REPUBLIC of TUNISIA
Ministry of Higher Education and Scientific Research University of Carthage
National Institute of Applied Sciences and Technology

END-OF-YEAR PROJECT

Sign Language Recognition

Curriculum: Industrial IT and Automation

Carried out by :
Nour Trabelsi
Ichrak Khaldi
Salwa Mhemed

Supervisor : Mr. Hamdi Med Ali
Reviewer : Mr. Tlili Saber

Academic year: 2024 - 2025

Acknowledgments

At the conclusion of this project, above all, we extend our sincere gratitude to our supervisor, Mr. Med Ali HAMDI, a professor at the National Institute of Applied Sciences and Technology (INSAT), for his warm welcome, thorough explanations, pertinent suggestions, and most importantly, his invaluable advice, which enabled us to successfully undertake this project under optimal conditions.

Contents

List of Figures

GENERAL INTRODUCTION

Sign language recognition (SLR) technology represents a pivotal advancement in fostering communication accessibility for individuals with hearing impairments. This report outlines a comprehensive approach to developing a real-time SLR system using machine learning, specifically convolutional neural networks (CNNs), for accurate gesture recognition.

The project aims to empower the deaf and hard of hearing community by translating sign language gestures into text or speech, facilitating seamless interaction in educational, professional, and social contexts. It reviews existing SLR applications and technological approaches, discusses the integration of computer vision techniques for preprocessing and feature extraction, explores embedded hardware options like Raspberry Pi for practical deployment, and concludes with insights into system implementation, experimental results, and ethical considerations.

By leveraging advancements in machine learning and computer vision, our system seeks to bridge the communication gap, making everyday interactions more inclusive. The deployment on the Raspberry Pi demonstrates the practicality of using affordable hardware for real-time applications, ensuring the system's accessibility to a broader audience.

The insights gained from this project highlight the potential of SLR technology to significantly improve the quality of life for those with hearing impairments, providing a foundation for future advancements in the field.

Abstract

Communication between humans and computers can be facilitated through various methods, one of the most intuitive being gesture recognition. This method is particularly valuable in virtual reality systems and for aiding individuals with hearing and speech impairments. Hand gestures allow deaf individuals to communicate in their daily lives without relying on spoken language. Sign language employs visually transmitted gestures that combine hand shapes, orientations, movements of the hands and arms, lip patterns, body movements, and facial expressions to convey thoughts.

To bridge the gap between sign language and speech, we initially developed our system on a laptop, creating a custom convolutional neural network (CNN) model. This system captures real-time images and converts them to speech, using text as an intermediate step. Given the limitations of a 2D system, our focus was on finger spelling, capturing alphabetic signs via fingerspelling and providing corresponding voice output. After successful development and testing, we deployed the system on a Raspberry Pi for practical application.

With this proposed system, we aim to support the speech-impaired community by facilitating more accessible communication.

Keywords: gesture recognition system, OpenCV, Raspberry Pi, CNN, sign language

chapter 1

State of the Art

1.1 Introduction

It's very difficult for mute people to convey their message to normal people in their day-to-day life. Since normal people are not trained on sign language, the communication becomes very difficult. In an emergency, when a mute person travelling or amongst new people's communication with nearby people or conveying a message becomes very difficult. Here we proposed a smart speaking system that helps mute people in communicating to normal people using hand motions and gestures. The system makes use of a camera to capture hand movement. Text to speech convertor use to convert text into voice.

1.1.1 Project Objective and Motivation:

The primary aim of this project was to develop a neural network capable of classifying the letters of the American Sign Language (ASL) alphabet from images of signing hands. This endeavor marks a preliminary step towards creating a comprehensive sign language translator that can convert ASL gestures into written and spoken language. Such a tool would significantly reduce the communication barriers faced by deaf and mute individuals, facilitating better interactions in daily life. Given that normal people are generally not trained in sign language, mute individuals often struggle to convey their messages, especially in emergencies or unfamiliar settings. This project proposes a smart speaking system that uses a camera to capture hand movements and a text-to-speech converter to translate these gestures into audible speech, thereby enhancing communication between mute individuals and the broader community.

1.1.2 Importance and Technological Approach:

The statistics highlight the global prevalence of deaf and mute individuals, with estimates ranging from 700,000 to 900,000 people worldwide. In the United States alone, approximately 600,000 individuals are deaf, and about 6,000,000 people report significant hearing trouble. Despite the rapid advancements in technology, there remains a lack of substantial developments aimed at improving communication for deaf and mute individuals. This project leverages the power of artificial intelligence (AI), specifically neural networks, to address this gap. AI's ability to simulate human intelligence processes, including image recognition and natural language processing, makes it an ideal tool for translating sign language into written and oral forms. By creating algorithms that transform visual hand

gestures into understandable speech, this project aspires to foster inclusivity and reduce the communication barriers that deaf and mute individuals face.[1]

1.2 Sign Language Recognition

Sign Language Recognition (SLR) is a breakthrough facilitated by emerging technologies like artificial intelligence and machine learning, aimed at bridging communication barriers for the deaf community. Despite years of research, there's still no universally high-performance solution, but recent attention from researchers is driving the development of commercially viable options. Various methods and approaches are being explored, each with its unique strengths. These diverse strategies contribute to the ongoing efforts to create effective SLR solutions.[2]

1.2.1 Applications :

- **Education and Learning:**

Integration of SLR systems in educational settings supports deaf or hard-of-hearing students by providing real-time translation of sign language into written or spoken language. This fosters inclusive learning environments and facilitates communication between teachers and students.

- **Customer Service and Support:**

SLR systems facilitate communication between customer service representatives and sign language users, enhancing overall customer experience and ensuring effective communication in support settings.

- **Accessibility in Public Spaces:**

Incorporating SLR systems in public spaces like train stations, airports, or government offices ensures vital information is accessible to sign language users. Real-time translation of announcements, instructions, or directions promotes equal access to information.

- **Rehabilitation and Therapy :**

SLR systems assist in rehabilitation and therapy programs for individuals with speech or hearing impairments by providing real-time feedback and assistance in learning and practicing sign language, thereby improving the effectiveness of therapy sessions.

1.2.2 Existing Systems :

Existing systems for sign language recognition can be categorized into two types: static and dynamic recognition. Static recognition involves detecting gestures from a 2D image, while dynamic recognition captures gestures in real-time using a camera. Static recognition is particularly useful in scenarios where capturing frames in a live camera preview is challenging due to factors like poor lighting, weather conditions, or camera quality.

One notable solution in the realm of static recognition is the Intelligent models for Sign Language Recognition (SLR), which detect signs from camera inputs or images. Although this solution has provided significant benefits for the deaf community, its adoption

remains limited due to the lack of accessibility and user-friendly alternatives.

On the other hand, smart gloves represent a dynamic recognition approach. These gloves feature thin and stretchable sensors along the length of each finger, capturing finger placements and hand motions that correspond to individual letters, numbers, words, and phrases in real-time. However, this solution presents practical challenges, as it involves numerous sensors and cables, which may not function optimally together.

Figure 1.1: Sign language recognition glove (Daily Mail)

1.3 Machine Learning Models

Machine learning models are categorized as either supervised or unsupervised. If it's a supervised model, it's then sub-categorized as a regression or classification model.

Decision Trees:

A predictive approach that determines the class of an object by following a tree-like flow chart based on certain conditions.[3]

Support Vector Machines (SVMs):

An algorithm that separates datasets using lines (hyperplanes) and maximizes the distances between the separating line and the different samples on either side.[3]

K-Nearest Neighbors (KNN):

A lazy learner that does not create a model immediately from the training data but instead memorizes the training data and uses it to make predictions by finding the nearest neighbor from the whole training data.[3]

Neural Network :

a computational model inspired by the human brain, consisting of layers of interconnected nodes (neurons) that process input data to identify patterns and make predictions.[3]

We opted to use a Convolutional Neural Network (CNN) for our sign language recognition model because CNNs are highly effective for image-related tasks. They excel at automatically learning spatial hierarchies of features from images through convolutional layers, making them ideal for recognizing and distinguishing the intricate hand shapes and movements in sign language images or videos.

1.3.1 Deep learning:

Deep learning uses multiple processing layers to learn data representations with various levels of abstraction. It has significantly advanced fields like speech recognition, visual object recognition, object detection, drug discovery, and genomics. By employing the backpropagation algorithm, deep learning adjusts internal parameters to refine these representations layer by layer.[1]

Convolutional Neural Network (CNN):

A CNN is a type of deep learning algorithm particularly well-suited for image recognition and processing tasks. It consists of multiple layers, including:

- Convolutional layers: These apply convolutional operations to input images using filters (kernels) to detect features like edges, textures, and complex patterns.
- Pooling layers: These downsample the spatial dimensions of the input, reducing computational complexity.
- Activation functions: Non-linear functions (e.g., ReLU) introduce non-linearity to the model.
- Fully connected layers: Responsible for making predictions based on high-level features learned by previous layers.[4]

1.4 Computer Vision Techniques

- Image Processing and Enhancement : Techniques to improve image quality, adjust lighting, remove noise, and enhance clarity.
- Feature Detection and Description: Methods to identify key points, edges, corners, or blobs in images.
- Segmentation: Partitioning images into meaningful regions or segments based on characteristics like color, intensity, or texture.
- Object Detection and Recognition: Algorithms to locate and classify objects within images or video frames.
- Motion and Tracking: Techniques to analyze and track the movement of objects over time in video sequences.
- 3D Vision: Methods for depth estimation, 3D reconstruction, and understanding spatial relationships from 2D images.

-> The techniques used in our system are : Feature Detection and Object Detection and Recognition

1.5 Exploring Embedded boards Options

- Arduino Pro Mini: A smaller, minimalistic version of the Arduino Uno, suitable for projects with space constraints or where low power consumption is critical.[5]
- Arduino Nano: Ideal for sensor interfacing and low-power applications due to its compact size and low cost.[5]
- NVIDIA Jetson Nano: Suitable for AI and machine learning projects, offering high computational power for image processing and deep learning tasks.[6]
- ESP32: Excellent for IoT applications with built-in Wi-Fi and Bluetooth capabilities, enabling seamless connectivity to the internet and other devices.
- Raspberry Pi: A microcontroller board with MicroPython support, perfect for lightweight embedded projects requiring simple programming and GPIO access.[6]

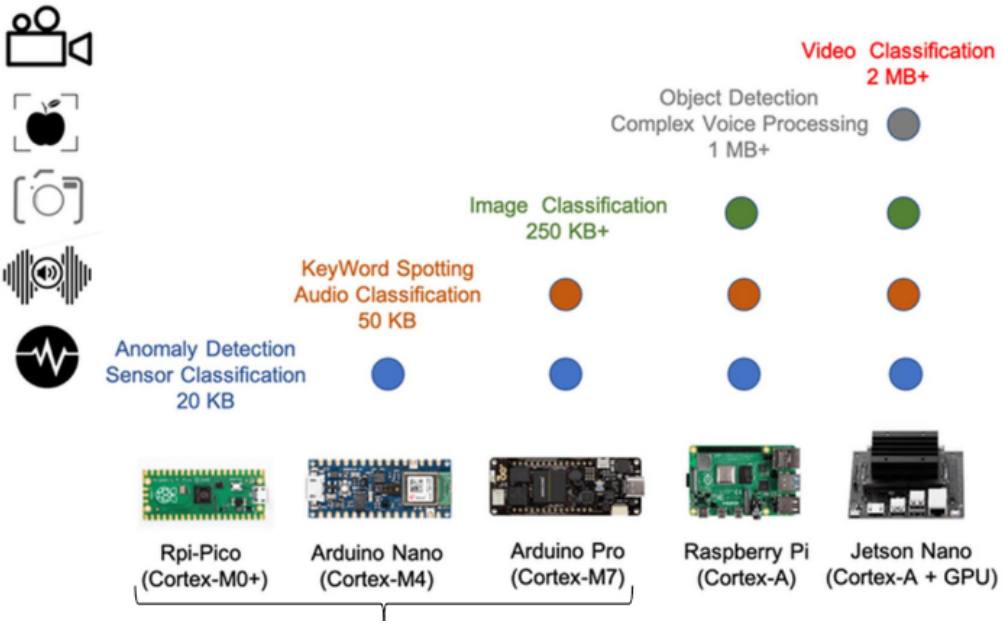

Figure 1.2: Embedded Boards Performance Comparison

We opted for **the Raspberry Pi** for our sign language recognition system because it strikes a perfect balance between cost, efficiency, and versatility. Its affordability ensures accessibility, while its robust processing power handles image recognition tasks effortlessly. With USB ports, it easily connects to cameras and speakers, essential for capturing gestures and converting them into speech. Its compatibility with various operating systems and programming languages simplifies development, and users can conveniently monitor and control the system through a laptop or phone interface.

1.6 Conclusion

This chapter reviewed the state of the art in sign language recognition, focusing on using artificial intelligence and deep learning to facilitate communication for deaf and mute in-

dividuals. We outlined the project's goal to develop a neural network that identifies ASL letters. Key computer vision techniques and the role of Convolutional Neural Networks (CNNs) were discussed. Additionally, the use of Raspberry Pi for real-time image recognition was highlighted. This chapter provides the foundation for developing an effective ASL recognition system.

chapter 2

Proposed Methodology

2.1 General System Architecture

Sign language recognition systems follow a pipelined architecture with data flowing sequentially between components. The initial stage involves image or video acquisition using a camera. This data is then fed into a pre-processing module, where operations like resizing, normalization, and background removal might occur to prepare the data for further processing. Next comes the hand detection component, which identifies and isolates the hand region within the frame. Extracted hand images are then passed to the classification module, which employs machine learning models to analyze the hand shape and posture, ultimately recognizing the intended sign language gesture. Finally, the recognized gesture can be visualized on the screen or translated into text or voice output for user comprehension.
The image below illustrates the recognition system architecture.

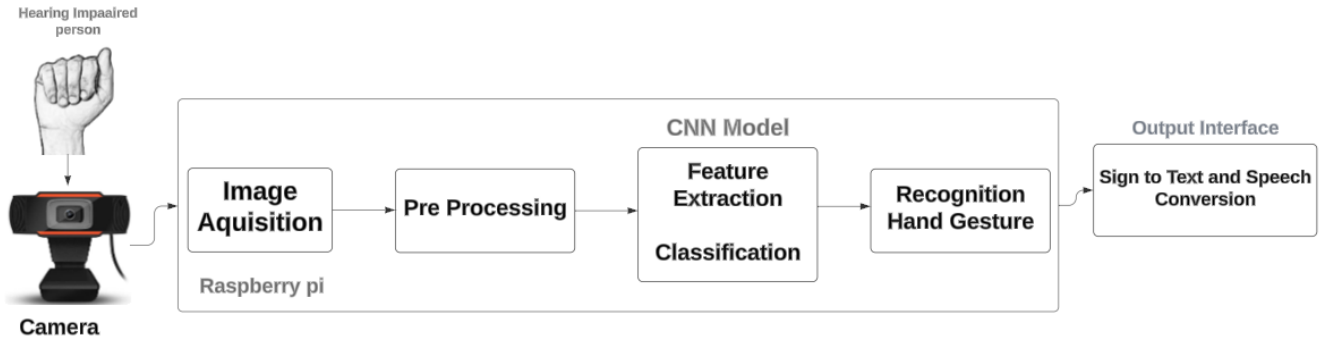

Figure 2.3: Recognition System Architecture

2.2 Sequence Diagram:

In the following sequence diagram, our system is depicted, detailing the interactions between the user, the camera, the Raspberry Pi, the CNN model, the text concatenator, the display interface, and the speaker.

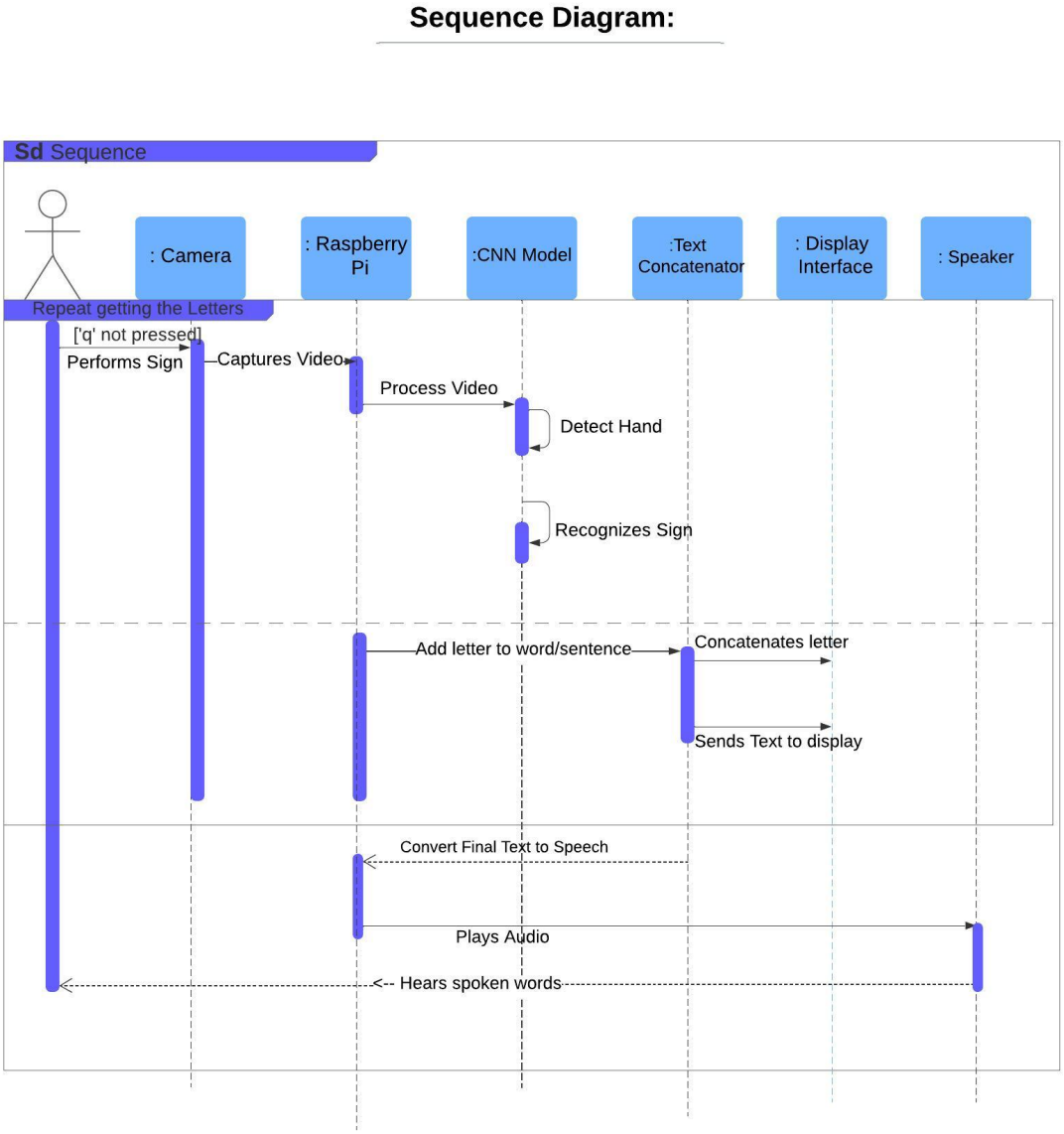

Figure 2.4: Sequence Diagram

2.3 Hardware and Software Requirements

2.3.1 Hardware components :

Raspberry Pi 3 Model B+ (Processing Unit): This serves as the heart of the system, responsible for capturing video input, processing images, and performing sign language recognition tasks. The Raspberry Pi 3 Model B+ offers a good balance of processing power, energy efficiency, and affordability, making it ideal for this project.

Its specifications include:

- 1.4 GHz 64-bit quad-core ARM Cortex-A53 CPU
- 1GB LPDDR2 SDRAM

- Dual-band 2.4 GHz and 5 GHz Wi-Fi LAN
- Bluetooth 4.2
- Gigabit Ethernet port[7]

In addition to the Raspberry Pi 3 Model B+, which serves as the processing unit, several other peripherals are essential for the setup and operation of the system. A mouse and keyboard are necessary for interacting with the Raspberry Pi during setup and configuration. A monitor is required to visualize the Raspberry Pi's output, especially during development and debugging. Additionally, Wi-Fi connectivity ensures that the Raspberry Pi can connect to the internet for updates, downloading software, and communicating with other devices. These components are illustrated in the accompanying image :

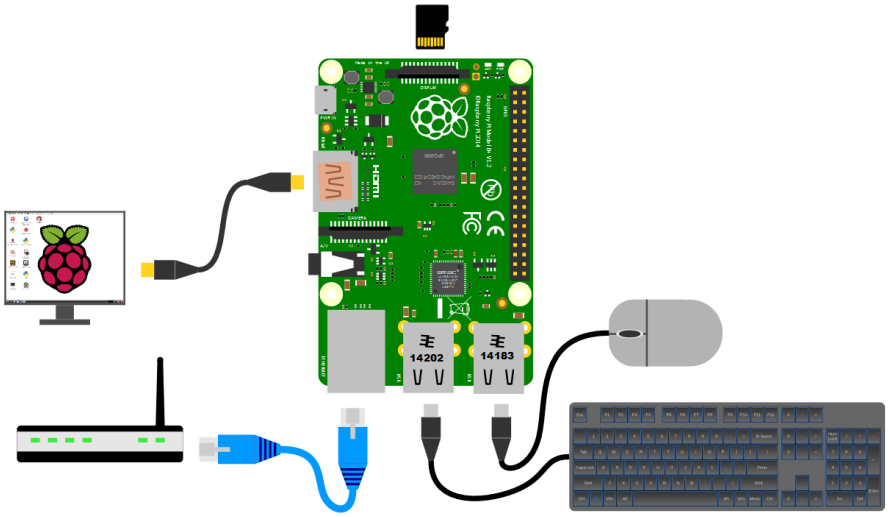

Figure 2.5: Hardware Components

To enable the sign language translation functionality,**USB camera** and **Speaker** are required. The USB camera captures real-time video or images of hand gestures used for sign language communication, connecting directly to the Raspberry Pi to provide a convenient and integrated solution. The speaker allows the system to provide audio feedback through synthesized voice using software like "pyttsx."

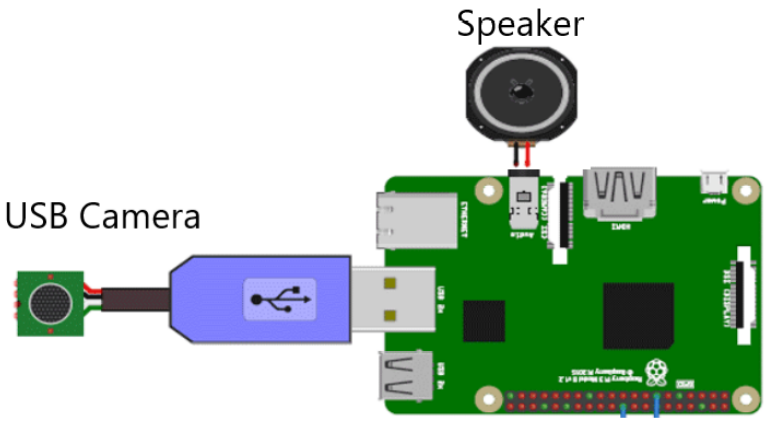

Figure 2.6: Hardware Requirements

2.3.2 Software Requirements :

1. **Operating System:**

   We utilized Raspberry Pi OS (formerly known as **Raspbian**), which is a Debian-based operating system specifically optimized for Raspberry Pi hardware. Raspberry Pi OS provides a stable and efficient platform for running applications and projects on Raspberry Pi devices. This operating system offers a wide range of pre-installed software packages and tools tailored to the needs of Raspberry Pi users, making it an ideal choice for our project.

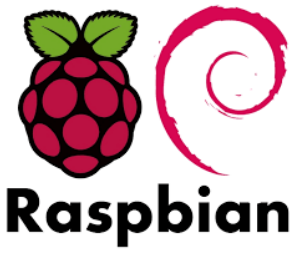

Figure 2.7: the Raspberry Pi OS

   The installation of Raspberry Pi OS was facilitated through the use of **Pi Imager**, a graphical utility developed by the Raspberry Pi Foundation. Pi Imager simplifies the process of installing operating systems onto Raspberry Pi devices by providing a user-friendly interface and step-by-step instructions. With Pi Imager,

   Additionally, we utilized **RealVNC** for remote desktop access to our Raspberry Pi environment. RealVNC is a cross-platform remote desktop software that enables users to remotely access and control Raspberry Pi devices from other computers or mobile devices.

2. **Development Environment:**

   PyCharm: Our primary development environment was PyCharm, a powerful integrated development environment (IDE) for Python programming. PyCharm provided a rich set of features for code editing, debugging, and version control integration, enhancing our productivity and code quality.

Figure 2.8: the PyCharm IDE

3. **Python libraries :**

- **OpenCV:**
  OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. It mainly focuses on image processing, video capture, and analysis including features like face detection and object detection. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being an open-source licensed product, OpenCV makes it easy for businesses to utilize and modify the code. [8]

Figure 2.9: open-source computer vision

- **Mediapipe :** Mediapipe is a graph-based framework for building multimodal (video, audio, and sensor) applied machine learning pipelines. It offers cross-platform customizable machine learning solutions for live and streaming data. Mediapipe together with OpenCV has allowed our team to train models so that the machine could detect our hand more consistently and accurately while testing the prototype. Mediapipe is very vital as our project would not have been successful without it.

  Mediapipe hands:

  The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. For example, it can form the basis for sign language understanding and hand gesture control and can also enable the overlay of digital content and information on top of the physical world in augmented reality. Mediapipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame.[9]

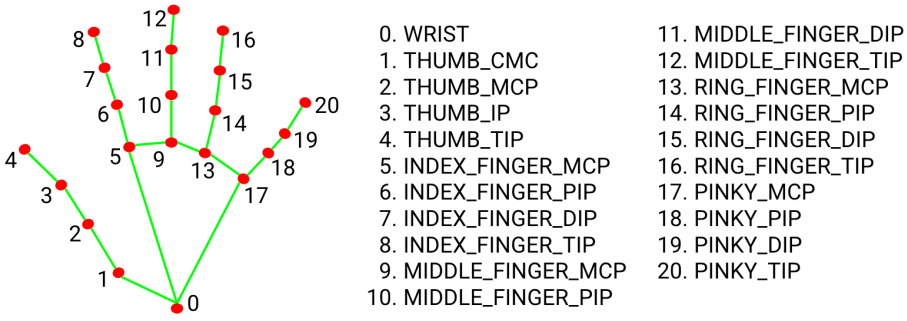

Figure 2.10: Mediapipe Hands

- **NumPy:** A fundamental library for numerical computing in Python. It offers efficient array manipulation capabilities useful for image representation and processing.
- **Matplotlib:** A plotting library used for creating static, animated, and interactive visualizations in Python. Useful for plotting data and model performance metrics.
- **scikit-learn:** A machine learning library in Python. It's used here for splitting the dataset into training and testing sets.
- TensorFlow and Keras: TensorFlow is an end-to-end open-source platform for machine learning, and Keras is its high-level neural networks API. They are used for building and training the CNN model.
- **cvzone:** A high-level module that simplifies many tasks such as hand tracking and gesture recognition. Used for detecting hand movements.
- **Tkinter:** A standard GUI (Graphical User Interface) library in Python. Used for creating graphical interfaces for the project.
- **PIL (Python Imaging Library):** A library that provides extensive file format support, an efficient internal representation, and powerful image processing capabilities.
- **OS:** A standard library for interacting with the operating system. Used for file and directory management.
- **Math:** A standard library providing mathematical functions.
- **Time:** A standard library for time-related functions.
- **TensorFlow:** A free and open-source software library for machine learning and artificial intelligence. It is a versatile tool that can be applied across a range of tasks, providing developers with the capabilities to build and deploy machine learning models efficiently. Developed by researchers at Google, TensorFlow has become a prominent framework in the field of machine learning, offering a wide array of functionalities for training, testing, and deploying machine learning models across various platforms and devices[10]

Figure 2.11: TensorFlow

- **TensorFlow Lite:** A mobile library designed to deploy models on mobile devices, microcontrollers, and other edge devices.We used TensorFlow Lite because we deployed the model on a Raspberry Pi, which has limited storage space.[11]

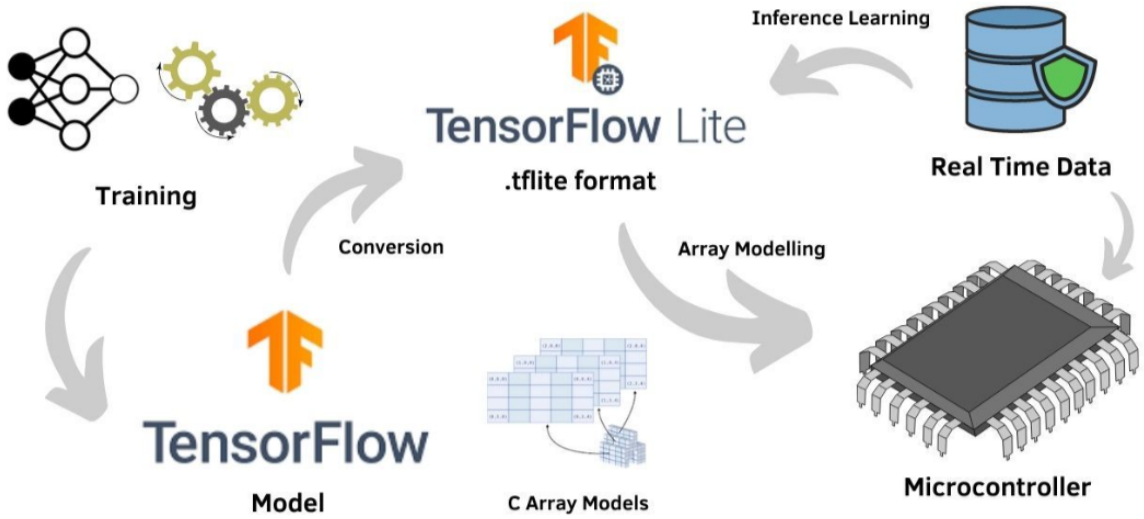

Figure 2.12: Migrating from TensorFlow to TensorFlow Lite for Efficient Deployments

- **Streamlit:**
  Streamlit is an open-source Python framework for data scientists and AI/ML engineers to deliver dynamic data apps with only a few lines of code.

2.4 Conclusion

In this chapter, we outlined how to develop a sign language recognition system using computer vision techniques. The system captures gestures with a USB camera, processes them to isolate hand movements, and uses machine learning to recognize sign language gestures. We chose the Raspberry Pi 3 Model B+ for its affordability and processing power. Key software includes OpenCV and TensorFlow for image processing and machine learning tasks. Chapter 3 will cover implementation details such as model training and system setup.

chapter 3

Practical realization

3.1 Introduction

This project is developed for the impaired people and would be beneficial as they can communicate with everyone. In our system a camera is placed in front of the physically impaired person. The physically impaired person will place a hand with particular action in front of the camera. When he makes the gestures, the camera will capture the exact positions of the fingers and hand and perform image processing and then feature extraction would be done. This way physically impaired person will be able to go through the entire sentence that he wants to communicate. Later on, this sentence will be translated into speech by some text-to-speech algorithms so that it would be audible to everyone through the speakers. By using this system, the physically impaired people and mostly mute people would be benefited as they can communicate with anyone freely which indeed would be great achievement for the mankind.

3.2 Model Implementation Steps

Below are the detailed steps followed in the implementation of our CNN model :

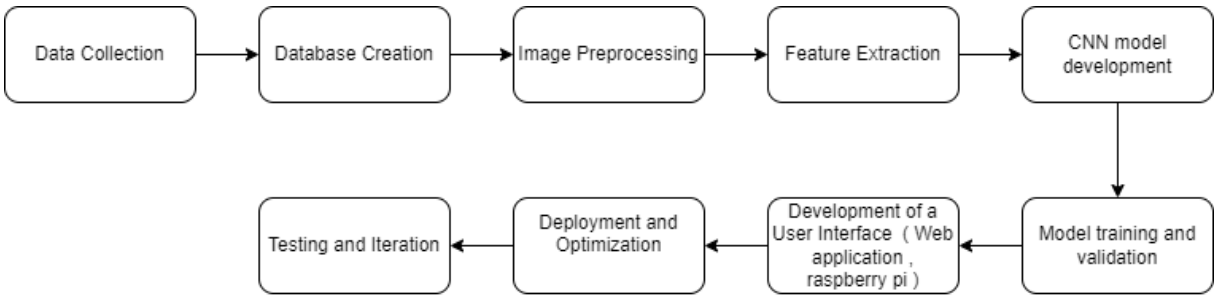

Figure 3.13: End-to-End CNN Image Classification Workflow

3.2.1 Data Collection:

the data used in this project is a mix of a data of our own that we collected using a python algorithm and a webcam , the the compiled dataset of American Sign Language (ASL) called the ASL Alphabet from Kaggle user Akash and the data set “ American

Sign Language Dataset “ from Kaggle user AYUSH THAKUR . The number of data in total is about 66,700 images

3.2.2 Database Creation:

After collecting the data , We divided it into 37 classes, (26 for the alphabet ,10 for numbers from 0 to 9 , and 1 for space) , each having about 2300 images.

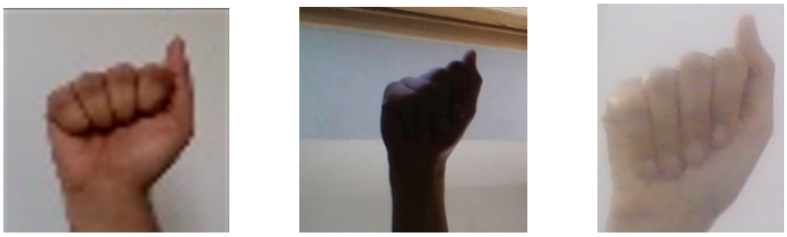

Figure 3.14: Examples of the signed letter A with different hands

3.2.3 Image Preprocessing:

Resizing and Normalization : consists of resizing each image to (50x50 pixels) , converting it to grayscale , and normalizing it by scaling the pixel values to the range of [0,1] to ensure uniformity.
the preprocessing was done using the different functions of the cv2 library .
This is a crucial step to prepare the images before feeding it into the model . It allows us to eliminate unwanted distortions and enhance specific qualities of the image and it helps shorten model training time and speed up model inference.

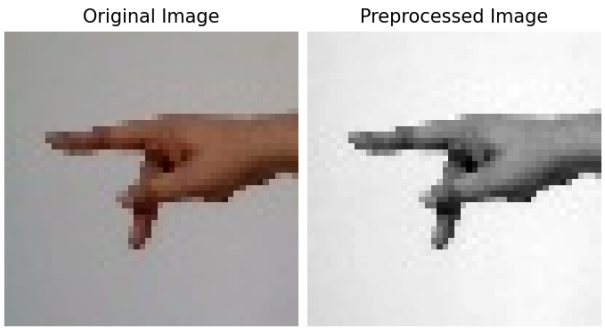

Figure 3.15: Example of an image before and after preprocessing

Data Augmentation

3.2.4 Feature extraction:

This part consists of transforming the preprocessed data to a format that can be fed into the CNN model . In our model , the extraction occurs within the CNN layers.

3.2.5 CNN Model Development:

The model used in this classification task is a fairly basic implementation of a Convolutional Neural Network (CNN).
As the project requires classification of images, a CNN is the go-to architecture.
This model consists of convolutional blocks containing two 2D Convolutional Layers with ReLU activation, followed by Max Pooling and Dropout layers.
These convolutional blocks are designed to progressively extract and downsample the features from the input images.
The model starts with a 2D Convolutional Layer with 32 filters and a kernel size of 3x3, followed by a Max Pooling layer with a pool size of 2x2. This pattern is repeated with a 2D Convolutional Layer with 64 filters. After the convolutional blocks, the output is flattened and passed through several Fully Connected layers with ReLU activation, each followed by Dropout layers to avoid overfitting. The final layer is a Dense layer with 37 units and softmax activation, which outputs the probability distribution over the 37 categories.
This architecture effectively combines convolutional layers for feature extraction with dense layers for classification.

3.2.6 Model Training:

For training we used the Adam optimizer with a learning rate of 0.001, trained over 10 epochs with a batch size of 32, and we employed the categorical cross-entropy as a loss function. Adam optimizer was chosen for its efficiency in convergence and handling of momentum, ensuring robust learning dynamics throughout the training process. This configuration helped optimize our model's performance, achieving competitive results in our evaluation metrics.

3.2.7 Model Evaluation:

to evaluate the model , we used several metrics :

1. **Accuracy :** it measures the ratio of correct predictions to total predictions, providing an overall performance indicator. During training, accuracy is tracked for both the training and validation sets.

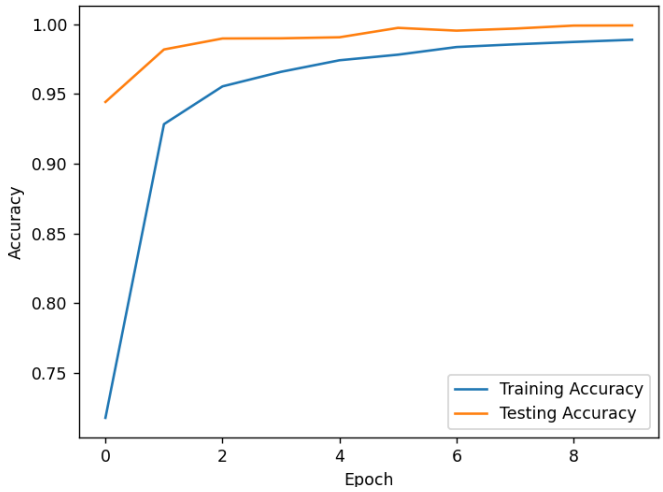

Figure 3.16: Training and Validation Accuracy

The plot shows the training and testing accuracy of our CNN model over 10 epochs. The Training accuracy starts at around 0.1 in the first epoch and it increases rapidly over the first epochs then it becomes steady reaching almost 0.99 by the 10th epoch. That shows that our model is learning effectively from the training data and is improving its performance over time.

The testing accuracy starts higher than the initial training accuracy, around 0.95 It quickly reaches a high value, close to 0.99 by the 3rd epoch. After reaching this point, the testing accuracy stabilizes, showing minor fluctuations but remaining consistently high.

-> The training accuracy and testing accuracy are both very high and close to each other, suggesting that the model is not overfitting to the training data.

2. **Loss, specifically categorical crossentropy :** it evaluates the difference between predicted and actual class distributions, with lower values indicating better performance.

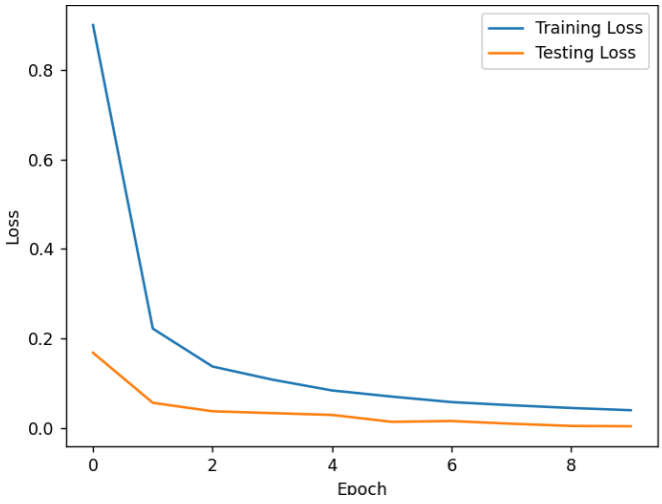

Figure 3.17: Training and Validation Loss

The training loss starts relatively high, around 0.9 in the first epoch. It decreases rapidly in the initial epochs, dropping significantly by the second epoch. The loss continues to decrease gradually, reaching close to 0.1 by the 10th epoch. This indicates that the model is effectively learning from the training data and minimizing the error.
The testing loss starts lower than the initial training loss, around 0.2. It decreases rapidly in the first two epochs, showing a similar trend to the training loss. After the initial drop, the testing loss stabilizes and continues to decrease slightly, staying very low and close to 0 by the 10th epoch.
-> The low testing loss indicates that the model is performing well on unseen data.

3. **The confusion matrix :** it visualizes the model's performance by comparing true labels with predicted labels, highlighting correctly and incorrectly classified instances for each class.

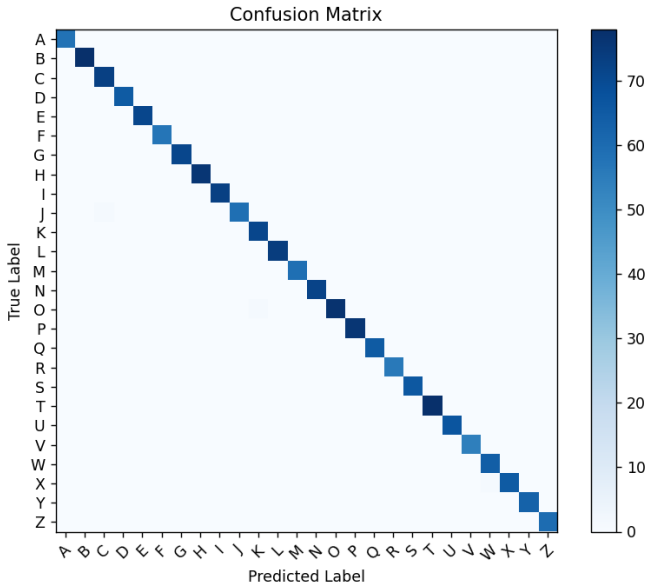

Figure 3.18: The Confusion Matrix

The confusion matrix is predominantly diagonal, with strong dark blue squares along the diagonal line from the top-left to the bottom-right. This indicates that the model correctly predicts the majority of the instances for each letter, signifying high accuracy.

4. **The F1 score :**the harmonic mean of precision and recall, is particularly useful for imbalanced datasets as it balances false positives and false negatives. Together, these metrics offer a comprehensive evaluation of the model's effectiveness and areas for improvement.
F1 Score = 0.98
-> our model has achieved both high precision and high recall which means that our model has effectively learned the patterns in the data and can generalize well to unseen examples.

3.2.8 Development of a user Interface :

We used Tkinter to build a user-friendly interface for our model.

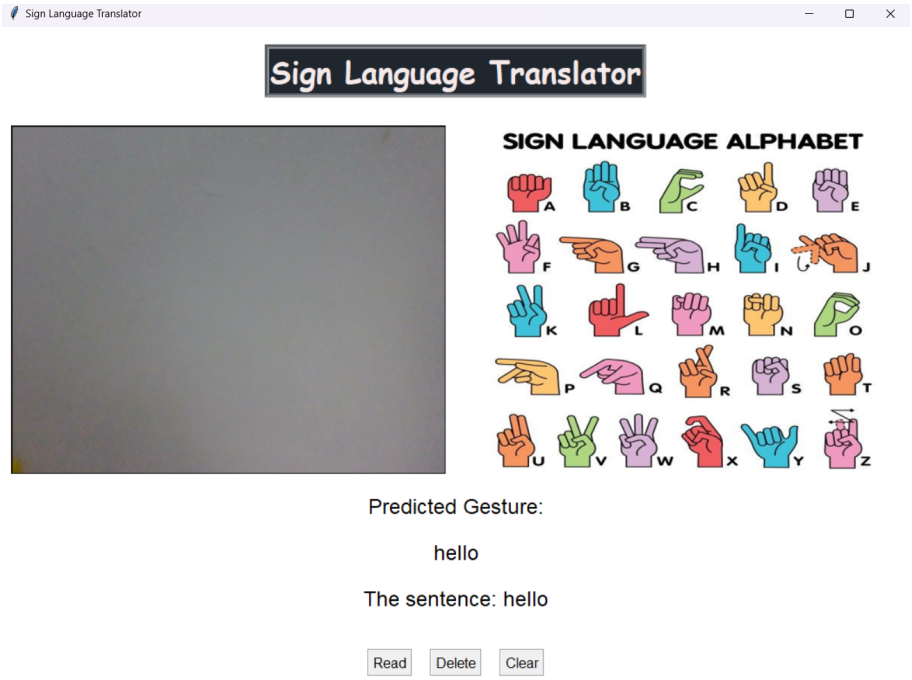

Figure 3.19: User Interface Using Tkinter

We Also used streamlit to build a simple web application

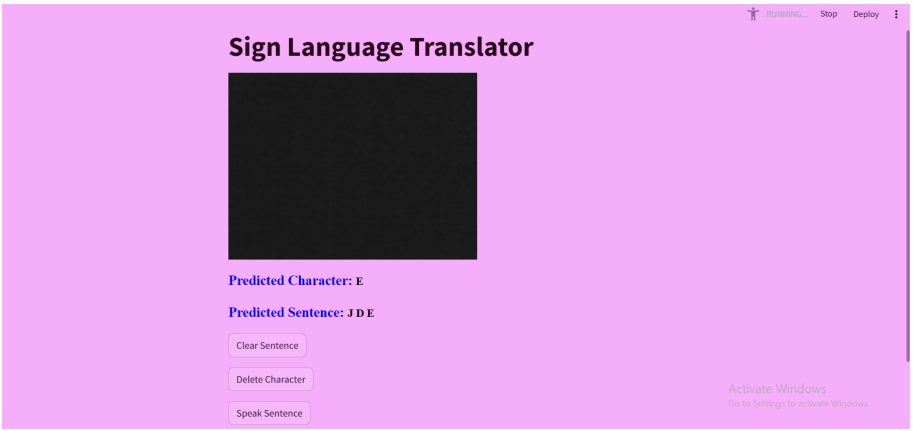

Figure 3.20: User Interface Using Streamlit

3.3 Integration with Raspberry Pi

1.set up Raspberry Pi 3B+ :
Installation of Raspberry Pi OS on microSD card:

The Raspberry Pi requires an operating system to function effectively, just like any other computer. Despite its small size, the Raspberry Pi is a fully capable computer. The operating system enables applications to interact with and utilize the hardware.

Installing Raspberry Pi OS on a microSD card is the first step to making the Raspberry Pi operational and ready for use.

- We downloaded and installed the latest version of the Raspberry Pi Imager.
- We insert the SD card into the computer with a card reader or an adapter.
- We are open Raspberry Pi Imager and choose the required OS from the list presented.

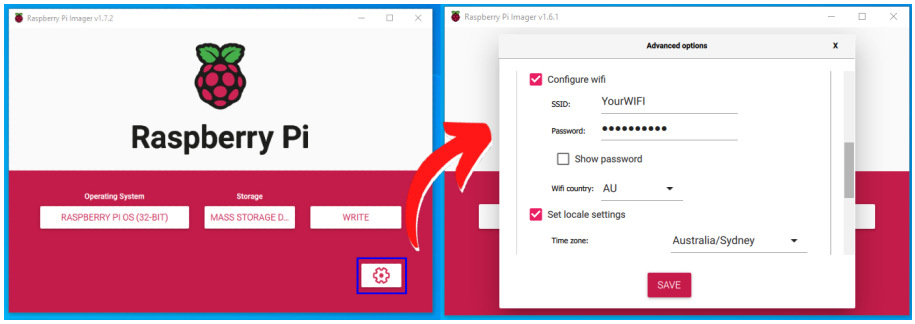

Figure 3.21: PI imager

- In the device box, we have selected the correct SD card. Click "Write" and wait for the writing process to complete.
- We Exit the imager and eject the SD card

2. Virtual Network Computer Setup :

Running the VNC server on the Raspberry Pi enables us to establish a connection via Ethernet or by specifying the device's IP address in VNC Viewer. This connection allows for controlling the Raspberry Pi's interface by forwarding keyboard and mouse events to the VNC server. Additionally, the VNC server sends screen updates back to the viewer, ensuring real-time synchronization of the desktop environment.
Upon activation of the VNC connection, users can conveniently view and interact with the Raspberry Pi's desktop directly within the window of their computer or mobile device, simplifying the management and monitoring of the Raspberry Pi environment.

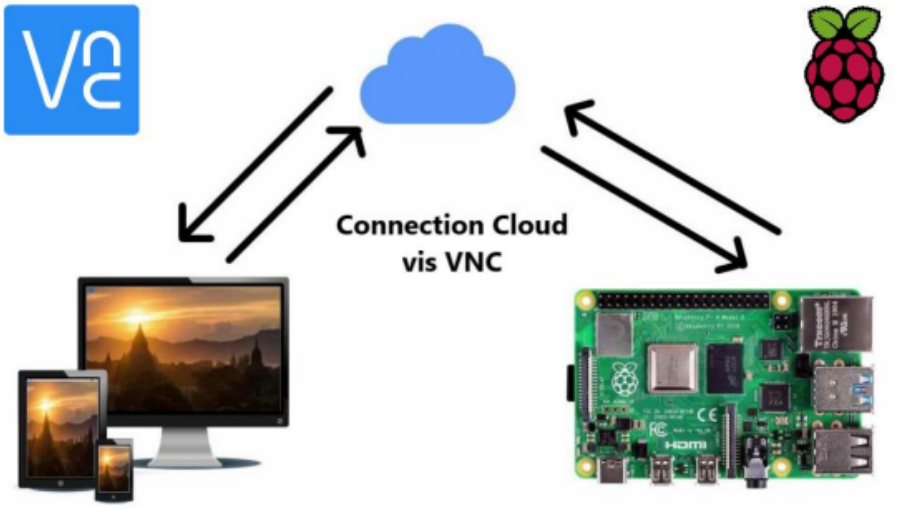

Figure 3.22: RealVNC and Raspberry Pi

3. Virtual Environment Setup and Library Installation :
To streamline the installation of requisite libraries for our sign language recognition system on the Raspberry Pi, we leveraged virtual environments.
Thonny, an intuitive Python Integrated Development Environment (IDE), facilitated the creation of virtual environments, enabling segregated installations of libraries tailored to individual projects. This methodology ensures that dependencies remain isolated, preventing conflicts with system-wide installations, and offers convenient management of project-specific requirements.
By employing virtual environments, we surmounted any installation hurdles encountered, guaranteeing that our sign language recognition system had unfettered access to all indispensable dependencies. The Raspberry Pi's graphical user interface (GUI) provided an optimal environment for Thonny, as depicted below:

Figure 3.23: Thonny IDE location on Raspberry Pi

To establish the virtual environment and install the requisite libraries, we executed a series of commands:

```
sudo apt install -y python3-pip python3-virtualenv
cd ~/Desktop
mkdir projects
cd ~/Desktop/projects
python3 -m venv projectsenv
ls -l
pip3 install opencv-contrib-python
source projectsenv/bin/activate
```

These commands orchestrated the installation of pip3 and Python 3 virtual environment, creation of a virtual environment dubbed 'projectsenv', installation of the 'opencv-contrib-python' library, and activation of the virtual environment for project utilization. This meticulous setup process ensured a seamless and efficient establishment of the sign language recognition system on the Raspberry Pi.

Subsequently, we proceeded to install all requisite Python libraries, including but not limited to numpy, cv2 (OpenCV), mediapipe, and tkinter. However, a notable challenge arose during the installation of Keras, a component of TensorFlow, due to the Raspberry Pi's limited RAM capacity. Consequently, we adopted an alternative approach to accommodate the system's constraints.

4. Code Execution and Results :
Upon execution, the system successfully captured live video input from the camera and processed it to detect hand gestures. The **Mediapipe** library facilitated the extraction of hand landmarks and key points, enabling precise localization of hand movements.

The sign language recognition system accurately interpreted the detected gestures, translating them into corresponding letters, numbers, or words. Utilizing **OpenCV** for image processing and manipulation, the system displayed the recognized gestures in a graphical user interface created using **tkinter**.

Despite the hardware constraints encountered during the installation of Keras, we successfully converted our proprietary model to **TensorFlow Lite** (TFLite), which is the cornerstone of our project. The conversion process involved optimizing the model to ensure it runs efficiently on the limited computational resources of the Raspberry Pi. TFLite's lightweight and efficient model execution capabilities allowed the system to perform gesture recognition tasks swiftly and accurately.

The resulting system demonstrated its capability to interpret sign language gestures in real-time on the Raspberry Pi, maintaining high performance and accuracy. Below is the result:

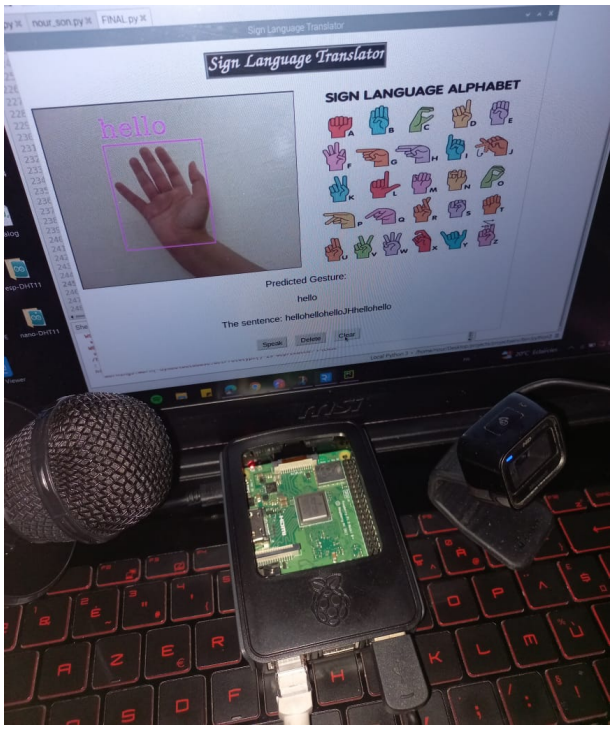

Figure 3.24: Sign Language System on Raspberry Pi

3.4 Results

We tested the model by signing our names , these are the results

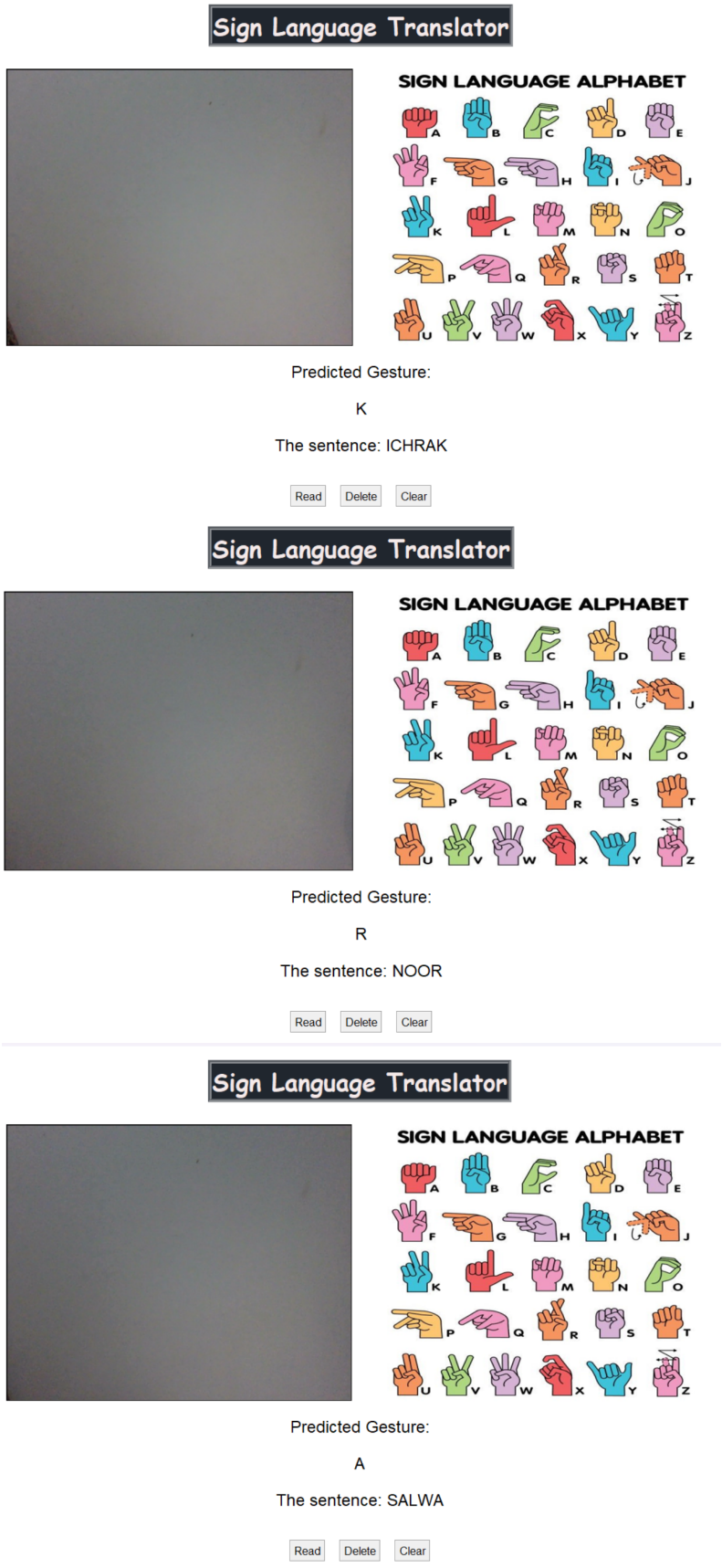

Figure 3.25: Model Results on Our Team Members Names

3.5 Reflection

3.5.1 Difficulties and Challenges :

While the project has shown promising results, there are certain limitations that constrain its effectiveness and applicability such as

Data Availability :

Collecting a large and diverse dataset is essential for training an accurate model. This dataset must include various signers of different ages, genders, ethnicities, and signing styles to ensure the model generalizes well. Additionally , High-quality data is required, with consistent lighting, resolution, and framing. accurate and detailed annotation of each sign is necessary, which can be time-consuming and costly or difficult to find online. There are relatively few large, publicly available sign language datasets compared to datasets for other computer vision tasks. This limitation can limit the performance of the model.

Similarity Between Signs

Despite efforts to minimize error rates, the model may still struggle with accurately distinguishing between signs, especially those that are visually similar. Many signs can be very similar, differing only in hand shape, orientation, or slight movements. Distinguishing these subtle differences is quite difficult and it requires high precision in the model's feature extraction and recognition processes.

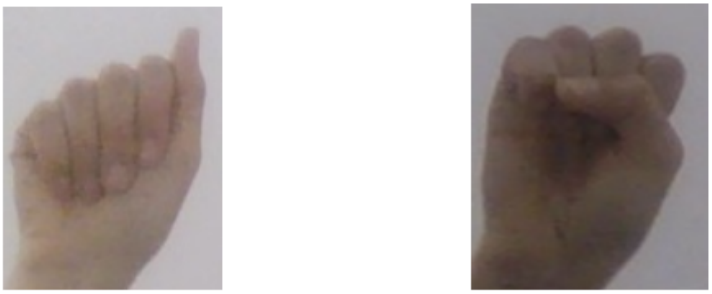

Figure 3.26: similarity between the ASL gesture for A and S

Figure 3.27: similarity between the ASL gesture for R , U and D

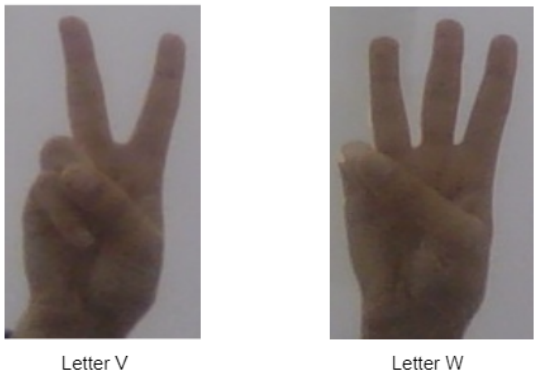

Figure 3.28: similarity between the ASL gesture for V and W

Computational Requirements

Training the CNN model on a large dataset requires substantial computational resources, with high-performance GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units) often necessary to handle the intensive computations involved.Additionally, even with powerful hardware, the training process was time-consuming, especially when fine-tuning hyperparameters or experimenting with different architectures. These long training times slowed down the development process and iteration speed, making it difficult to advance the model efficiently.

Real-Time Processing

Sign language is dynamic, with varying speeds and fluidity in signing. The model needs to handle rapid movements and transitions between signs without losing accuracy. Achieving real-time recognition requires efficient algorithms and powerful hardware. Balancing speed and accuracy is a significant technical hurdle.

Hardware Limitations of Raspberry Pi 3 B+ :

While affordable and accessible, the Raspberry Pi 3 B+ has significant hardware limitations. Its constrained processing power, memory, and storage limit the complexity of neural network models that can be deployed. Real-time processing may be hindered, requiring careful algorithm optimization. Additionally, the need for peripheral devices and reliable power and network connectivity can introduce operational constraints. These limitations may restrict the project's applicability in resource-constrained environments and scenarios requiring high computational performance, affecting widespread adoption and scalability.

3.5.2 Ethical Issues:

Representation

The dataset needs to be diverse enough in order to accommodate people of all skin tones and sizes in all environments. A bias in data could possibly disadvantage deaf people of a certain ethnic group.

Accessibility

Ensuring that the model works equitably across different skin tones, lighting conditions, and backgrounds is essential to avoid discriminatory outcomes.

Error Rate

If the model has a high error rate, it could lead to misunderstandings and miscommunications, causing frustration and possibly severe consequences in critical situations.

Conclusion and Future Perspectives

The inspiration behind this work originated from the desire to address the language barriers faced by the hearing-impaired communities.
Through this project, we successfully developed a sign language translator capable of converting gestures into text and voice outputs, utilizing a custom convolutional neural network (CNN) model deployed on a Raspberry Pi. Our system leveraged the MediaPipe library for hand gesture detection and OpenCV for robust image processing, enabling real-time interpretation of sign language gestures. These interpreted gestures were then displayed as corresponding text in a user-friendly graphical interface created with Tkinter.

Our project not only showcases the efficacy of utilizing affordable and accessible hardware like the Raspberry Pi but also highlights the potential of assistive technologies to positively impact communication for individuals who are deaf or hard of hearing. However, while we have achieved significant milestones, there's still scope for further refinement to create a more efficient and practical application.

Moving forward, we aim to expand our project to detect and translate Tunisian Sign Language, which presents a greater complexity due to its extensive dataset and dynamic gestures involving movement, not just static images. This will involve collecting and annotating a large dataset of Tunisian Sign Language gestures, as well as enhancing our model to handle the increased complexity and variability of these gestures.

Bibliography

[1] Forbes. Deafness statistics, 2024. Accessed: 2024-05-24.

[2] World Health Organization. Deafness and hearing loss, 2024. Accessed: 2024-06-01.

[3] Databricks. Machine learning models, 2024. Accessed: 2024-06-01.

[4] DataCamp. Introduction to convolutional neural networks (cnns), 2024. Accessed: 2024-05-30.

[5] Hackerboards. Board technical specifications comparison, 2022. Accessed: 2024-05-30.

[6] All3dp. raspberry pi vs jetson nano differences, 2024. Accessed: 2024-05-30.

[7] Raspberry Pi. Raspberry pi 3 model b, 2024. Accessed: 2024-05-23.

[8] Sebastian Raschka. *Python machine learning*. Packt publishing ltd, 2015.

[9] learnopencv. Introduction to mediapipe, 2024. Accessed: 2024-05-26.

[10] Lemagit. Tensorflow, 2024. Accessed: 2024-05-26.

[11] TensorFlow. Tensorflow lite, 2024. Accessed: 2024-05-23.