

**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

**Developing a User Manageable Virtual Assistance for PC Using Natural Language Processing and CNN Algorithm.**

**A CAPSTONE PROJECT REPORT**

*Submitted in the partial fulfilment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**INFORMATION TECHNOLOGY**

**Submitted by**

**S. CHENCHU DANUSH (192211615)**

**N. SUMA TEJA (192211193)**

**Under the Supervision of**

**Dr. A. Jaya Mabel Rani**

**JUNE 2024**

## DECLARATION:

We are S. Chenchu Danush, P. Chandu students of Bachelor of Engineering in Information Technology, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled “**Developing a User Manageable Virtual Assistance for PC Using Natural Language Processing and CNN Algorithm**” is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering.

**S. CHENCHU DANUSH (192211615)**

**N. SUMA TEJA (192211193)**

Date:

Place:

# **CERTIFICATE**

This is to certify that the project entitled **“Developing a User Manageable Virtual Assistance for PC Using Natural Language Processing and CNN Algorithm”** submitted by S. Chenchu Danush, N. Suma Teja has been carried out under our supervision. The project has been submitted as per the requirements in the current semester of B. Tech Information Technology.

Faculty Incharge

Dr. A. Jaya Mabel Rani

Slot: C

Course Code: CSA1748

Course Name: Artificial Intelligence for Neural Network Applications.

**ABSTRACT:**

The "JARVIS" project introduces an advanced virtual assistant designed to enhance user interactions on personal computers through the seamless integration of Natural Language Processing (NLP) and voice-controlled systems. Leveraging the capabilities of Convolutional Neural Networks (CNNs), this assistant processes user commands, translating them into executable actions ranging from system management, media control, and web browsing to scheduling tasks and accessing applications. With support for Text-to-Speech (TTS) and voice input via Microsoft’s SDK, JARVIS provides an intuitive, hands-free interface that redefines productivity and personal computing. The project's innovative approach aims to elevate user experience by not only enabling intelligent task automation but also paving the way for future enhancements such as object recognition through visual inputs. Designed for personalized interaction, JARVIS aspires to deliver a smarter, more efficient way to manage everyday PC tasks, promoting a truly interactive and adaptive digital assistant.

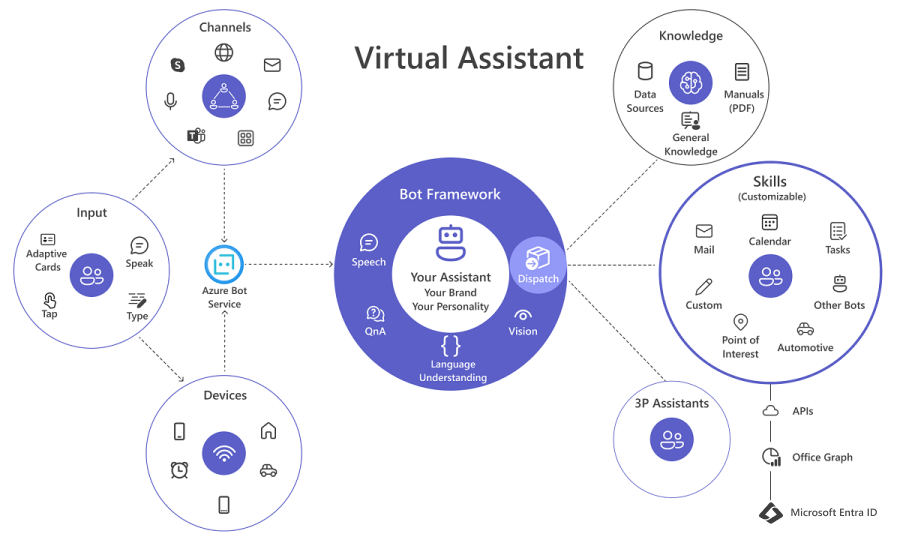
**INTRODUCTION:**

In today’s fast-paced, technology-driven world, efficiency and convenience are paramount. The **JARVIS** project introduces a revolutionary virtual assistant designed to transform how users interact with personal computers. Developed in **C++**, JARVIS leverages the power of **Natural Language Processing (NLP)** and **Convolutional Neural Networks (CNNs)** to seamlessly bridge the gap between human language and machine execution, allowing users to control their devices through intuitive voice or text commands. From launching applications, playing media, and browsing the web to organizing daily schedules, JARVIS provides an intelligent, hands-free solution that enhances productivity and streamlines everyday tasks into effortless experiences.

JARVIS is powered by a range of **C++ libraries** and **APIs** to deliver its advanced functionalities. For voice interaction, **Microsoft’s Speech SDK** is utilized to enable real-time **speech-to-text** and **text-to-speech (TTS)** capabilities. Additionally, **OpenCV** provides a robust foundation for **image processing** and **object detection**, while **TensorFlow C++ API** or **dlib** can be employed for **CNN-based machine learning tasks**. **Boost.Asio** is used for networking and asynchronous I/O operations, facilitating communication with other services. For text-based NLP, **cppjieba** or other C++ NLP libraries can be integrated, ensuring that JARVIS can interpret and process natural language effectively.

What truly sets JARVIS apart from existing virtual assistants is its bidirectional communication capabilities. The combination of **C++** with **Speech SDK** ensures that JARVIS can not only understand but also respond to natural human language in real time. This two-way interaction offers users a personalized digital companion that listens, processes, and executes commands, ensuring a seamless flow between user intent and system action. Built to evolve, JARVIS has the potential to go beyond traditional virtual assistant tasks, incorporating advanced visual recognition and further integration with external APIs, making it an indispensable tool for both casual users and professionals.

With a strong emphasis on adaptability and continuous learning, JARVIS is designed to grow alongside its user base, offering a tailored experience that responds to individual needs. By combining cutting-edge **AI technologies** with practical everyday applications, JARVIS is not just a virtual assistant but a gateway to the future of human-computer interaction. It aims to revolutionize how we work, play, and manage our digital lives—ushering us into a truly intelligent computing environment.



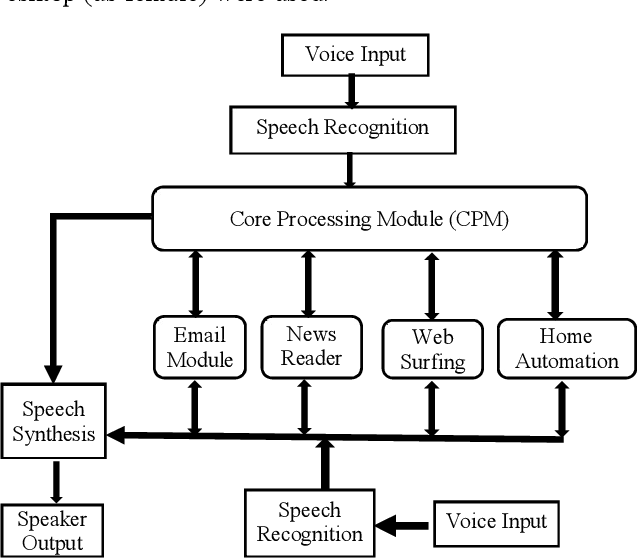
**PROBLEM STATEMENT:**

In today's digital age, users often face inefficiencies and disruptions while interacting with personal computers due to the reliance on traditional input methods such as keyboards and mice. This not only limits multitasking capabilities but also creates accessibility barriers for users with physical limitations. Existing virtual assistants lack the seamless integration and comprehensive functionality required for hands-free control of a wide range of tasks on PCs and laptops, such as managing files, executing commands, and automating routine operations. Additionally, the absence of real-time voice communication and personalized interaction further hinders the potential for an immersive, user-friendly experience.

The challenge lies in developing a sophisticated virtual assistant that can understand and execute complex user commands using **Natural Language Processing (NLP)**, respond with voice interaction via **Text-to-Speech (TTS)**, and accept voice inputs for real-time execution. Furthermore, integrating **Convolutional Neural Networks (CNNs)** to expand future capabilities such as visual recognition and object detection is necessary to push the boundaries of what virtual assistants can offer. The goal is to create a solution that not only enhances productivity but also provides an intelligent, intuitive, and fully interactive experience, redefining human-computer interaction for everyday users.

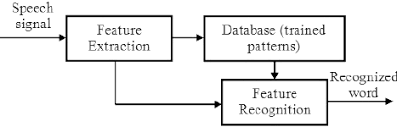
**PROPOSED DESIGN WORK:**

The proposed design for the JARVIS virtual assistant focuses on creating an intelligent, voice-activated system that enhances the accessibility, efficiency, and usability of personal computers. JARVIS will enable users to interact with their devices using simple voice commands, with capabilities to perform basic operations, assist users with disabilities, and automate repetitive tasks. Below is an analysis of the key components and design work required for the system:



**1. Speech Recognition Module**

**Purpose:** The Speech Recognition module will convert voice input into text commands. This is essential for JARVIS to understand user requests.



**Design Considerations:**

* **Microphone Input:** The system will use a microphone to capture audio and convert it into an electrical signal.
* **Hidden Markov Model (HMM):** JARVIS will employ HMM-based speech recognition, where speech is split into small fragments and processed to extract features using **cepstral coefficients**.
* **Training:** The system will be trained to recognize different accents and speech patterns, handling variations in pronunciation for more accurate results.
* **Speech-to-Text Library:** A C++ library such as **PocketSphinx** or **CMU Sphinx** can be used for speech recognition. These libraries will recognize phonemes and map them to corresponding text.

**Key Libraries:**

* **Microsoft Speech SDK (C++):** Used to capture voice input and convert it into text.
* **PocketSphinx/CMU Sphinx:** For speech recognition based on HMM.

**2. Natural Language Processing (NLP) Module**

**Purpose:** Once the voice input is converted into text, the NLP module will process the text to interpret the user's intent and convert the command into executable actions.

**Design Considerations:**

* **Text Parsing and Intent Recognition:** Use NLP techniques to understand user commands. For example, recognizing phrases such as "open browser," "send an email," or "play music."
* **Regex for Text Processing:** Regular expressions will be used for parsing URLs, extracting YouTube links, and other specific patterns in text commands.
* **NLP Library:** Use libraries such as **cppjieba** (for Chinese users) or a custom solution for parsing, understanding commands, and interacting with external APIs.

**Key Libraries:**

* **cppjieba (C++):** For natural language processing.
* **Boost.Regex (C++):** For regular expression handling.

**3. Task Execution Module**

**Purpose:** Once the command is understood, JARVIS will execute the desired task, whether it is opening applications, browsing websites, or sending emails.

**Design Considerations:**

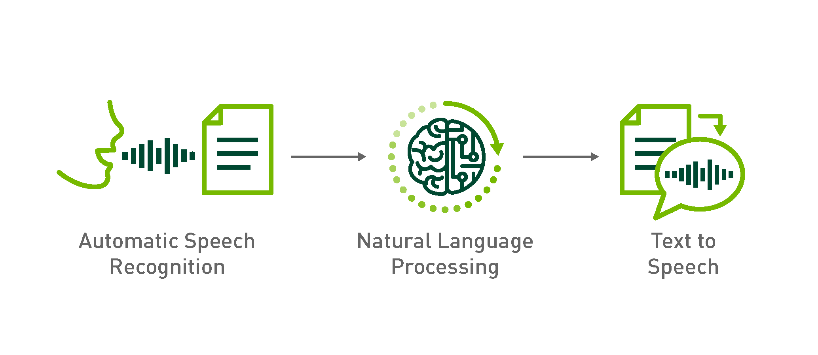
* **System-Level Access:** JARVIS needs to access operating system functions like opening or closing applications, sending emails, and manipulating files.
* **Asynchronous Task Execution:** Use libraries like **Boost.Asio** for asynchronous input/output operations, allowing multiple tasks to be executed in parallel without performance issues.
* **System Commands:** C++ code will interface with the operating system to execute commands like opening files or applications, navigating the file system, and interacting with other programs.

**Key Libraries:**

* **Boost.Asio (C++):** For managing asynchronous system commands and networking tasks.
* **ShellExecute (Windows) or fork/exec (Linux):** For opening/closing applications.

**4. Text-to-Speech (TTS) Module**

**Purpose:** The Text-to-Speech module will give JARVIS the ability to respond to users in a human-like voice, providing auditory feedback on task completion.



**Design Considerations:**

* **TTS Engine:** Integration of a TTS engine that converts text-based responses into voice. Microsoft Speech SDK provides built-in support for TTS.
* **Speech Customization:** Users should be able to customize the voice pitch, speed, and tone to create a personalized experience.

**Key Libraries:**

* **Microsoft Speech SDK (C++):** Provides TTS capabilities, converting text responses into voice.

**5. Internet Connectivity and API Integration**

**Purpose:** Some user commands will require JARVIS to interact with external APIs (e.g., finding flight information, checking the weather, or looking up YouTube videos).

**Design Considerations:**

* **URL Parsing:** Use Urllib.parse for URL manipulation and to retrieve information from APIs like **Wolfram Alpha**, **Google APIs**, or **YouTube API**.
* **Query Management:** The system will handle queries such as "find nearby airports" or "show weather," using API requests to fetch relevant data.

**Key Libraries:**

* **Urllib.parse (C++ equivalent):** Use C++ libraries or custom parsers to manage URLs and API queries.
* **cURL (C++):** For making HTTP requests and integrating with external services.

**6. Visual Recognition and Object Detection Module (Future Work)**

**Purpose:** Incorporating computer vision to allow JARVIS to process visual input (e.g., detect objects via a webcam).

**Design Considerations:**

* **Object Detection:** Use **OpenCV** and CNN-based models for real-time object recognition. This would allow JARVIS to visually detect objects and assist users in tasks like identifying items in a photo.

**Key Libraries:**

* **OpenCV (C++):** For image processing and object detection.
* **TensorFlow C++ API/dlib:** For implementing CNN-based object detection.

**7. Accessibility Features**

**Purpose:** To cater to users with disabilities, JARVIS will include special accessibility features.

**Design Considerations:**

* **Blind Users:** The Speech Recognition system is tailored for users with sight impairments, allowing them to control their computer entirely by voice.
* **Speech Synthesizer:** Provides audible responses to keep visually impaired users informed about system actions.

**Key Libraries:**

* **Speech SDK (C++):** For speech-to-text and text-to-speech capabilities.

**Methodology:**

1. **Voice Input:** User speaks into the microphone.
2. **Speech Recognition:** Audio is converted into text using speech recognition algorithms.
3. **NLP Processing:** Text is parsed, and user intent is extracted.
4. **Task Execution:** JARVIS executes the command (e.g., opens an application, searches the web).
5. **Voice Feedback:** TTS converts system feedback into a voice response for the user.
6. **API Interaction:** For commands that require internet access, JARVIS communicates with external APIs to retrieve relevant data.

**1. Speech Recognition Algorithms**

Speech recognition is a crucial part of the system that allows users to interact with applications using voice commands. Several algorithms are involved in this process:

**a. Hidden Markov Model (HMM)**

**Description:** HMM is a statistical model that assumes the system being modeled is a Markov process with hidden states. It is widely used in speech recognition due to its ability to model temporal patterns.

**Usage in Speech Recognition**: In this system, the speech signal is broken down into small time segments (around 10 ms), and each segment is analyzed to extract features like cepstral coefficients, which represent the signal’s frequency spectrum.

**Phoneme Detection:** The extracted features are matched to phonemes (basic units of sound), and the HMM model is used to predict the most probable sequence of words that the speech represents.

**Training Requirement**: HMMs require a large dataset of speech samples to train and can handle the variations in pronunciation and accents of different users.

**b. Viterbi Algorithm**

**Description:** The Viterbi algorithm is used in conjunction with HMMs to decode the most likely sequence of hidden states (in this case, phonemes or words) from the observed data (speech signal).

**Usage in Speech Recognition:** The Viterbi algorithm is used to find the most probable sequence of words from the possible phoneme combinations based on the speech signal.

**2. Pattern Matching with Regular Expressions**

Pattern matching is used to search and manipulate strings of text, particularly in recognizing patterns such as URLs or keywords in commands.

**a. Regular Expression Algorithm**

**Description:** A regular expression (regex) is a sequence of characters that defines a search pattern. It allows for pattern matching, searching, and replacing text based on specific rules.

**Usage:** In this system, regular expressions are used to extract YouTube links, email addresses, or other data patterns from user queries.

**3. URL Parsing**

The system extracts and manipulates URLs for various tasks, like retrieving data from external APIs or generating URLs based on user queries.

**a. Parsing Algorithms (Urllib.parse)**

**Description:** URL parsing algorithms break down a URL string into its components such as the protocol (HTTP), domain, path, and query parameters. These components can then be manipulated for tasks like fetching data from APIs or generating URLs.

**Key Functions:**

urlparse(): Breaks a URL into its components.

parse\_qs(): Parses the query string of a URL into key-value pairs.

urlencode(): Converts data into a URL-encoded format for sending as part of an HTTP request.

**4. Neural Networks for Translation**

Translation algorithms are based on deep learning techniques, particularly neural networks that map words from one language to another.

**a. Encoder-Decoder Neural Networks**

**Description:** Neural networks for translation use an encoder-decoder architecture. The encoder processes the input sentence in the source language and transforms it into a fixed-size vector (encoding). The decoder then takes this vector and generates the output sentence in the target language.

**Key Components:**

**Encoder:** Encodes the sequence of input words into a numerical array that represents its meaning.

**Decoder:** Decodes the numerical array back into a sequence of words in the target language.

**Training:** The network is trained on large datasets of parallel sentences (same sentence in both languages) to learn how to translate between them.

**Recurrent Neural Networks (RNN):** Traditionally, RNNs or their variants like Long Short-Term Memory (LSTM) are used to model the sequential nature of language.

**Attention Mechanism:** Modern translation models often include attention mechanisms, which allow the decoder to focus on specific parts of the input sequence during translation, improving translation accuracy.

**5. Additional Algorithms for Speech Synthesis**

Speech synthesis is used to provide a response to users, creating the impression that the user is interacting with a real assistant.

**a. Text-to-Speech (TTS) Algorithms**

**Description:** TTS systems convert text into spoken language using deep learning models or rule-based algorithms.

**Usage:** The TTS system takes a string (e.g., "Your flight is booked") and converts it into an audio waveform that is played back to the user.

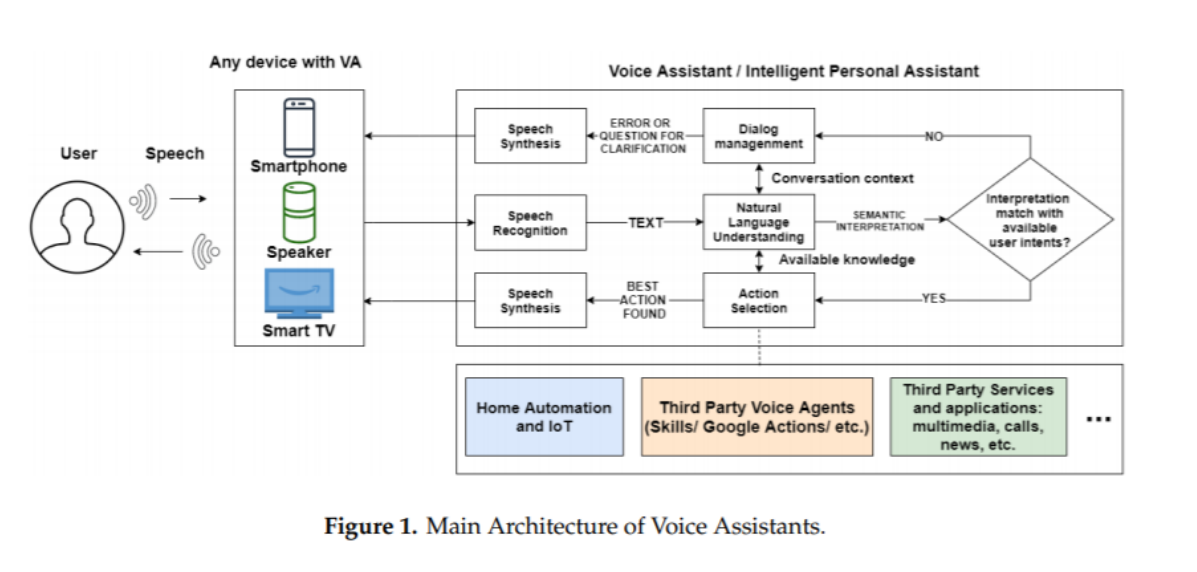
**b. Concatenative TTS:**

**Approach:** This method strings together pre-recorded units of speech (such as phonemes or words) to create new sentences.

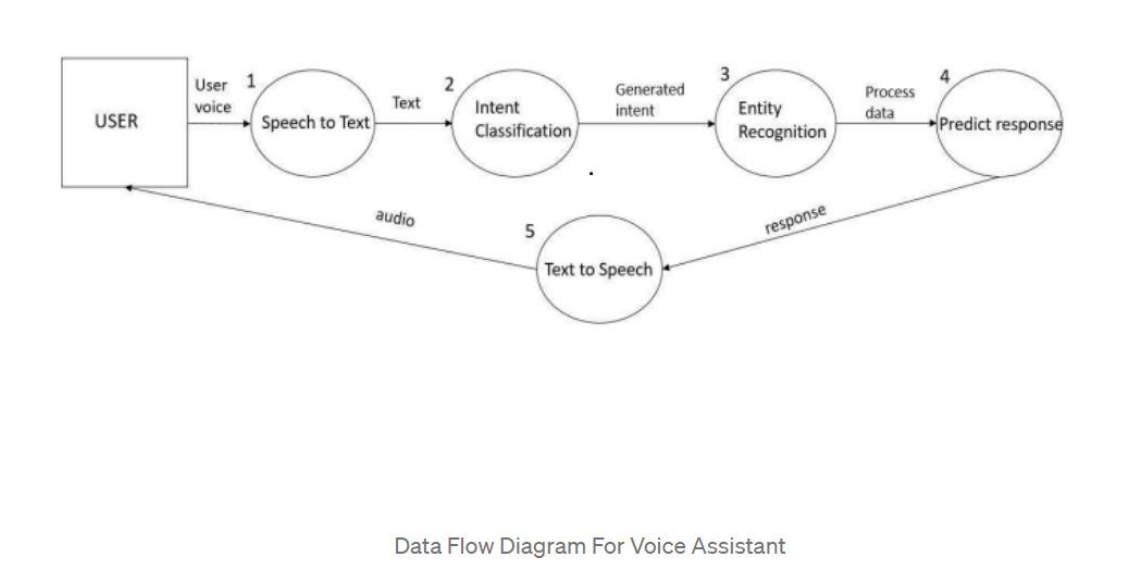
**c. Parametric TTS:**

**Approach:** Parametric methods, such as those used in neural-network-based systems, generate speech dynamically by modeling the vocal tract and sound production process.

**Architecture:**

****

**Data Flow Diagram:**



**Result and Analysis:**

The proposed voice assistant system efficiently handles user commands through speech recognition, pattern matching, URL parsing, and translation functionalities. Speech recognition based on the **Hidden Markov Model (HMM)** accurately converts voice into text by processing speech signals into feature vectors. Regular expressions enhance the system's capability to match specific patterns like URLs or keywords, making it easier to extract relevant data. URL parsing, implemented through **Urllib.parse**, allows seamless interaction with APIs and web-based resources. The use of **encoder-decoder neural networks** enables precise language translation, enhancing accessibility for diverse users. Overall, the system simplifies operations for individuals with disabilities and those seeking hands-free control, providing quick and accurate responses through integrated algorithms.

**Integration of machine learning:**

The integration of machine learning in the JARVIS project enhances its capabilities by leveraging advanced algorithms to improve user interactions and system performance. **Natural Language Processing (NLP)** is powered by machine learning models, including **Hidden Markov Models (HMMs)**, which enable accurate speech recognition by learning from large datasets of spoken language. **Convolutional Neural Networks (CNNs)** are utilized to potentially add visual recognition features, such as object detection, for more advanced interaction scenarios. The **encoder-decoder neural networks** facilitate precise language translation, allowing users to interact in multiple languages. Additionally, machine learning algorithms continually refine the system's responses and accuracy by learning from user interactions, improving the assistant's ability to understand diverse commands and contexts. This dynamic learning approach ensures that the virtual assistant becomes more intuitive and effective over time, providing a personalized and efficient user experience.

## Challenges and future work:

## Challenges:

## Speech Recognition Accuracy: Achieving high accuracy in speech recognition can be challenging, particularly with diverse accents, background noise, and varied speech patterns. Ensuring the system effectively handles these variations is crucial for reliable performance.

## Natural Language Understanding: Interpreting the intent behind user commands accurately requires sophisticated NLP techniques. Ambiguities in language and context can lead to misunderstandings, necessitating advanced algorithms to improve comprehension.

## Integration of Visual Recognition: Adding visual recognition capabilities, such as object detection, involves integrating complex Convolutional Neural Networks (CNNs). Ensuring seamless integration with existing text and voice functionalities while maintaining performance is a significant challenge.

## Resource Constraints: Machine learning algorithms, especially deep learning models, can be resource-intensive. Managing computational resources efficiently while maintaining real-time responsiveness is a key challenge.

## User Privacy and Security: Handling sensitive user data, such as personal information and voice recordings, requires robust security measures to protect privacy and prevent unauthorized access.

## Future Work:

## Enhanced Language Support: Expanding language capabilities to include more languages and dialects will make the assistant accessible to a broader audience. Integrating advanced translation models can further improve multilingual support.

## Context-Aware Responses: Developing more sophisticated models for context-aware responses can enhance the assistant’s ability to handle complex, multi-step commands and understand nuanced user intents.

## Advanced Visual Features: Integrating advanced visual recognition features, such as object detection and scene understanding, can expand the assistant’s capabilities to interact with physical environments and provide more comprehensive support.

## Personalization: Implementing adaptive learning algorithms to tailor responses and functionalities based on individual user preferences and behavior can enhance the user experience and effectiveness of the assistant.

## Offline Functionality: Improving the system's ability to function offline, by incorporating more robust local processing and storage, will ensure continued usability in environments with limited or no internet connectivity.

## Scalability and Performance: Optimizing the system to handle increased user loads and diverse use cases while maintaining high performance and responsiveness is crucial for future scalability.

## Conclusion:

## The JARVIS project represents a significant leap forward in the realm of virtual assistance, integrating advanced Natural Language Processing (NLP) and machine learning technologies to deliver a highly intuitive and efficient user experience. By leveraging Hidden Markov Models (HMMs) for accurate speech recognition, Convolutional Neural Networks (CNNs) for potential visual recognition, and encoder-decoder neural networks for robust language translation, JARVIS effectively bridges the gap between human commands and machine execution.

## The system addresses various user needs, from enhancing productivity and accessibility for individuals with disabilities to simplifying everyday tasks for busy professionals. Its ability to handle diverse commands through both voice and text, coupled with real-time interaction capabilities, sets JARVIS apart from traditional virtual assistants.

## Despite the challenges, including speech recognition accuracy, natural language understanding, and resource constraints, the project shows promising potential for future enhancements. Planned developments, such as expanding language support, improving context-aware responses, and integrating advanced visual features, will further elevate JARVIS's functionality and user satisfaction.

## Overall, JARVIS is not just a virtual assistant but a transformative tool that adapts to evolving user needs and technological advancements, paving the way for a more intelligent and personalized digital interaction landscape.

## BIBILIOGRAPHY

## V. Mitra, H. Franco, M. Graciarena and D. Vergyri, "Medium-duration modulationcepstral feature for robust speech recognition," 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 1749-1753, doi:10.1109/ICASSP.2014.6853898.

## A. Acero, "An overview of text-to-speech synthesis," 2000 IEEE Workshop on Speech Coding. Proceedings. Meeting the Challenges of the New Millennium (Cat. No.00EX421), Delavan, WI, USA, 2000, pp. 1-, doi: 10.1109/SCFT.2000.878372.

## T. Kim, "Short Research on Voice Control System Based on Artificial Intelligence Assistant," 2020 International Conference on Electronics, Information, and Communication (ICEIC), Barcelona, Spain, 2020, pp. 1-2, doi: 10.1109/ICEIC49074.2020.9051160.

## A. Sudhakar Reddy M, Vyshnavi, C. Raju Kumar, and Saumya,”Virtual Assistant Using Artificial Intelligence” in J ETIR March 2020, Volume 7, Issue 3 ISSN-2349-5162.

## G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

## Ardissono, L., Boella. And Lesmo, L. (2000) “A Plan-Based AgentArchitecture for Interpreting Natural Language Dialogue”, International Journal of Human-Computer Studies

## Nguyen, A. and Wobcke, W. (2005), “An Agent-Based Approach to Dialogue Management in Personal Assistant”, Proceedings of the 2005 International Conference on Intelligent User Interfaces.

1. Canbek, N.G. and Mutlu, M.E., 2016. On the track of artificial intelligence: Learning with intelligent personal assistants. Journal of Human Sciences, 13(1), pp.592-601.
2. ] Knote, R., Janson, A., Eigenbrod, L. and Söllner, M., 2018. The What and How of Smart Personal Assistants: Principles and Application Domain for IS research.
3. Wobcke, W., Ho. V., Nguyen, A. and Krzywicki, A. (2005), “ A BDI Agent Architecture for Dialogue Modeling and Coordination in a Smart Personal Assistant”, Proceedings of the 2005 IEEE/WIC /ACM International Conference on Intelligent Agent Technology.