estimation have been primarily aimed at optimizing these frameworks for scalability and minimizing the computational overhead. This is extremely important for applications that need real-time processing, such as your Body Gesture Decoder System (BGDS). MediaPipe's impressive posture estimation pipeline can thereby be adapted to improve performance in complicated or mobile gesture recognition contexts. BGDS offers the same gesture classification across diverse real-time applications, but without requiring expensive hardware or specialized sensors, by using the extensive body, face, and hand landmarks offered by MediaPipe [28].

Machine Learning for Gesture Classification

Machine learning is largely a prerequisite for gesture recognition systems, which are required to categorize gestures using spatial knowledge about posture estimation [29,30]. Most early approaches had simple classifiers like Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN), but as more diverse and large datasets became available, these techniques were frequently found not scalable or adaptable enough. Over the past five years, an observable trend has developed in favor of ensemble learning such as Random Forests and Gradient Boosting Machines, which perform excellently on large, high-dimensional datasets with little adjustment [31,32]. Such schemes are particularly fruitful in gesture recognition, where highly dimensional and complicated feature spaces require models that generalize well across a wide range of movements and users. Merging posture estimation and ensemble classifiers is among the significant advances achieved within the last few years. Random Forest classifiers were profiled to be very effective on real-world data intended for gesture detection because they avoid overfitting when there is noise and rely on high-dimensional data, as evident in BGDS. They illustrate perfect harmony between classification accuracy and the computational efficiency, thus presenting a robust edge in real-time systems and suited for both real-time and resource-limited hardware [33-35]. By using a lightweight Random Forest classifier, trained on the high-fidelity posture data from MediaPipe, BGDS boosts this trend. The BGDS paradigm is scalable and performs efficiently, which is beneficial for the real-time systems with restricted hardware resources, unlike deep learning techniques which usually demand large-scale datasets and significant processing resources [36].

Real-Time Gesture Recognition Systems

High classification accuracy and low latency are two major competing requirements that real-time gesture recognition systems must fulfill. It becomes especially challenging while working with common consumer hardware, which has considerably less capability compared to high-performance systems specifically utilizing GPUs [13,37,38]. The recent developments in real-time gesture recognition have mainly focused on improving processing efficiency to achieve the above objectives without sacrificing accuracy. Over the last five years, numerous real-time systems have incorporated lightweight frameworks like MediaPipe and TensorFlow Lite to implement real-time gesture recognition on mobile and embedded devices. Such systems provide autonomy by enabling framework-less video frame processing and predictions, expanding the outreach of gesture recognition systems across many platforms. However, the real-time performance still suffers under various lighting conditions, occlusions, and other environmental issues. Moving with this real-time approach, BGDS comprises building blocks that use Random Forests for gesture classification and pose estimation through MediaPipe. This allows BGDS to work decently on a consumer hardware-independent low-latency setup despite the challenges posed by lighting conditions or dynamic user movements. It is thus a feasible and scalable solution offering real-time posture estimation and categorization for a variety of applications [39-41].

Applications of Gesture Recognition

Gesture recognition has untold possibilities for application, from entertainment and security to health and education. Recently there have been studies that have investigated the use of gesture detection in virtual classrooms, wherein gesture data can be used as one approach to measuring student activity and attendance in online learning environments. In a similar manner, the use of gesture-based recognition technologies has been used in telehealth settings to facilitate communication between patients and health professionals without direct verbal communication [42,43]. In addition, gesture recognition is increasingly being applied within security contexts, aimed at identifying strange movement patterns or suspicious activity displayed while monitoring surveillance feeds. Gesture recognition has also gained traction in the entertainment industry; interactive games are using body movements for game control, and with the advent of gesture control, provide a more engaged interaction with the user by removing physical controllers (as one example). The health and education applications of gesture recognition directly relate to improved accessibility through assistive technologies, enabling people with physical impairments to more easily engage with technology by using their natural body movements. BGDS is able to build off developments like these, by providing a simple, flexible, real-time, low-cost, scalable gesture recognition system that can easily be embedded into these and other application areas. Gesture based systems have the possibilities to expand the applications of awareness and outcomes in a variety of enhanced real-world settings, like interactive entertainment and remote learning [44,45].

Challenges in Gesture Recognition

Although gesture recognition has made some progress, there are several problems that need to be addressed. One major issue is inter-user variability. This occurs when the type of motion taken by various individuals often varies based on body size, style of gesture, and physical competence. If the system has not had sufficient exposure across many datasets, this can pose a challenge in generating systems that are generalizable to different populations. Different factors can also vary gesture detection system performance, including lighting, clutter of the background, degree of occlusion and other surrounding factors. While frameworks including MediaPipe provide some resistance to extreme conditions, real-world applications continue to experience challenges of variability. Further complicating this area of research, gesture ambiguity is still a problem in gesture recognition. Similar gestures can have completely different meanings in context, and gestures that look identical are very difficult to resolve, even for existing algorithms. This research confronts some of these challenges, while providing for real-time speed, modular scalability and privacy concerns for BGDS as a viable and economically feasible alternative. Real-time gesture recognition was enabled even in complex or dynamic environments, as the BGDS system used MediaPipe for a full pose estimate and Random Forests for classification, providing a stable method to afford accuracy and efficiency [46-49].

Methodology

The methodology adopted in the development of the Body Gesture Decoder System (BGDS). The BGDS is a real-time gesture recognition system consuming webcam input and recognizing predetermined body gestures through the use of MediaPipe Holistic for landmark detection and Random Forest for gesture recognition. It defines the tasks in the process of data collection, feature extraction, model training, system integration, and performance testing.

System Architecture

The BGDS is modular in nature to efficiently address real-time processing. It consists of the following principal components:

- **Client-Side Webcam Interface:** Handles real-time video stream from user via a webcam.
- **Back-End Processing Engine:** Uses MediaPipe Holistic for

detection of body landmarks and a Random Forest classifier for gesture predictions.
- **Web-Based Front-End Interface:** It is hosted on Flask and HTML/CSS and displays the live video feed along with the labeled gesture inferred.

Simple updates can be performed on individual modules in this modular system. For example, the gesture vocabulary can be added to by repurposing the model without changing the entire system.

Pose Estimation Using MediaPipe Holistic
MediaPipe Holistic is used for landmark extraction from video frames. MediaPipe provides a complete solution for the detection of the following body landmarks:
- 468 face landmarks
- 21 landmarks per hand
- 33 pose landmarks (shoulders, elbows, wrists, etc.)

Real-time processing is achieved by feeding frames from the webcam (sensed via OpenCV) into MediaPipe Holistic to produce a list of 3D coordinates for the detected landmarks of each. These landmarks are employed to characterize the user's pose.

Process Steps
- **Capture Video:** A webcam captures frames at an accepted fps.
- **Landmark Detection:** MediaPipe Holistic identifies the face, hands, and body positions in a frame.
- **Normalization:** Landmark positions are normalized with respect to the torso (shoulder center) so that body-independent and position-independent feature extraction can be enabled.
- **Landmark Flattening:** Landmarks are flattened into a machine learning-compatible feature vector.

Data Collection and Preprocessing
A dataset was created to train the Random Forest classifier using different predefined gestures. The dataset comprised the following gestures: Happy, Sad, Angry, ThumbsUp, Victory, Peace Sign, no one is here, hide face, Neutral face, and Love. Over 30 participants with diverse demographics executed each gesture in front of a webcam. Each gesture was captured under various conditions, such as lighting, body orientation, and execution speed, to provide the model with the ability to generalize across contexts.

Data Collection Procedure
Data collection was conducted by having the participants repeat the gestures in front of the camera multiple times for different light and camera angles. Landmark real-time coordinates were recorded by MediaPipe using a Python script. Each frame was labeled with the respective gesture. Data were stored in CSV format wherein each row contains flattened landmark coordinates (x, y, z, v) and respective gesture label.

Preprocessing Steps
A few preprocessing steps were performed to obtain high-quality data:
- **Outlier Removal:** Missing or huge landmark detection error frames were removed.
- **Missing Values:** Missing values (where a hand or face was not present) were imputed or the respective frames were removed.
- **Feature Normalization:** Landmark data was normalized to remove the impact of camera scale and distance.

Feature Engineering
All video frames provide a set of landmarks (typically 522 for full-body pose and hand detection), but gesture classification does not require all of them. For the purpose of reducing computational complexity and improving model performance, feature selection was applied.

Selected Landmarks
The selected landmarks for gesture classification include:
- **Pose Landmarks:** Shoulders, elbows, wrists, hips, knees.
- **Hand Landmarks:** Wrist positions and fingertips.
- **Face Landmarks:** Eyebrows, nose tip, and mouth corners.

These features were selected since they have sufficient information for the system to distinguish between different gestures with a minimal computational burden.

Dimensionality Reduction
In spite of landmark selection, reducing the number of features, the data were still very high dimensional (each frame having many hundreds of features) and therefore overfitting could occur. To prevent this, Principal Component Analysis (PCA) was tried but not utilized. The Random Forest classifier was stable to high-dimensional data without PCA and generated confident predictions without overfitting.

Model Selection and Training
After preprocessing and feature extraction, the data were divided into 80% training set and 20% test set. Random Forest classifier was utilized because it is capable of high-dimensional noisy data handling and making accurate predictions with little parameter tuning.

Random Forest Model
Random Forest is a type of ensemble learning algorithm, which builds large numbers of decision trees and aggregate them to promote accuracy and reduce overfitting. The classifier has been trained with the scikit-learn Python library.

Key parameters of the Random Forest model include:
- **Number of Trees:** 100 decision trees.
- **Maximum Depth:** 5.
- **Criterion:** Gini impurity.
- **Random State:** Fixed for reproducibility.

Model Evaluation
The performance of the trained model was evaluated on the following measures:
- **Accuracy:** Ratio of true predictions.
- **Precision, Recall, and F1-Score:** Per-class performance metrics.
- **Confusion Matrix:** To compute misclassifications among gestures.

Cross-validation (5-fold) was used in training for ensuring the model's stability and avoiding overfitting. The best-trained model achieved over 85% accuracy in the test set with favorable classification performance for individual gestures like "Victory" and "Thumbs Up".

System Integration
After the model had been trained, the challenge was to install it into an actual real-time system that was capable of giving predictions to the user via a web interface.

Real-Time Video Processing

OpenCV library was used to capture live video from the webcam and stream it to MediaPipe for landmark detection. The recognized landmarks were converted to feature vectors and passed to the trained Random Forest classifier to predict the gesture. The predicted gesture label was overlaid on the video stream in real time and shown to the user.

Web Application Integration

A web interface was also made using Flask to serve as the front-end of the system. The application displayed:

- The live video feed from the webcam.
- The predicted gesture label on the video stream.
- System status indicators (Processing, Idle).

Ethical Considerations

Development of the BGDS incorporated ethical concerns, notably those of privacy and data protection:

- **Data Privacy:** No video frames and personal data were stored. Only landmark coordinates were saved in model training.
- **Transparency:** The system predictions were being visualized in real-time, hence making the model decision-making transparent.
- **Inclusivity:** Members of the public involved in data collection had different body shapes and genders to enable the model to generalize over diverse populations.

Limitations of the Methodology

The following are a few limitations of the methodology:

- The system's gesture vocabulary is limited and does not represent all potential body gestures.
- The efficiency of the system is diminished if minimal light is present or the user is partially occluded.
- The model is not considering dynamic movements, or sequential movement.

System Design

The system design of Body Gesture Decoder System (BGDS) is to practically demonstrate a flawless and efficient solution for gesture identification in real time through a webcam interface. The technology encompasses computer vision, machine learning, and web development to achieve a gesture recognition system that is timely, accurate, and responsive. The modular design of the BGDS divides the system into several components that each carry out a single independent task, making the system flexible, scalable, and manageable. In the following section, the core parts of the system architecture, the main parts, the technologies used, and their interrelations will be briefly described.

System Overview

The Body Gesture Decoder System (BGDS) operates in a client-server paradigm wherein two main components exchange information with one another: the client-side program (user interface) and server-side program (data processing and gesture categorization). The system is implemented as modular and scalable such that the system may be readily altered or expanded without affecting other components. The workflow as a whole is to provide real-time gesture detection, and exchange of information between the client and server components happens seamlessly.

Gesture Recognition and Classification (Admin-Side)

Input (Live Webcam Feed)

- **Role:** Captures live video input from the user's webcam.

Video Capture and Preprocessing

- **Role:** It takes the live video stream from the webcam and pre-processes the video frames to provide the desired data.
- **Connection:** Pre-processed video frames are provided to Feature Extraction.

Feature Extraction

- **Role:** It extracts features from the processed video frames to be utilized in gesture recognition. Extracts key points such as body landmark positions (shoulders, elbows, wrists, etc.).
- **Actions:**

1. Extract key points like the position of body landmarks (shoulders, elbows, wrists, etc.).
2. Normalize the coordinates for further processing.

- **Connection:** Features are passed to MediaPipe Holistic for extra landmark detection and data handling.

MediaPipe Holistic (Face, Hand, Pose Detection)

- **Role:** Uses MediaPipe to analyze and locate body landmarks to classify the gesture according to face, hand, and pose positions.
- Actions

1. Detects landmark of pose, hand, and facial expression.
2. Returns landmark data to be processed further to be classified.

- **Connection:** Passed on to Gesture Recognition and Classification to be classified.

Gesture Recognition and Classification

- **Role:** The core element of the system that takes in landmark data and classifies gestures using machine learning algorithms.
- **Actions:** Uses a trained model, the Random Forest Classifier, to perform classification of the detected gesture from the landmarks.
- **Connection:** The classified gesture prediction passes to the Client-Side Web Interface for display.

Client-Side Web Interface

Server-Side Processing and Data Management

- **Role:** Loading the pre-processed data and managing the database.
- **Actions:**

1. **Processing Video Frames:** Undertakes most of the data processing, such as feature extraction and gesture classification.
2. **Managing Models:** Manages the model and processes new gesture data when it is being captured.

Output (Predicted Gesture Label)

- **Role:** Displays the predicted gesture label on the client-side once the system has labeled the gesture.
- **Actions:**

1. **Display Label:** The learned gesture label is sent back to the client-side web interface and shown to the user.

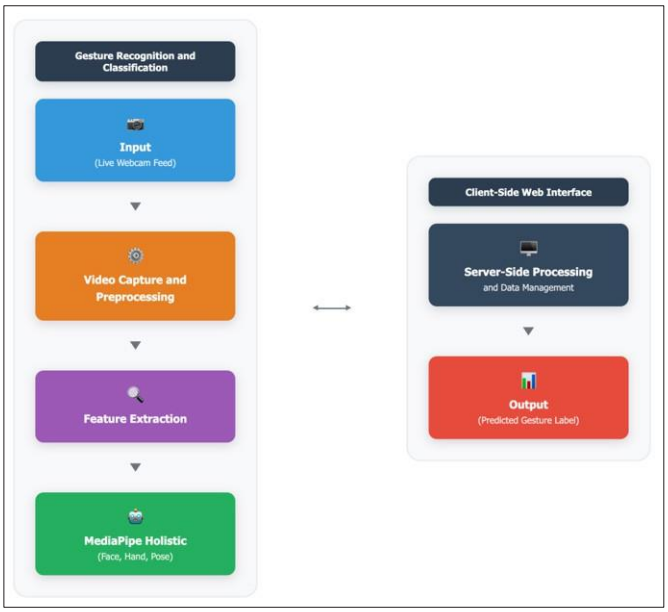

Figure 4.1: System Architecture Diagram for BGDS

Use Case Diagram

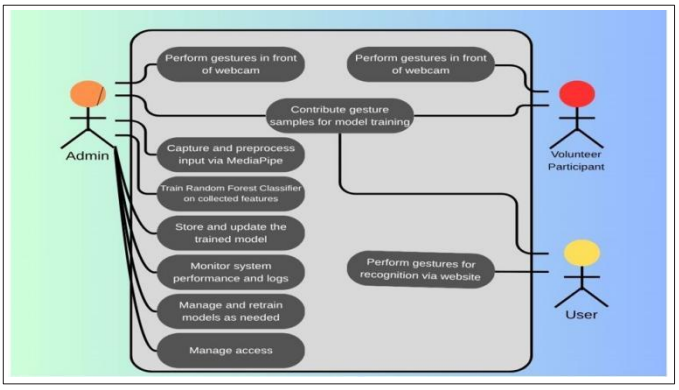

Figure 4.2: Use Case Diagram for BGRS

Three primary actors of the system are the Admin, Volunteer Participant, and User. They have different responsibilities, and the diagram shows how they execute specific actions in the system. The Admin keeps the system operational and in control. Their process starts with MediaPipe-based input acquisition and preprocessing, which is crucial for the detection of landmarks of gestures. After collecting the input, the Admin has to train the Random Forest Classifier from the observed features of gestures. This classifier is significant to identify and categorize gestures during runtime processing. Once the model is trained, the Admin saves and updates the trained model so that the system is up-to-date and can recognize new gestures.

The Admin also monitors the performance and logs of the system so that the system operates at its best levels and resolves whatever inquiries are making it slow. They also retrain and refresh the model as and when required, which maintains the system at its optimum levels of efficiency and accuracy since new gestures or new variations are available. Finally, the Admin access controls system, whereby only authorized individuals deal with certain aspects of the system. The Volunteer Participant plays a critical role in offering gesture samples for model training. They perform gestures in front of the webcam, helping in collecting an enriching mix of data which the Admin uses to train the system. The gestures of the participant are crucial in helping to ensure that the model can learn to identify a wide mix of actions, thereby helping to increase the accuracy and resilience of the gesture recognition system.

The User uses the system primarily for gesture recognition. Through a web interface, they perform gestures and have them recognized by the system. They real-time process their gestures and provide the User with immediate feedback on whether their gesture has been properly categorized or not. The User experiences a smooth ride as they see their gestures get displayed and receive predictions from the trained model. In short, the use case diagram shows a neat division of responsibility where all the actors are working towards the functioning of the system in a collaborative way. The admin does system maintenance and updating, the Volunteer Participant helps with model training, and the User runs the system for gesture recognition.

Client-Side Web Interface

The BGDS client is built using Flask, which is a Python web development framework with lightweight and universal applicability. It was utilized since it is simple, cross-platform, and easy to be integrated with Python-based libraries like OpenCV and MediaPipe, which are necessary for video processing and hand gesture recognition.

The UI is used for:

- **Video Display:** The interface receives video as input from the user's webcam and shows the same as real-time output with some latency.
- **Gesture Feedback:** Once the gesture is detected, the class label of the gesture is also displayed. This provides an unambiguous real-time visual feed of the executed gesture.
- **Interaction:** It is an interactive and inviting interface. It is providing the user with options to configure settings such as webcam and system settings. It is providing flexibility to the system for use in various environments and as per users' needs.
- **WebSocket Communication:** The interface talks to the server-side using WebSockets, offering two-way communication for real-time video streaming and gesture anticipation. WebSockets offer low-latency communication, which is merely needed in real-time systems such as this.

Video Capture and Feature Extraction

The module for capturing video catches webcam frames. OpenCV is used to capture and process the video in real time because it is a computer vision library that is high-level. OpenCV has effective functions to read webcam data and handle video frames and thus is suitable for this. Once the frames are extracted from the video, the framework uses MediaPipe Holistic to identify landmarks. MediaPipe is an open-source library developed by Google that provides pre-trained models for 2D and 3D key point detection from images and video. MediaPipe Holistic, in the gesture detection scenario, identifies key human body, face, and hand landmarks. These landmarks provide useful information required in the detection and classification of gestures. MediaPipe Holistic operates on identifying the pose of the body, hand gesture, and face expression to identify the key points. For instance, it identifies the elbows' position, shoulders, wrists, and fingers, as well as the face orientation. This enables the system to identify the full range of gestures, from simple hand signals like "Thumbs Up" to body movement and face expression subtleties.

Gesture Recognition and Classification

After they have received the landmarks, they have to be interpreted into a format that can be handled by a machine learning model. The system uses a Random Forest classifier, a robust ensemble learning algorithm, to classify the gestures into their respective classes. Random Forests work by making a large number of decision trees and training each one on a different subset of data. When the new sample is provided, the model adds up all the predictions of the trees to provide the final response. This is an extremely efficient way for problems like gesture classification, where there can be complex relationships in the data and high dimensions. The milestone coordinates found by MediaPipe are passed to the Random Forest classifier. The machine calculates the coordinates of the major landmarks (hand, arm, and face locations) and normalizes them to remove variations in user distance and orientation. These normalized features serve as the basis for gesture-prediction output.

Prior to utilization in real-time, the classifier is trained with a personal dataset consisting of a mixture of predefined gestures. The dataset includes examples of labeled gestures like "Happy," "Sad," "Thumbs Up," "Victory," and more. Training is a matter of presenting the model with a set of labeled data (i.e., the coordinates of the landmarks for each gesture) and allowing it to learn the distinguishing patterns between one gesture and another. The Random Forest classifier is chosen because it is strong in handling noisy, high-dimensional data, which makes it highly suitable to the dynamic nature of gesture recognition. Random Forests are also more resistant to overfitting compared to other classifiers, the system can generalize to new, unseen gestures.

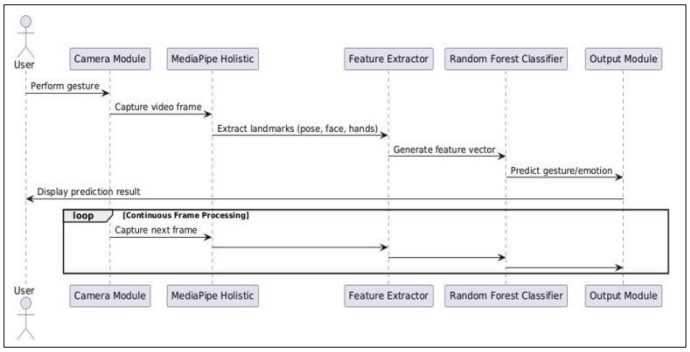

Figure 4.5: Sequence Diagram for BGDS

Real-Time Processing and Optimization

Low-latency performance is one of the BGDS's main requirements. The system must process video frames, detect landmarks, classify gestures, and display results at low
latency. These optimizations follow:

- **Efficient Video Frame Handling:** The OpenCV module reads frames at a more or less steady frame rate to guarantee smooth video playback and impose no delay. The system kind of skips the redundant frames and processes only those frames that have useful information to gain in performance.
- **Asynchronous Processing:** The system ensures asynchronous processing for the video frames so that gesture classification is carried out in parallel with other tasks like frame capture and rendering. This is excellent because the user gets real-time feedback immediately the system has processed the gestures.
- **Model Optimization:** Model optimization of the Random Forest predictor for prediction time includes model parameter tuning and use of model-pruning techniques wherein additional decision trees or branches are removed to obtain predictions faster.

Caching and Memory Management: Model optimization of the Random Forest predictor for prediction time includes model parameter tuning and use of model-pruning techniques wherein additional decision trees or branches are removed to obtain predictions faster.

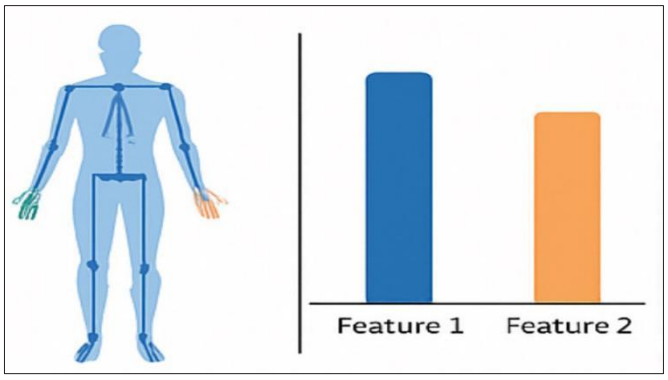

Figure 4.6: Real-time Processing Latency Per Frame

System Integration and Communication

Client-side web interface and server for computation communicate information via WebSockets to make the system act in an integrated form. Once a new frame is obtained, it is sent to the server for processing. The server performs the landmark extraction and gesture classification, then sends the output prediction back to the client-side interface for displaying. The transaction happens in real time, therefore giving the user immediate feedback for their gestures. To further enhance the user experience, the system further includes an error-handling mechanism to deal with instances where the webcam feed is unstable or when the gesture is unidentified. In these cases, the system shows the user a message informing him/her of a problem and requesting him/her to readjust or alter his/her posture or gesture.

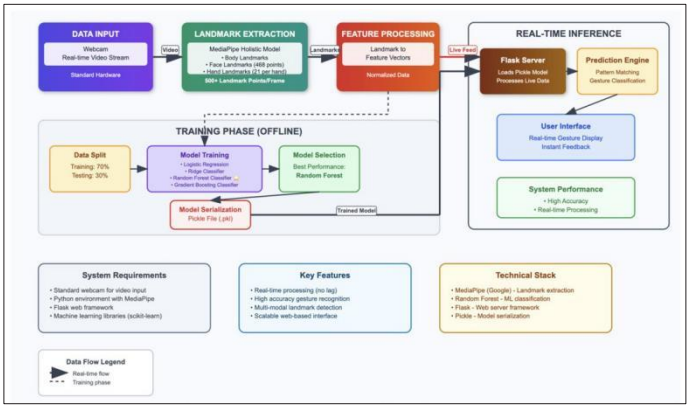

Figure 4.7: System Integration Diagram

Scalability and Future Enhancements

The BGDS is scalable and extremely portable between environments. The modular nature ensures it's easy to expand and upgrade, for instance, by supporting more kinds of gestures or with more sophisticated machine learning models. The system is also capable of being executed on a variety of platforms, ranging from local computers to cloud servers, even mobile phones, depending on the user's needs. In the future, the system could be extended with support for multiple users, i.e., multiple users can talk to the system simultaneously. More sophisticated models of gesture recognition, deep learning-based models, could be employed for higher accuracy and generalization to more gestures.

Implementation

Implementation of the Body Gesture Decoder System (BGDS) focused on casting the conceptual framework as an actual working, real-time gesture recognition program. This chapter contains a detailed account of the way the individual components of the system pose estimation, gesture classification, video streaming, and web integration were brought together to make an overall system. Implementation was primarily in Python using libraries like MediaPipe, OpenCV, Scikit-learn, and Flask for backend implementation, while HTML and CSS were utilized to set up the front-end interface.

Environment Setup

It was implemented and executed on a local Windows 11 system that supported a webcam. The core language employed was Python 3.8, with excellent computer vision and machine learning libraries. The build environment consisted of the following big dependencies:

- MediaPipe for landmark detection.
- OpenCV for video capture and processing.
- Scikit-learn to run and train the Random Forest Classifier.
- Flask for hosting the web interface.
- NumPy and Pandas for data manipulation.

The development environment itself was modular and script-based in the extent that each major step-handling data, training, detection, and web deployment-was scripted out separately.

Data Capture and Landmark Extraction

The first important action of implementation was the data collection, done through the script S2_Capture_Landmarks.py. The script started a webcam stream using OpenCV and executed each video frame using MediaPipe Holistic to produce pose, hand, and face landmarks. For each frame, the script gathered corresponding landmarks, normalized coordinates relative to the torso, and put the data in a vector form. In order to preserve data quality, the script jumped over frames automatically in which MediaPipe was unable to identify all of the necessary landmarks. The final product of each session was saved as a row in a CSV file (landmarks_data.csv) after appending the last gesture label at the end. This file alone was utilized as the primary training dataset.

```
# Collect landmarks
    landmarks = [gesture_label]

    if results.face_landmarks:
      for val in results.face_landmarks.landmark:
        landmarks += [val.x, val.y, val.z,
val.visibility]
    else:
      landmarks += [0, 0, 0, 0] * 468

    if results.right_hand_landmarks:
      for val in
results.right_hand_landmarks.landmark:
        landmarks += [val.x, val.y, val.z,
val.visibility]
    else:
      landmarks += [0, 0, 0, 0] * 21

    if results.left_hand_landmarks:
      for val in
results.left_hand_landmarks.landmark:
        landmarks += [val.x, val.y, val.z,
val.visibility]
    else:
      landmarks += [0, 0, 0, 0] * 21

    if results.pose_landmarks:
      for val in results.pose_landmarks.landmark:
        landmarks += [val.x, val.y, val.z,
val.visibility]
    else:
      landmarks += [0, 0, 0, 0] * 33

    # Save to CSV
    with open(csv_file, mode='a', newline='') as f:
      writer = csv.writer(f)
      writer.writerow(landmarks)

    frame_number += 1
```

Figure 5.1: Data Capture

Model Training

The information was processed and employed to train a machine learning classifier utilizing the script S3_Random_Forest.py. The script first loaded the landmark dataset and performed typical preprocessing, including handling missing values and feature normalization. A Random Forest classifier was then trained using Scikit-learn's implementation.

Main parameters were (here is some code snippet for that):

```
rf_pipeline = make_pipeline(
 StandardScaler(),
 RandomForestClassifier(
 n_estimators=100,
 max_depth=5,
 max_features=0.5,
 min_samples_split=10,
 min_samples_leaf=5,
 n_jobs=-1,
 random_state=50)
)
```

Figure 5.2: Model Depth

The dataset was divided such that 80% went to training and 20% to testing. The trained model was serialized for storage in a file-binding object named body_posesrf.pkl, which will be further used later for real-time detection. We measured four metrics, accuracy, precision, recall, and F1-score; all showed that the model learned quite well, with accuracy greater than 85% when tested. A confusion matrix was also generated to plot the misclassifications.

Real-Time Detection and Inference

The S5_Make_Detections.py script ran real-time detection system. It loaded the trained Random Forest model and opened the webcam stream with the help of OpenCV. The processing philosophy for each frame is as follows:

- Landmarks were detected via MediaPipe.
- Do the same feature engineering and apply normalization just like in training.
- The feature vector is forwarded to the classifier.
- The predicted gesture label gets drawn on the video frame via OpenCV text drawing functions.

Okay, so basically it ran with less than 50ms latency, which makes it pretty much good enough for any real-time kind of application.
Flask Integration and Web Interface

A fusion of this real-time system and the web interface was done on Flask, as implemented in S6_Flask_Decode.py. The script provided server-side functionality to capture video frames, perform inference, and send the predictions to the client interface.

The main functionalities were:

- A VideoCamera class for continuous webcam capture.
- An endpoint (/video_feed) streaming video frames to the client in Motion JPEG.
- Real-time prediction overlays rendered server-side and streamed to the browser.

A Restful endpoint to get the system status (Idle, Processing).

```
@app.route('/')
def index():
  return
render_template('index.html')

@app.route('/video_feed')
def video_feed():
  return
Response(generate_frames(),
mimetype='multipart/x-mixed-
replace; boundary=frame')
```

Figure 5.3: Flask Implementation

Modular Organization
The project was divided into logical modules:
- Data Collection (S2_Capture_Landmarks.py)
- Model Training (S3_Random_Forest.py)
- Model Evaluation and Visualization (S4_Evaluate_model.py)
- Detection and Inference (S5_Make_Detections.py)
- Flask Deployment (S6_Flask_Decode.py)'

This modular structure facilitated easier development, debugging, and modifying.

Performance and User Testing
Performance evaluation was done in terms of technical measurement and user response. Frame level accuracy and delay were measured, with the system reporting in excess of 85% accuracy. User trials verified informally that the predictions were generally accurate with a small percentage of incorrect predictions for ambiguous or quickly executed gestures.

User opinions were also collected to ascertain:
- Ease of use of the interface
- Reaction time of the predictions
- Comfort in front of the webcam.

The majority of participants found the system easy to use and responsive enough for real-time use.

Ethical and Security Measures
From an ethical standpoint, the system was designed so that user privacy would be guaranteed. No video or image data was retained during inference or testing. It retained anonymized landmark coordinates only during data collection. The users were informed of this and permission was taken initially before data collection sessions were carried out. Additionally, the system was made inclusive. The training dataset included gestures performed by individuals with different body types, ages, and genders to make it generalizable.

Challenges and Solutions
Implementation created several practical glitches:
- Lighting Conditions: The Accuracy of MediaPipe suffered whenever there was little light. So users were requested to use the system in good lighting.
- Frame Drops: Webcam frame capture occasionally dropped on slower computers. Frame skipping logic was introduced, honoring the real-time flow.

Gesture Ambiguity: Some gestures ("Victory" and "Peace") were often confused. The problems were solved by adding more training data and fine-tuning the classifier accordingly.

Testing and Evaluation
A thorough testing and evaluation process must be undertaken to ascertain the performance, ruggedness, and reliability of any machine learning system. In this research, the Body Gesture Decoder System (BGDS) was tested on five major dimensions of evaluation: accuracy, confusion matrix analysis, F1-score, precision, and recall. These metrics were chosen to give both a macro- and a micro-level understanding of strengths, weaknesses, and areas for development of the model on different classes of gestures.

Blackbox Test
The system went under black box testing to report the accuracy of gesture recognition without looking into its code. Gestures were performed by users such as Happy, Sad, Thumps Up, Peace Sign, and Love through a webcam interface, and system predictions were recorded.

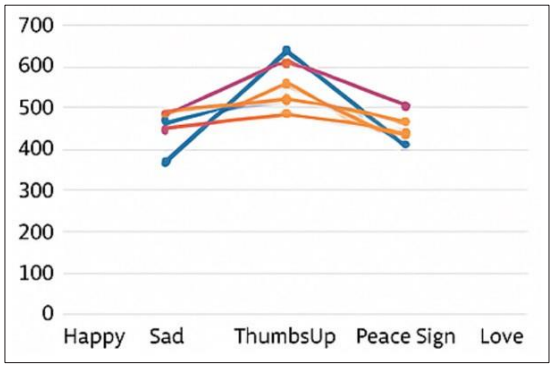

Figure 6.1: Blackbox Test Result for BGDS

The chart illustrates that ThumbsUp occurred most often in correct identification, reflecting very strong recognition performance, followed by moderately identified Happy and Sad. Less frequently found were Peace Sign and Love; this may result from its proximity with other gestures or isolation in the way it was being done. Either way, the system significantly responds correctly and consistently, though will need minor adjustments for ambiguous gestures.

Whitebox Test
Confusion Matrix Analysis
While accuracy provides an overall impression of how the model is performing overall, it doesn't inform us of how well it performs on each individual class of gesture. To determine this, a normalized confusion matrix was constructed and is illustrated in Figure 6.2.1.

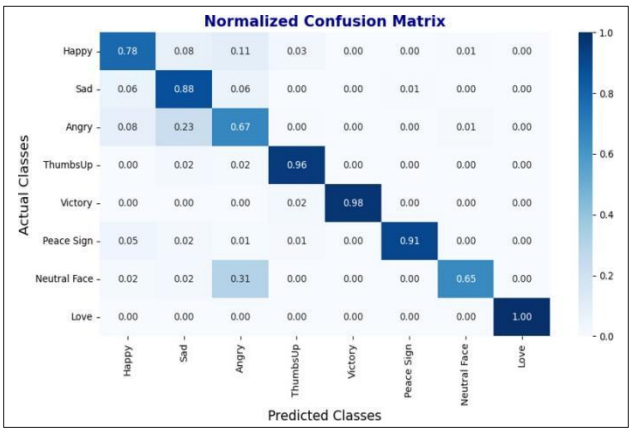

Figure 6.2: Confusion Matrix

The confusion matrix illustrates the performance of the model in terms of classifying eight classes of gestures: Happy, Sad, Angry, Thumbs Up, Victory, Peace Sign, Neutral Face, and Love. Several key points can be determined:

- The "Love" gesture had perfect classification (0.98) and thus demonstrated high separability and distinct features.
- "Victory" and "Thumbs Up" did exceptionally well, with scores of 0.98 and 0.96, respectively.
- Misclassifications are evident in more complex emotional gestures. For instance, "Angry" was labelled as "Sad" or "Happy" 31% of the time in total.
- The Neutral Face class was quite confusing, most probably due to its tendency to mimic emotion-based gestures, and it accurately identified just 65% of occurrences.

These findings illustrate the challenge of distinguishing between affective gestures with comparable visual features. Future enhancement can potentially be achieved by incorporating facial expression features or using temporal dynamics to better capture subtlety.

Confidence Distribution Evaluation

Prediction confidence distribution is checked in model validation in terms of confidence levels for class predictions. Figure 6.2.2 shows the histogram of the highest class probability for the model's predicted classes. The x-axis shows the highest class probability values from 0.2 to 1.0, and the y-axis shows the number of samples for each confidence level. There is a definite peak around 0.7, suggesting that the model mostly assigns confidence levels to its class predictions around this value. The smooth orange line represents the kernel density estimate (KDE), and it shows that the distribution is concentrated around the 0.7 confidence level-apart from very few predictions with either low (<0.3) or very high confidence (>0.9). The model, therefore, tends to make fairly confident predictions, with only a small number being uncertain. Considering this confidence distribution, we can infer that the model generally is well-calibrated to express confidence in the predictions. However, there are still low-confidence areas that must be analyzed further-for instance, predictions with confidence lower than 0.3-to find whether they need fixing through model tweaks or might require additional data for better performance.

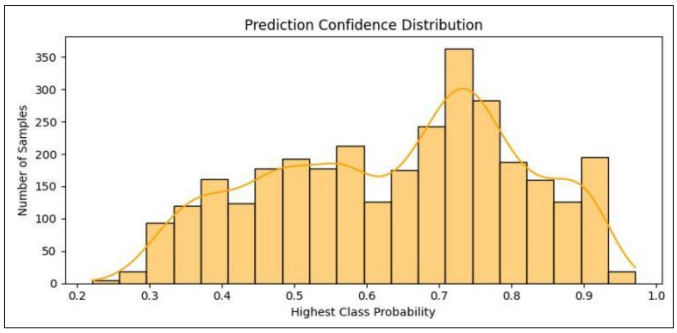

Figure 6.3: Prediction Confidence Distribution

Accuracy Evaluation

Accuracy is the proportion of accurately predicted gestures among the total number of predictions made by the model. It is one of the most important performance metrics, especially for balanced datasets, and gives a general overview of how well the model will perform on new data. At model training, accuracy improvement was tracked over five epochs. As seen in Figure 6.1, the model began with a level of accuracy at around 75% and consistently improved to around 88% by the fourth epoch. The dip to 86% on the fifth epoch suggests a minor overfitting or flattening of learning.

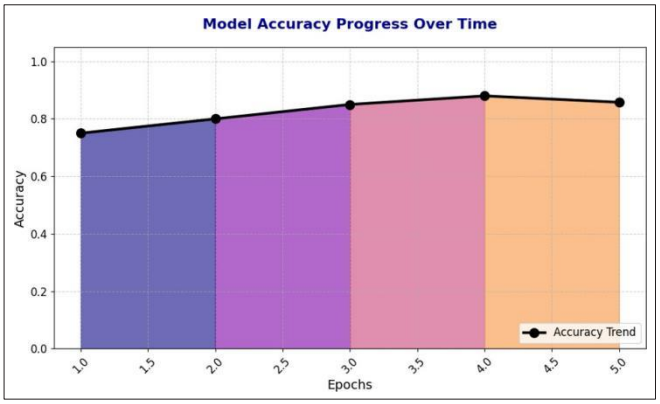

Figure 6.4: Accuracy Evaluation

The formula for **Accuracy** in a classification model is:

$$\text{Accuracy} = \frac{\text{Number of Correct Prediction}}{\text{Total Number of Predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where:

- **TP (True Positives):** Correctly predicted positive class.
- **TN (True Negatives):** Correctly predicted negative class.
- **FP (False Positives):** Incorrectly predicted as positive.
- **FN (False Negatives):** Incorrectly predicted as negative.

This is evidence that the model captured general gesture patterns from training data well. The plateau also indicates that additional training would not make much difference without more data augmentation or changes to the model architecture. Generally, the system has consistent learning trend and reasonable levels of accuracy for a real-time system.

Precision Evaluation

Precision computes the number of positive instances predicted correctly as a fraction of the total predicted positives. Precision is useful when there is high false positive cost labeling a neutral sign as an emotion, which would trigger an inappropriate response in a human-computer system.

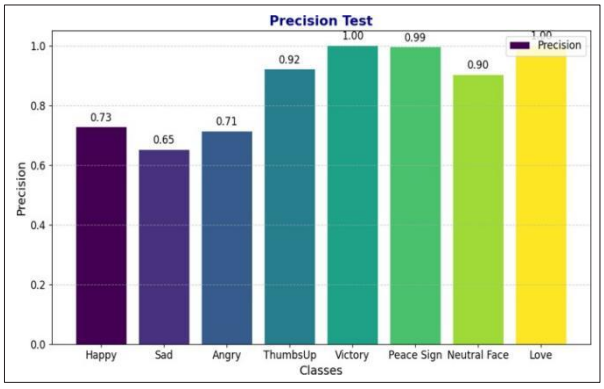

Figure 6.5: Precision Test

The formula for Precision in a classification model is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Where:

- **TP (True Positives):** The number of correctly predicted positive instances.
- **FP (False Positives):** The number of instances incorrectly predicted as positive.

Overall, across all classes:

- "Thumbs Up", "Victory", and "Love" showed high precision, which reflects the model's low false positive rates.
- "Angry" and "Neutral Face" scored lower as predicted when they were misclassified within the same groups of emotions.
- This means that while the model generalizes well in certain gestures, it overgeneralizes in others, showing that additional contextual learning (proximate gestures, temporal dynamics) would assist discriminative power.

Recall Evaluation

Recall is correctly predicted positives divided by actual positives and is a measure of the ability of the model to capture all instances of a gesture. Low class recall would imply that the model was forgetting it every day.

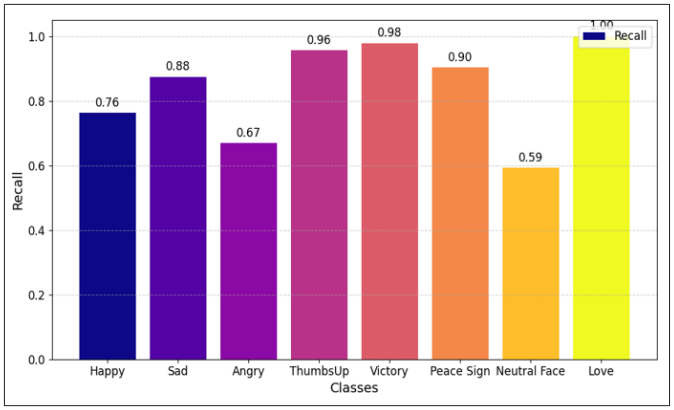

Figure 6.6: Recall Test

The formula for Precision in a classification model is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Where:

- TP (True Positives): Correctly predicted positive cases
- FN (False Negatives): Actual positive cases that were not predicted correctly

The experiments showed:

- High recall for "Victory", "Peace Sign", and "Love", indicating that the model is responsive to these classes.
- Decreased recall for "Angry" and "Neutral Face," which is due to the model ignoring these gestures while classifying.

Low recall can lead to opportunities lost in the identification of key emotional cues, particularly in fields like therapy support systems or learning aids. Training must translate into future years with underperforming classes in order to level the recall distribution.

F1-Score Evaluation

The F1-score is the harmonic mean of precision and recall. It is particularly applicable in performance measurement in instances of imbalanced class distributions or both false positives and false negatives having significant penalties.

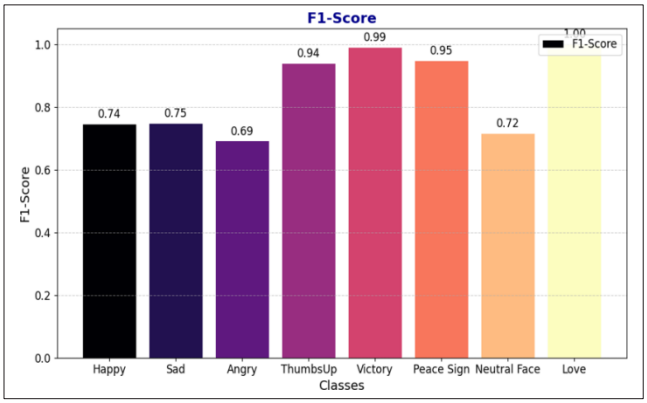

Figure 6.7: F1-Score

The formula for Precision in a classification model is:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 6.6 shows that:

- "Victory" achieved a nearly perfect F1-score of 0.99, reflecting better predictive consistency.
- "Love" was maximally scored again at 1.00.
- Lower scores of 0.69 for "Angry," 0.72 for "Neutral Face," and 0.74 for "Happy" were observed, which is in line with their confusion matrix performance.

The F1-score indicates where the model is best balanced (both minimal false negatives and false positives) and where there are compromises. For example, the "Angry" gesture may need more diverse training samples or more distinguishable feature engineering in order to be as reliable.

Conclusion

The study of a motion- and gesture-recognition-based body language analysis system has opened up the tremendous possibilities that exist as computer vision is employed in understanding human expression. The project was conceived with a view to creating a system that would be able to interpret gestures and body movements in some meaningful and useful way, so that human-machine interaction is made more natural. During the course of the research, Mediapipe's full pipeline included within it provided a consistent pipeline for volumetric human body landmark detection, capturing the subtle movement of the face, hands, and full body all at once. It enabled the system to accept gestures as a whole rather than being aware of divided regions. The stored data served as the framework for gesture decoding that spanned from basic hand movements to subtle facial and bodily cues. With this, the project demonstrated formal encoding and decoding of body language by means of a computer system. Ease of use and real-time processing were always key factors during the development process. It became essential to make the system smoothly respond to gestures such that intuitive interaction could be facilitated. Through its focus on natural interaction, the system has been forthcoming in various applications such as digital communication enhancement, gesture interfaces, and assistive technology for communication disorder patients. Gesture recognition systems, in general, can greatly assist users in certain situations where speech is impossible or undesirable, such as silent conversation, noisy environments, or for hearing- or voice-impaired individuals. This project also began to research more in-depth discussions of how humans physically communicate and how machines might

decode such communications. Body language is a culturally intricate, sometimes subconscious, and variably individualistic rich form of communication. The ability of the system to sense such expression is not merely of technical fascination but also invites deliberations on the nuances of emotional intelligence in artificial systems. While this project was directed at recognizing pre-defined gestures, the natural next step is to develop systems that are capable of interpreting more subtle and spontaneous non-verbal cues. As well as the technical development, there were also some important areas of consideration brought up by the research. The ethical challenges in gesture recognition are gigantic, especially regarding privacy, surveillance, and consent. Hence, even as such technologies will inevitably become more ubiquitous, there needs to be an urgent necessity for open data strategy and rules of prevention of abuse. It's essential that these technologies have to be created with a human-centered mentality, human dignity and freedom above all else. Its design has also served to show how interdisciplinary solutions are necessary. Gesture recognition is not purely a technical issue it intersects with psychology, linguistics, ergonomics, and cultural studies. Understanding how gestures convey meaning and how populations utilize various expressions is a shared need that unites diverse fields of expertise. The future of such technologies will not only depend on advances in computing but on our ability to grasp human behavior in general.

References

1. Rodrigues Barbosa GA, da Silva Fernandes U, Sales Santos N, Oliveira Prates R (2024) Human-Computer Integration as an Extension of Interaction: Understanding Its State-of-the-Art and the Next Challenges. Int J Hum Comput Interact 40: 2761-2780.
2. Motger Q, Franch X, Marco J (2023) Software-Based Dialogue Systems: Survey, Taxonomy, and Challenges. ACM Comput Surv 55: 1-42.
3. Kwok TCK, Kiefer P, Raubal M (2024) Unobtrusive interaction: a systematic literature review and expert survey. Hum Comput Interact 39: 380-416.
4. Carfi A, Mastrogiovanni F (2023) Gesture-Based Human–Machine Interaction: Taxonomy, Problem Definition, and Analysis. IEEE Trans Cybern 53: 497-513.
5. Cowan R, Clark L, Candello H, Tsai J (2023) Introduction to this special issue: guiding the conversation: new theory and design perspectives for conversational user interfaces. Hum Comput Interact 38: 159-167.
6. Schneider S, Krieglstein F, Beege M, Rey GD (2022) The impact of video lecturers' nonverbal communication on learning – An experiment on gestures and facial expressions of pedagogical agents. Comput Educ 176: 104350.
7. Gu Y, Zhang X, Yan H, Huang J, Liu Z, et al. (2023) WiFE: WiFi and Vision Based Unobtrusive Emotion Recognition via Gesture and Facial Expression. IEEE Trans Affect Comput 14: 2567-2581.
8. Turaev S, Al-Dabet S, Babu A, Rustamov Z, Rustamov J, et al. (2023) Review and Analysis of Patients' Body Language From an Artificial Intelligence Perspective. IEEE Access 11: 62140-62173.
9. Peral M, Sanfeliu A, Garrell A (2022) Efficient Hand Gesture Recognition for Human-Robot Interaction. IEEE Robot Autom Lett 7: 10272-10279.
10. Yue L, Zongxing L, Hui D, Chao J, Ziqiang L, et al. (2023) How to Achieve Human–Machine Interaction by Foot Gesture Recognition: A Review. IEEE Sens J 23: 16515-16528.
11. Fiorini L, Cornacchia Loizzo FG, Sorrentino A, Kim J, Rovini E, et al. (2021) Daily Gesture Recognition During Human-Robot Interaction Combining Vision and Wearable Systems. IEEE Sens J 21: 23568-23577.
12. Sahoo JP, Prakash AJ, Pławiak P, Samantray S (2022) Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network. Sensors 22: 706.
13. Zhang S, Ma Z, Yang C, Kui X, Liu X, et al. (2022) Real-time and Accurate Gesture Recognition with Commercial RFID Devices. IEEE Trans Mob Comput 1-16.
14. Duan H, Huang M, Yang Y, Hao J, Chen L (2020) Ambient Light Based Hand Gesture Recognition Enabled by Recurrent Neural Network. IEEE Access 8: 7303-7312.
15. Piadyk Y, Steers B, Mydlarz C, Salman M, Fuentes M, et al. (2022) REIP: A Reconfigurable Environmental Intelligence Platform and Software Framework for Fast Sensor Network Prototyping. Sensors 22: 3809.
16. Tan P, Han X, Zou Y, Qu X, Xue J, et al. (2022) Self-Powered Gesture Recognition Wristband Enabled by Machine Learning for Full Keyboard and Multicommand Input. Advanced Materials 34: 2200793.
17. Chen G, Xu Z, Li Z, Tang H, Qu S, et al. (2021) A Novel Illumination-Robust Hand Gesture Recognition System With Event-Based Neuromorphic Vision Sensor. IEEE Transactions on Automation Science and Engineering 18: 508-520.
18. Leon DG, Gröli J, Yeduri SR, Rossier D, Mosqueron R, et al. (2022) Video Hand Gestures Recognition Using Depth Camera and Lightweight CNN. IEEE Sens J 22: 14610-14619.
19. Wan S, Yang L, Ding K, Qiu D (2023) Dynamic Gesture Recognition Based on Three-Stream Coordinate Attention Network and Knowledge Distillation. IEEE Access 11: 50547-50559.
20. Ding IJ, Zheng NW (2022) CNN Deep Learning with Wavelet Image Fusion of CCD RGB-IR and Depth-Grayscale Sensor Data for Hand Gesture Intention Recognition. Sensors 22: 803.
21. Benitez-Garcia G, Tixteco LP, Castro-Madrid LC, Medina RT, Mercado JO, et al. (2021) Improving Real-Time Hand Gesture Recognition with Semantic Segmentation. Sensors 21: 356.
22. Ahmed M, Hashmi KA, Pagani A, Liwicki M, Stricker D, et al. (2021) Survey and Performance Analysis of Deep Learning Based Object Detection in Challenging Environments. Sensors 21: 5116.
23. Li M, Liu Y, Liu X, Sun Q, You X, et al. (2021) The Deep Learning Compiler: A Comprehensive Survey. IEEE Transactions on Parallel and Distributed Systems 32: 708-727.
24. Achararit P, Hanif MA, Putra RVW, Shafique M, Hara-Azumi Y (2020) APNAS: Accuracy-and-Performance-Aware Neural Architecture Search for Neural Hardware Accelerators. IEEE Access 8: 165319-165334.
25. Cao Z, Hidalgo G, Simon T, Wei SE, Sheikh Y (2021) OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. IEEE Trans Pattern Anal Mach Intell 43: 172-186.
26. Zhang J, Zhang D, Yang H, Liu Y, Ren J, et al. (2023) MVPose: Realtime Multi-Person Pose Estimation Using Motion Vector on Mobile Devices. IEEE Trans Mob Comput 22: 3508-3524.
27. Al Koutayni MR, Rybalkin V, Malik J, Elhayek A, Weis C, et al. (2020) Real-Time Energy Efficient Hand Pose Estimation: A Case Study. Sensors 20: 2828.
28. Artacho B, Savakis A (2023) Full-BAPose: Bottom Up Framework for Full Body Pose Estimation. Sensors 23: 3725.
29. Jiang S, Kang P, Song X, Lo B, Shull P (2022) Emerging Wearable Interfaces and Algorithms for Hand Gesture

Recognition: A Survey. IEEE Rev Biomed Eng 15: 85-102.
30. Nogales RE, Benalcázar ME (2021) Hand gesture recognition using machine learning and infrared information: a systematic literature review. International Journal of Machine Learning and Cybernetics 12: 2859-2886.
31. Kadkhodaei H, Eftekhari Moghadam AM, Dehghan M (2021) Big data classification using heterogeneous ensemble classifiers in Apache Spark based on MapReduce paradigm. Expert Syst Appl 183: 115369.
32. Yang Y, Lv H, Chen N (2023) A Survey on ensemble learning under the era of deep learning. Artif Intell Rev 56: 5545-5589.
33. Fang Y, Lu H, Liu H (2023) Multi-modality deep forest for hand motion recognition via fusing sEMG and acceleration signals. International Journal of Machine Learning and Cybernetics 14: 1119-1131.
34. Hu WH, Rea C, Yuan QP, Erickson KG, Chen DL, et al. (2021) Real-time prediction of high-density EAST disruptions using random forest. Nuclear Fusion 61: 066034.
35. Dong X, Taylor CJ, Cootes TF (2021) A Random Forest-Based Automatic Inspection System for Aerospace Welds in X-Ray Images. IEEE Transactions on Automation Science and Engineering 18: 2128-2141.
36. Singh A, Bevilacqua A, Le Nguyen T, Hu F, McGuinness K, et al. (2023) Fast and robust video-based exercise classification via body pose tracking and scalable multivariate time series classifiers. Data Min Knowl Discov 37: 873-912.
37. Choi J-W, Park C-W, Kim J-H (2022) FMCW Radar-Based Real-Time Hand Gesture Recognition System Capable of Out-of-Distribution Detection. IEEE Access 10: 87425-87434.
38. Lu Y, Le VL, Kim TTH (2023) A 184- μ W Error-Tolerant Real-Time Hand Gesture Recognition System With Hybrid Tiny Classifiers Utilizing Edge CNN. IEEE J Solid-State Circuits 58: 530-542.
39. Zhou F, Li X, Wang Z (2020) Efficient High Cross-User Recognition Rate Ultrasonic Hand Gesture Recognition System. IEEE Sens J 20: 13501-13510.
40. Ninos A, Hasch J, Zwick T (2021) Real-Time Macro Gesture Recognition Using Efficient Empirical Feature Extraction With Millimeter-Wave Technology. IEEE Sens J 21: 15161-15170.
41. Bose SR, Kumar VS (2022) In-situ recognition of hand gesture via Enhanced Xception based single-stage deep convolutional neural network. Expert Syst Appl 193: 116427.
42. Shanthakumar VA, Peng C, Hansberger J, Cao L, Meacham S, et al. (2020) Design and evaluation of a hand gesture recognition approach for real-time interactions. Multimed Tools Appl 79: 17707-17730.
43. Venugopalan A, Reghunadhan R (2023) Applying Hybrid Deep Neural Network for the Recognition of Sign Language Words Used by the Deaf COVID-19 Patients. Arab J Sci Eng 48: 1349-1362.
44. Lu Z, Cai S, Chen B, Liu Z, Guo L, Yao L (2022) Wearable Real-Time Gesture Recognition Scheme Based on A-Mode Ultrasound. IEEE Transactions on Neural Systems and Rehabilitation Engineering 30: 2623-2629.
45. Pareek P, Thakkar A (2021) A survey on video-based Human Action Recognition: recent updates, datasets, challenges, and applications. Artif Intell Rev 54: 2259-2322.
46. Chen C, Yu Y, Sheng X, Meng J, Zhu X (2023) Real-Time Hand Gesture Recognition by Decoding Motor Unit Discharges Across Multiple Motor Tasks From Surface Electromyography. IEEE Trans Biomed Eng 70: 2058-2068.
47. Jung I-T, Ahn S, Seo J, Hong JH (2024) Exploring the Potentials of Crowdsourcing for Gesture Data Collection. Int J Hum Comput Interact 40: 3112-3121.
48. Li YK, Meng QH, Wang YX, Yang TH, Hou HR (2023) MASS: A Multisource Domain Adaptation Network for Cross-Subject Touch Gesture Recognition. IEEE Trans Industr Inform 19: 3099-3108.
49. Del Rio Guerra MS, Martin-Gutierrez J (2020) Evaluation of Full-Body Gestures Performed by Individuals with Down Syndrome: Proposal for Designing User Interfaces for All Based on Kinect Sensor. Sensors 20: 3930.