Design and Implementation of an Intelligent Virtual Personal Assistant (IVPA) for Enhanced User Interaction and Task Automation

Muharam Ali M.I. [1] , Musharrif F.M. [1], Mukunthan Tharmakulasingam[1]
Prabhashana W.H.S.C.[1]
[1]Electrical and Electronic Engineering , University of Jaffna, Kilinochchi, Sri Lanka

Abstract: This paper presents the design and implementation of an Intelligent Virtual Personal Assistant (IVPA) aimed at enhancing user interaction and automating routine tasks. The IVPA incorporates natural language processing, machine learning algorithms, and a modular architecture to understand and process sign language inputs. This feature enhances accessibility for users with hearing impairments. The system integrates with various third-party services to perform tasks such as managing schedules, controlling smart home devices, and providing personalized recommendations. Through rigorous testing, the IVPA demonstrated high accuracy in spoken and sign language command execution, resulting in increased user satisfaction. The results suggest that the proposed IVPA can significantly improve efficiency and user experience in personal and professional environments. Future work will explore expanding the IVPA's capabilities and enhancing its adaptability to diverse user needs.

Keywords: Virtual Personal Assistant, Hand Gesture, Sign Language.

Introduction

Background and Motivation

Virtual Personal Assistants (VPAs) have become integral to modern digital interactions, aiding users in performing a variety of tasks, from managing schedules to controlling smart home devices. As these technologies evolve, there is an increasing demand for VPAs that are not only intelligent but also accessible to a diverse range of users. Traditional VPAs rely heavily on voice commands, which, while convenient for many, pose accessibility challenges for individuals with hearing impairments. Furthermore, the growing complexity of user needs requires VPAs to handle more sophisticated tasks with higher accuracy and efficiency. To address these challenges, there is a need for an Intelligent Virtual Personal Assistant (IVPA) that goes beyond voice-based interactions. An IVPA that can understand and process sign language inputs would significantly enhance accessibility, making these systems more inclusive. Additionally, integrating multiple input modalities and improving task automation capabilities can further enhance user experience and interaction with VPAs.

B. Problem Statement

Despite advancements in VPA technology, most systems lack support for non-verbal communication, such as sign language. This limitation reduces accessibility. Existing VPAs primarily focus on voice and text inputs, neglecting the needs of users who rely on sign language for communication [13]. Moreover, the current generation of VPAs often struggles with complex task execution, leading to suboptimal user experiences. Therefore, there is a pressing need to develop an IVPA that can effectively process sign language inputs and provide robust task automation.

Research Objectives

The primary objective of this research is to design and implement an Intelligent Virtual Personal Assistant (IVPA) that incorporates sign language recognition, alongside natural language processing capabilities. This IVPA aims to offer a more inclusive and efficient user experience by accommodating users with hearing impairments and improving the system's overall task execution accuracy. By integrating machine learning algorithms and modular architecture, the IVPA will be capable of handling diverse input modalities and automating routine tasks seamlessly.

Contributions

This paper makes several key contributions to the field of virtual personal assistants. First, it introduces a novel approach to sign language recognition, enabling the IVPA to understand and process sign language inputs effectively. Second, the paper details the integration of this capability with existing natural language processing technologies, creating a more versatile and accessible VPA. Finally, the paper reports on the testing and evaluation of the IVPA, demonstrating its effectiveness in improving user satisfaction and accessibility.

Literature Review

Overview of Virtual Personal Assistants(VPAs)

Virtual Personal Assistants (VPAs) have emerged as a key technology in human-computer interaction, allowing users to interact with digital systems naturally [14]. These assistants, such as Apple's Siri, Amazon's Alexa, and Google Assistant, are designed to understand and respond to natural language queries, manage tasks, and provide information across various domains. The evolution of VPAs has been driven by advancements in artificial intelligence (AI), natural language processing (NLP), and machine learning (ML), enabling these systems to learn from user interactions and improve over time.

User Interface

The data collection process for a camera-based interface or a depth sensor capturing the user's hand movements involves setting up the appropriate hardware, positioning it strategically for clear visibility, and carefully considering lighting conditions. Users are instructed to perform hand gestures while recording video or images from various angles. Optionally, data undergoes pre-processing for quality improvement. For camera-based interfaces, Python with the OpenCV library is used [2], while depth sensors employ infrared technology and SDKs. Data is labeled with corresponding hand gestures to train a gesture recognition model, prioritizing user privacy and consent. Diverse datasets ensure accurate and effective gesture recognition, laying the foundation for intelligent personal virtual assistants tailored to deaf individuals, and promoting accessibility and inclusivity in technology interactions [2].

Users can connect with technology naturally and simply by using camera-based hand movement input. However, it comes with several constraints, including dependence on visibility and lighting, restricted range and field of view, latency in real-time processing, and problems in identifying complicated motions. Visual data capturing also gives rise to privacy concerns. Solutions like better lighting, more cameras with a broader field of view, better processing algorithms, machine learning for precise gesture recognition, user calibration, and privacy protections can be used to overcome these restrictions. The usefulness of hybrid input techniques that combine camera-based input with other modalities can be improved [2]. Addressing these limitations allows camera-based hand movement input to enhance user interaction in various applications, advancing human-computer interaction technologies.

Real-Time Image Captured

Setting up a camera that can take live pictures or video frames and using programming tools or libraries like OpenCV to access the camera feed is necessary for obtaining real-time photos of the user's hand movements. The system uses motion detection or hand pose estimation, among other gesture recognition algorithms, to recognize when the user makes hand movements while continuously taking frames at a constant frame rate [3]. Pre-processing of the taken photos is an optional step that might improve their quality. The recorded frames can be seen on the screen when the user makes the gestures for real-time visualization. Real-time analysis and identification of hand gestures are made possible by feeding the pre-processed photos into the gesture recognition model. The technology facilitates smooth and responsive user interactions by generating relevant responses or initiating actions based on identified gestures. As long as the program is open, a continuous loop of real-time image capture ensures continued gesture recognition and real-time answers [3]. This real-time capability is particularly valuable in applications like sign language recognition, virtual reality interactions, and gesture-based interfaces, where immediate responsiveness is essential for a smooth user experience.

While providing a natural and dynamic user experience, real-time image capture has certain drawbacks in applications such as virtual reality and gesture detection. The main drawbacks are bandwidth consumption, hardware requirements, latency, frame rate smoothness, and environmental effects. To enhance real-time image capturing, numerous strategies can be employed. Adaptive frame rate management and optimized algorithms can provide smoother images and lower latency. GPUs and other hardware acceleration can free up processing power for image processing, improving performance.

In different lighting circumstances, image quality can be improved by the use of image enhancement and noise reduction techniques [3]. Data transfer can be optimized with the use of predictive caching and compression, particularly in cloud-based applications. User modifications and speedier recognition are facilitated by contextual analysis and feedback methods. Accuracy can be increased using calibration choices based on environment and user preferences. By addressing these limitations and applying suitable solutions, real-time image capturing can deliver enhanced interactive experiences in gesture recognition, augmented reality, and virtual reality applications.

Colour-Based recognition

The first technique is called pixel-based skin detection, and it categorizes each pixel in an image as either having skin or not depending on its surrounding pixels. The second technique, known as region skin detection, involves spatially processing skin pixels based on data like intensity and texture. Several formats of color space are obtained for skin segmentation, as itemized below:

- red, green, blue (R-G-B and RGB-normalized);
- hue and saturation (H-S-V, H-S-I and H-S-L);
- luminance (YIQ, Y-Cb-Cr and YUV).

To identify the region of interest, which is typically a hand, the image is processed to transform RGB color space to another color space. This technique can be used to identify the region using a variety of colors, including red, orange, pink, and brown. To determine the likely range of skin pixels with the band values for R, G, and B pixels, the training sample of skin regions is examined [6]. The pixel color should compare the colors in the region with the predefined sample color to identify skin regions.

Appearance-Based Recognition

To model visual appearance, such as the appearance of the hand, this method relies on extracting picture features and then comparing these parameters with features retrieved from the input image frames. Where the characteristics are computed straight from the pixel intensities without first undergoing any segmentation. Due to the simplicity of extracting 2D image features, the method can be implemented more quickly than the 3D model method.

The object features can be extracted using various techniques, such as pattern or image subtraction and foreground and background segmentation algorithms, foreground extraction to segment only ROI, the Adaboost algorithm to speed up the system, and classification [6].

Classifier Ready

The captured images undergo several pre-processing steps before training the gesture recognition classifier. These steps include image cleaning, resizing, optional filtering, normalization, color space conversion, and image enhancement. Optional feature extraction techniques can also be applied to derive meaningful characteristics. Proper data labeling ensures supervised learning during classifier training, leading to improved accuracy and performance in real-time applications.

Frame extraction reduces the pixel dimension for faster computation and also levitates the process initial set of raw data is reduced into a more manageable group for processing while describing the original data set as accurately and completely. When too large input data is processed by an algorithm it is suspected to be redundant, and then it can change to a reduced set of features [5,8].

Figure 1 shows the proposed feature extraction model uses a Deep Convolutional Neural Network (DCNN) to extract feature vectors from video frames. The chosen DCNN architecture is AlexNet, which comprises five convolutional layers and three fully-connected layers. The input image size for the network is defined as 227-by-217-by-3 (width, height, and RGB channels) [5,8].

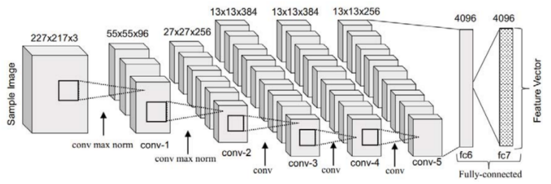


Figure 1: The Architecture of Proposed Feature Extraction Using CNN

The fc7 layer of the DCNN is utilized for feature extraction. This layer captures basic image features at the beginning of the network. As the network goes deeper, the subsequent layers process these primitive features and combine them to create higher-level image features, which are well-suited for classification tasks.

The advantage of using DCNNs, like AlexNet, lies in their ability to effectively extract potential features from diverse images, making them suitable for large-scale image processing tasks [5].

It's important to keep in mind that the field of deep learning is continually evolving, and newer techniques and architectures may have emerged since the publication of this model's paper. Staying updated with the latest research in the field is essential to leverage the most effective feature extraction methods.

Testing Against Training Model

Testing the pre-processed images against the trained gesture recognition model is a crucial step in assessing the model's performance and accuracy. After loading the pre-trained model, a separate test dataset is prepared, consisting of images that the model has not encountered during training. The model then processes each test image and generates predictions for the corresponding hand gestures. By comparing these predicted labels with the ground truth labels from the test dataset, various performance metrics such as accuracy, precision, recall, F1 score, or confusion matrix are calculated to evaluate the model's effectiveness [7]. If the model's performance falls short, fine-tuning options can be explored, such as adjusting hyperparameters or incorporating more diverse training data. Overfitting is checked to ensure that the model generalizes well to unseen data.

Real-time testing on new, unseen gestures allows the model's responsiveness and accuracy to be assessed under real-world scenarios. Iterative testing may be performed [7], refining the model until satisfactory results are achieved. By rigorously testing the model against unseen data and real-time scenarios, developers can ensure that the gesture recognition system performs optimally, enabling accurate and efficient recognition of hand gestures in applications such as sign language recognition or gesture-based human-computer interaction.

Gesture Identification

Gesture identification is a pivotal process in gesture recognition, where a trained model analyses pre-processed images of the user's hand movements to determine the specific gesture performed. The model compares extracted features with its training dataset, classifies the gesture into predefined classes, and assigns a corresponding label for generating appropriate responses or actions. This enables seamless and effective human-computer interactions in gesture-based interfaces and sign-language recognition systems [2]

The support vector machine is better because when you get a new sample (new points), you will have already made a line that keeps B and A as far away from each other as possible as shown in Figure 2, and so it is less likely that one will spill over across the line into the other's territory [2].

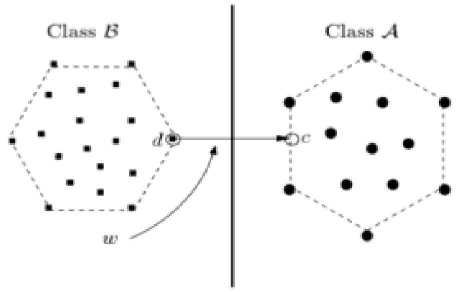


Figure 2: SVM Classification

Response

The gesture recognition system enhances user engagement and interactivity by identifying hand gestures and responding with actions, commands, or outputs. It uses algorithms to detect hand movements, image recognition using SVM models, and output conversion to text form. Feedback is provided for successful recognition and continuous gesture monitoring for effective human-computer interactions [1,12].

Wake up Interaction

The study emphasizes the value of a variety of wake-up strategies for conversational agents among young people, with a particular emphasis on avoiding the use of wake-words. While it was determined that the physical button (push-to-talk technique) was the best option for children, it's important to take into account the needs of other user groups, such as people with disabilities, such as Deaf and Hard of Hearing (DHH) users and those who interact with others through sign language [4].

Sign Language Detection Performance

Sign language detection performance analysis involves evaluating the accuracy and effectiveness of a system designed to recognize and interpret sign language gestures or movements.

Convolutional Neural Networks (CNNs), track hand gestures each video frames are processed, and a bounding box helps in tracking those hands. The bounded values of each pixel are passed through the layers of CNNs and help in predicting the exact match in identifying the gestures [2].

Convolutional Neural Network is applied to analyze visual imagery. Our framework gives 96.9% precision in gesture identification which is 25.78% higher compared to the past strategy [2].

Gesture Classification Using Support Vector Machine (SVM), used non-linear MCSVM for the classification of every alphabetic sign in the last section of this proposed model. SVM is a mostly used learning method for the classification of extracted features and regression. As SVM is a binary classifier based on the supervised learning approach for classifying data into two classes by drawing a hyperplane [8].

Summary of Literature Gaps

Despite significant progress in the development of VPAs, several gaps remain in the literature. Most notably, the integration of sign language recognition into VPAs is still in the early stages, with limited research addressing the challenges associated with real-time Sign Language Recognition (SLR) in diverse environments.

Additionally, while modular architectures and machine learning techniques have been widely adopted in the development of VPAs, there is a need for more research focused on improving the accessibility and inclusivity of these systems.

This research aims to address these gaps by developing an Intelligent Virtual Personal Assistant (IVPA) that incorporates sign language recognition, leverages machine learning for enhanced user interaction, and utilizes a modular architecture to support a wide range of input modalities. By focusing on accessibility and inclusivity, this research seeks to advance the state-of-the-art in VPAs and contribute to the development of more equitable and user-friendly technologies.

Methodology

Defining Objectives

Before starting work, the goals and objectives should be defined. So, we have come up with a well-defined objective.

The research project aims to create an inclusive Virtual Personal Assistant (VPA) system that enhances communication for disabled individuals using sign language. Utilizing machine learning and natural language processing technologies, the VPA can accurately interpret and respond to inputs, bridging the communication gap and providing a seamless interaction platform. The goal is to create an interface that accommodates various sign language dialects, enabling users to navigate tasks, access information, and interact with technology, fostering greater independence.

Choosing Right Technologies

For the development, the frameworks, programming languages, and libraries, as well as potential integration with other systems should be identified.

Our project integrates advanced technologies to create a system that recognizes and interprets hand gestures, specifically tailored to facilitate communication for individuals who rely on sign language or gestures as their primary mode of interaction. Using OpenCV for video capture, CVZone for hand detection and preprocessing, and Google's Teachable Machine for gesture recognition, our system aims to bridge communication barriers. This technology not only empowers the deaf and hard-of-hearing communities by enabling intuitive interaction with digital platforms but also finds application across various domains, promising a more inclusive and seamless future for human-computer interaction. The following Figure (3) illustrates the key functionalities and workflow of our system in achieving these goals.

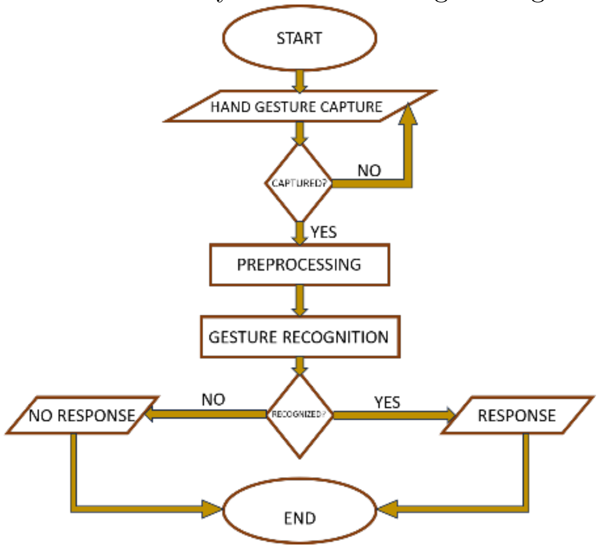


Figure 3: SVM Classification

Hand Gesture Capture

- OpenCV captures the live feed to the webcam
- From the CVzone platform, processes the input for hand detection

From CVzone, we used HandTrackingModule to detect hand. It is done with the help of the MediaPipe platform. MediaPipe works on the complete input image and provides a cropped image of the hand. MediaPipe finds the 21 hand landmarks on the cropped image of the hand. The hand landmarker model bundle contains a palm detection model and a hand landmarks detection model.

The palm detection model localizes the region of hands from the whole input image, and the hand landmarks detection model finds the landmarks on the cropped hand image defined by the palm detection model.

Preprocessing

- Image processing techniques clean and isolate hand gestures. Also, the image is set to a white background for the clear recognition.

Resizing the input images to a consistent resolution helps standardize the data and facilitates computational efficiency. In our case, we resize the image to a value of 300 maximum. Converting the image to a different color space, such as grayscale, can enhance the visibility of relevant features for hand detection. Applying filters, such as Gaussian filters and median filters, helps in reducing noise and smoothing the image. It is done in the Teachable Machine for classification.

Gesture Recognition

- Data fed into Google's Teachable Machine for training. This is a free and easy way of training the classification models.
- Trained model interprets and recognizes hand gestures with the help of CVzone's classifier modules.

Google Teachable Machine utilizes transfer learning and the MobileNet architecture for image classification tasks. Transfer learning is a machine learning technique that leverages knowledge gained from solving one problem and applies it to a different but related problem. In the context of image classification, the Teachable Machine uses a pre-trained MobileNet model as the backbone. MobileNet is a lightweight deep neural network architecture optimized for mobile and edge devices. The model has been pre-trained on a large dataset, typically for a general image classification task. During the transfer learning process, the Teachable Machine takes advantage of the knowledge encoded in the MobileNet model and fine-tunes it on the specific classes or labels relevant to the custom image classification task. The Classifier class from the cvzone.Classification Module is used for image classification. It loads a pre-trained model from the Teachable Machine and associated labels. The getPrediction method is used to obtain predictions for the input image, and the result is a class label. It uses forward-pass inference to recognize of correct output. Figure 4 shows the successfully detected gesture for the letter A. The recognized letter is assigned to do the task as pre-described. The selection of OpenCV, CVZone, and Google's Teachable Machine for this project stems from their unique and complementary capabilities. OpenCV, renowned for its robust computer vision functionalities, serves as the foundation for capturing real-time video feeds and initiating image processing tasks crucial for hand gesture recognition.

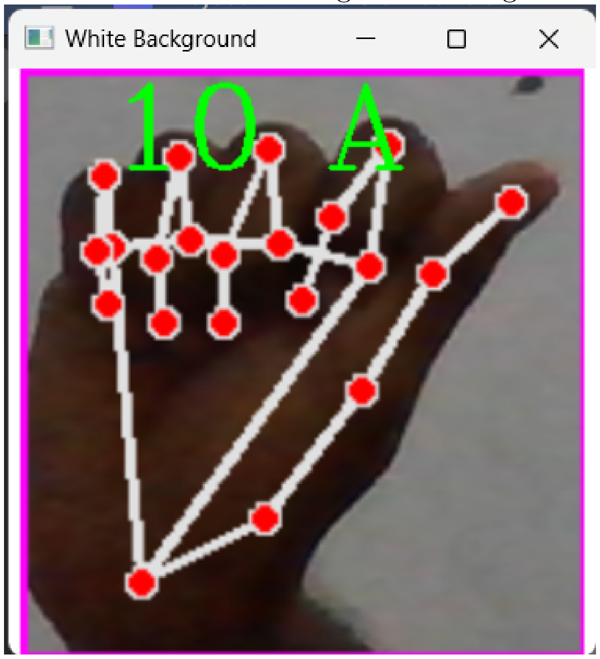


Figure 4: Detected Letter A

CVZone supplements this process with its specialized tools, facilitating precise hand detection and preprocessing, ensuring the accurate extraction of gestures from complex backgrounds. Google's Teachable Machine, a powerful machine learning platform, was chosen for its user-friendly interface and adeptness in training models for gesture recognition, providing the intelligence to interpret and respond to these gestures accurately. The synergy between these technologies forms the backbone of our system, combining their strengths to create an accessible and inclusive platform for intuitive communication through hand gestures.

Google's Teachable Machine leverages transfer learning with MobileNet as the underlying model architecture for training purposes. MobileNet is a lightweight convolutional neural network (CNN) architecture designed for mobile and embedded vision applications. It is known for its efficiency in terms of computational requirements and memory footprint while maintaining reasonably good performance in image classification tasks.

Data Collection and Annotation

For a successful Sign Language Recognition System, we have to choose a convenient sign language. For this purpose, we have chosen the American Sign Language (ASL) Alphabet as our data.

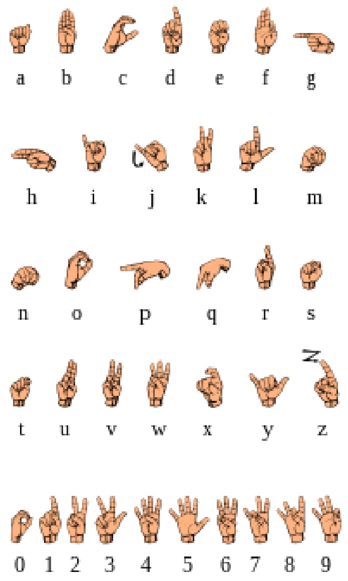


Figure 5: ASL Alphabet [15]

Figure 5 shows every sign of ASL alphabets. ASL boasts a rich linguistic structure and a robust vocabulary, making it an ideal choice for inclusivity in communication. Integrating ASL into a virtual personal assistant system aims to bridge communication gaps and ensure accessibility for individuals who rely on sign language as their primary mode of communication, fostering a more inclusive and user-friendly experience for all. The data is collected through the webcam with different signers.

Design and Train the Model

For the development of our sign language recognition system, substantial strides have been made in the 'Design and Train the Model' phase. The project involves the integration of powerful libraries, including OpenCV for hand gesture capture, CVZone for efficient processing, and Google's Teachable Machine for model training. The decision to use American Sign Language (ASL) aligns with our goal of fostering inclusivity and accessibility. The model training process is conducted through transfer learning on Google's Teachable Machine, which utilizes the MobileNet architecture. This approach enables the model to leverage pre-trained knowledge and adapt efficiently to recognize a diverse range of ASL gestures.

The training dataset has been meticulously curated to encompass various signs, ensuring the model's proficiency across a broad spectrum of hand movements. Additionally, the incorporation of image preprocessing techniques enhances the dataset's quality and aids in training robust models. The next steps involve fine-tuning the model parameters, validating its performance on separate datasets, and optimizing it for seamless deployment on the chosen Raspberry Pi 4 platform. As we progress, continuous iteration and refinement based on training results and user feedback will be pivotal in achieving a highly accurate and user-friendly sign language recognition system.

Task Creation

We have created some tasks that can be included in our systems. For this purpose, we have used Some APIs to integrate with different platforms. There are some more applications to be added in the future. The created tasks are listed below:

Weather

The application retrieves weather information based on the user's location using the OpenWeatherMap API and the 'geocoder' library. The obtained weather details are then displayed in a Tkinter window and read aloud using the 'pyttsx3' library. The weather information includes the location, weather description, temperature, and feels-like temperature. The application utilizes threading to enable concurrent execution of the text-to-speech functionality, ensuring a responsive user experience. The integration of APIs, text-to-speech, and graphical user interface components makes this application a concise tool for providing real-time weather updates with spoken feedback.

Music

The music application utilizes Tkinter and Pygame to create a basic music player. The Tkinter Graphical User Interface (GUI) includes a list of songs, play, pause, next, and previous buttons with corresponding functionalities. The script initializes the pygame mixer for audio playback and defines functions for music control. Users can load MP3 files from a specified folder, play, pause, and navigate through the playlist. The integration of Tkinter and Pygame provides a straightforward graphical interface for interacting with the music player, making it a user-friendly tool for playing and managing audio tracks.

ChatGPT

The chat application uses Tkinter and integrates it with OpenAI's GPT-3.5 Turbo model. The application allows users to input text, submit it to the GPT-3.5 Turbo model for generating responses, and display both the text and the spoken response in a Tkinter graphical user interface. The application includes the necessary components for handling user input, interacting with the OpenAI API, displaying responses, and incorporating speech synthesis for an interactive and user-friendly experience.

RESULTS

The recognition model identifies hand gestures with good accuracy for a limited set of gestures. The system can identify the gestures and give the exact letter in the window as shown in figure 6 where the model recognizes the letter B. The accuracy of each letter is higher than 80% thus we can succeed in the program with a good accuracy. We tested about 40 inputs for each sign and we got 34, 34, and 36 correct recognized inputs for letters A, B, and C respectively. Table 2 illustrates the results of accuracy. We can able to execute every task for the first time since the model identifies the letters accurately.

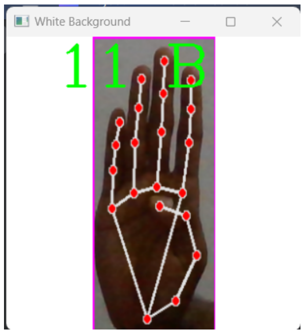
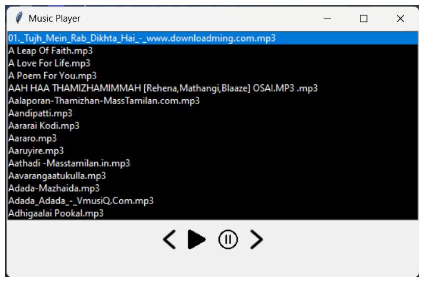


Figure 6: Recognized Letter B

With the recognized letters the system can do the tasks which was assigned. The system will give the outputs in text format and also the system will read the text outputs. Figure 7 is the task Music that the system executes for the recognized letter B.



```
01._Tujh_Mein_Rab_Dikhta_Hai_-_www.downloadming.com.mp3
A Leap Of Faith.mp3
A Love For Life.mp3
A Poem For You.mp3
AAH HAA THAMIZHAMIMMAH [Rehena,Mathangi,Blaaze] OSAI.MP3 .mp3
Aalaporan-Thamizhan-MassTamilan.com.mp3
Aandipatti.mp3
Aararai Kodi.mp3
Aararo.mp3
Aaruyire.mp3
Aathadi -Masstamilan.in.mp3
Aavarangaatukulla.mp3
Adada-Mazhaida.mp3
Adada_Adada_-_VmusiQ.Com.mp3
Adhigaalai Pookal.mp3
```

Figure 7: Recognized Letter B

Table 1 illustrates the starting time of the tasks in a Raspberry Pi 4 and an i7 Laptop. As a result, the time taken to open a specific application on Raspberry Pi 4 is comparatively greater than on a laptop. This is because the laptop's VGA, RAM size and Processor are much better than the Raspberry Pi's specifications.

For the comparison we have chosen a laptop with a processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, RAM: 8GB DDR4, VGA: NVIDIA GeForce MX230, and OS: Windows 11 Home Single Language. The following tables show the result.

Table 1: Starting time on different platforms

Tasks	Laptop with 8Gb RAM and 17 Processor	Raspberry Pi4 Model B 4Gb Ram
Weather	1.555 sec	2.285 sec
Music	1.079 sec	1.627 sec
ChatGpt	0.082 sec	0.218 sec

Table 2: Accuracy of used letters

Letter	Accuracy
A	85%
B	85%
C	90%

In terms of latency, the integration of powerful libraries like OpenCV, TensorFlow, Pygame, and others in our system brings a diverse set of capabilities but can contribute to a slightly higher latency in system response times. The complex functionalities and sophisticated algorithms inherent in these libraries demand computational resources, potentially impacting the speed of gesture recognition, interpretation, and subsequent system responses. Factors such as image processing, model inference, and real-time interaction necessitate considerable processing time, which, when combined, may result in a slightly elevated latency in the system's responsiveness. While these libraries empower the system with robust functionalities, their computational demands imply that optimizing and fine-tuning various processes will be crucial to mitigate latency and deliver a smoother, more instantaneous user experience.

Some delay measurements are listed below.

a Initialization of the classifier: about 6 seconds.
This happens during the initialization of the Classifier object with the provided model and labels.

b Hand Detection Time: The HandDetector takes an average of 2 seconds to find hands in each video frame. This time may vary based on the complexity of the image and the efficiency of the hand-detection algorithm.

c) Classification Time: The time taken to classify hand gestures using the Classifier is approximately 1 second per step of prediction. This duration is obtained from TensorFlow's output during the getPrediction method.

d Overall Performance: It takes about 15 to 17 seconds to start to execute a task. This includes check the included libraries, camera initialization time, and above-mentioned times.

Until now, our system has some advanced features and capabilities that make our system unique.

- Hand Gesture Recognition: The system utilizes computer vision techniques to detect and recognize hand gestures captured by a webcam in real time.
- Gesture Classification: The detected hand gestures are classified into predefined categories using a machine learning model (neural network) loaded with the Classifier class.
- Task Execution: Upon recognizing a specific hand gesture, the system triggers corresponding tasks. For example, displaying weather information, playing songs, or interacting with a chatbot.
- Weather Display: The system can display weather information, indicating integration with external weather-related functionality.

- Music Playback: The system can play songs. This functionality involves accessing a directory of music files and playing them.
- Chatbot Interaction: There is a feature for interacting with a chatbot (GPT-3-based) based on hand gestures. Users can potentially communicate or ask questions.
- Real-time Video Processing: The system captures and processes video frames from a webcam in real-time, indicating a live interaction scenario.
- User Interface: The system provides a user interface by displaying processed images and relevant information for the outputs.

Figure 8 shows the working flow of our system. First, the Camera will capture the input with the help of OpenCV and the captured image will be preprocessed and identify the hand region. Next, the gesture will be recognized by the model and it will execute the desired task. As a result, the response will be given for the recognized hand gesture.

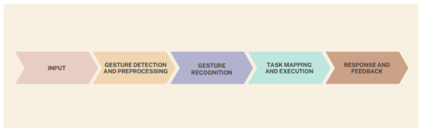

INPUT
GESTURE DETECTION AND PREPROCESSING
GESTURE RECOGNITION
TASK MAPPING AND EXECUTION
RESPONSE AND FEEDBACK

Figure 8: Recognized Letter B

Conclusion

A system capable of recognizing and understanding hand gestures holds immense importance in fostering accessibility and inclusivity. For individuals who communicate primarily through sign language or gestures, this technology enables seamless interaction with digital devices. It enhances accessibility for the deaf and hard-of-hearing communities, allowing them to navigate, communicate, and interact with technology more intuitively. Moreover, in various fields such as human-computer interaction, healthcare, and even education, this system can facilitate more natural and efficient communication, breaking barriers and creating a more inclusive environment for all users.

We have created a Virtual Personal Assistant system with limited ASL inputs and tasks like ChatGPT, music, and weather. While the current system demonstrates a strong capability to execute tasks based on input gestures, the accuracy of gesture recognition remains a challenge, particularly when compared to the advanced systems discussed in the literature. We acknowledge that the system's accuracy can be significantly enhanced by optimizing the capture of gesture stills against a clear background and employing a more powerful and sophisticated training platform.

Despite these challenges, our system has proven its potential as a valuable tool for individuals with special disabilities. The successful execution of tasks demonstrates the system's practical utility, and it sets the stage for future developments. In subsequent iterations, we aim to expand the system's capabilities by incorporating additional ASL gestures, improving recognition accuracy, and integrating more complex tasks. We believe that with continued refinement, this project will evolve into a robust and reliable product, offering enhanced independence and inclusivity for individuals with disabilities, and contributing to a more accessible digital world for all.

Future Work

Looking ahead, we plan to enhance the system by incorporating additional tasks such as making calls, sending messages, and setting reminders. These features will further broaden the utility of the Virtual Personal Assistant, making it even more versatile and helpful for users. A critical focus for future work will be improving the accuracy of letter detection, which is essential for reliable communication. Additionally, we recognize the need to address the high latency currently observed on the Raspberry Pi 4, as reducing this delay is crucial for a smoother user experience. Our ultimate goal is to refine and expand the system, transforming it into a comprehensive and fully functional solution that can significantly benefit individuals with special needs.

References

[1] V. Kepuska and G. Bohouta, "Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa, and Google Home)," in 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2018, pp. 99-103.

[2] Poornima, N. and Murugan, M. "Improved gesture precision virtual personal assistant (IGP-VPA) system for speech impaired people". I-manager's J Pattern Recogn, 2019, p.17

[3] D. Someshwar, D. Bhanushali, V. Chaudhari and S. Nadkarni, "Implementation of Virtual Assistant with Sign Language using Deep Learning and TensorFlow," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 595-600, doi: 10.1109/ICIRCA48905.2020.9183179.

[4] Glasser, A., Mande, V. and Huenerfauth, M. "Accessibility for deaf and hard of hearing users: Sign language conversational user interfaces". In Proceedings of the 2nd Conference on Conversational User Interfaces, 2020, pp. 1-3.

5 Obi, Y., Claudio, K.S., Budiman, V.M., Achmad, S. and Kurniawan, A. "Sign language recognition system for communicating to people with disabilities". Procedia Computer Science, 2023, pp.13-20.

[5] Ismail, M.H., Dawwd, S.A. and Ali, F.H. "A Review on Arabic Sign Language Recognition." Journal of Advances in Computer and Electronics Engineering, 2021, 6(12), pp.112

[6] Roy, S. and Das, B. "Understanding the Motion Adaption of Machine Using Long Short-Term Memory Networks for Voiceless Virtual Assistant". International Journal of Digital Technologies, 2023, 2(1).

[7] M. R. Islam, U. K. Mitu, R. A. Bhuiyan and J. Shin, ""Hand Gesture Feature Extraction Using Deep Convolutional Neural Network for Recognizing American Sign Language, 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), Poitiers,

France, 2018, pp. 115-119, doi: 10.1109/ICFSP.2018.8552044.

[8] Imrie P, and Bednar P. (2013). "Virtual Personal Assistant", in: Martinez M. and Pennarolaecilia F. (editors). Proceedings of 10th Conference of the Italian Chapter of AIS, 'Empowering society through digital innovations', UniversitÃă Commerciale Luigi Bocconi in Milan, Italy, December 14th, 2013. ISBN: 978-88-6685-007-6

[9] R. A. Bhuiyan, A. K. Tushar, A. Ashiquzzaman, J. Shin and M. R. Islam, "Reduction of gesture feature dimension for improving the hand gesture recognition performance of numerical sign language,". 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2017, pp. 1-6, doi: 10.1109/ICCITECHN.2017.8281833

[10] K. N., R. V., S. S. S. and D. R., "Intelligent Personal Assistant - Implementing Voice Commands enabling Speech Recognition," 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 2020, pp. 1-5, doi: 10.1109/ICSCAN49426.2020.9262279.

[11] V. Kepuska and G. Bohouta, "Improving Wake-Up-Word and General Speech Recognition Systems", 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, 2017, pp. 318-321, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.67.

[12] D.Someshwar, D.Bhanushali, V. Chaudhari and S. Nadkarni, "Implementation of Virtual Assistant with Sign Language using Deep Learning and TensorFlow,"" 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 595-600, doi: 10.1109/ICIRCA48905.2020.9183179.

[13] Dr. D. S. Hirolikar 'Virtual assistant with sign language', International Journal of Scientific Research in Science, Engineering and Technology, 2023, pp. 297-301. doi:10.32628/ijsrset23103107.

[14] Wikipedia. (2024, May. 18). American Manual Alphabet [Online]. Available: https://en.wikipedia.org/wiki/American_manual_alphabet