# TINYML BASED SIGN LANGUAGE INTERPRETER FOR MEDICAL CONSULTATION

PROJECT REPORT

Submitted by

**AISWARYA MS** (TVE20CS009)

**ANAMIKA S RAJ** (TVE20CS018)

**NIRUPAMA TS** (TVE20CS084)

**SNEHA SANI** (TVE20CS108)

to

The APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

**Bachelor of Technology**

in

Computer Science and Engineering

**Department of Computer Science and Engineering**

**College of Engineering Trivandrum**

Kerala

May 1, 2024

# DECLARATION

We undersigned hereby declare that the project report **TinyML based sign language interpreter for medical consultation**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Dr Ajeesh Ramanujan** . This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

**Place**: Thiruvananthapuram

**Date**: May 1, 2024

Aiswarya MS

Anamika S Raj

Nirupama TS

Sneha Sani

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**COLLEGE OF ENGINEERING TRIVANDRUM**

## CERTIFICATE

This is to certify that the report entitled **"TinyML based sign language interpreter for medical consultation",** submitted by **Aiswarya MS**, **Anamika S Raj**, **Nirupama TS**, **Sneha Sani** to the **APJ Abdul Kalam Technological University** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

| | | |
|---|---|---|
| **Dr. Ajeesh Ramanujan** | **Dr. Piyoosh P** | **Dr.Salim A** |
| Associate Professor | Assisstant Professor | Professor |
| Department of CSE | Department of CSE | Department of CSE |
| (Guide) | (Coordinator) | (Head of Department) |

## ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude and heartful indebtedness to our guide **Dr. Ajeesh Ramanujan, Associate Professor**, Department of Computer Science and Engineering for his valuable guidance and encouragement in pursuing this project.
We are also thankful to **Dr.Salim.A** , Head of the Department for his help and support.
We also extend our hearty gratitude to project co-ordinator, **Dr. Piyoosh P**, College of Engineering Trivandrum for providing necessary facilities and his sincere co-operation.
Our sincere thanks is extended to all the teachers of the Department of Computer Science and Engineering and to all my friends for their help and support. Above all, we thank God for the immense grace and blessings at all stages of the project.

Aiswarya MS

Anamika S Raj

Nirupama TS

Sneha Sani

# ABSTRACT

Sign Language is mainly used by mute community to exchange information between their own community and with other people. It is a language where people use their hand gestures to communicate as they can't speak or hear. In this project, we aim to implement a TinyML model for doctor consultations with sign language users. The system recognizes the hand gestures and then convert them into text so that the health care provider can easily understand. To enable two-way communication the doctor's speech is converted to text using a speech-to-text module. The proposed ML model will be designed to operate on a single-board computer, equipped with a camera to capture the user's sign language gestures. This can be done using edge computing techniques, where the model is deployed on a device at the edge of the network rather than in the cloud. The system will be trained using a dataset of American Sign Language gestures and validated against human translations. Model Optimization, Platform Selection, Framework Selection and Model Deployment are the high-level steps followed to implement the ML model on an IOT device. The project's primary objective is to develop and deploy an efficient and accurate ML model to an IOT device that can enable effective communication between sign language users and doctors during medical consultations. The final product will be an easy-to-use system that can help bridge the communication gap and improve health care outcomes for sign language users..

# Contents

# List of Figures

# List of Tables

# ABBREVIATIONS

(List in the alphabetical order)

| | |
|---|---|
| **IoT** | Internet of Things |
| **ML** | Machine Learning |
| **KB** | Kilobytes |
| **MB** | Megabytes |
| **ML** | Machine Learning |
| **TF** | TensorFlow |
| **TFlite** | TensorFlow Lite |
| **TinyML** | Tiny Machine Learning |

# Chapter 1

# Introduction

Communication is fundamental to healthcare, enabling effective interactions between healthcare providers and patients. However, for individuals with hearing impairments, the barrier to communication can be substantial. Sign language serves as a crucial medium of communication for the deaf and hard-of-hearing community, allowing them to express themselves and understand information. In the context of medical consultations, bridging this communication gap is paramount to ensure comprehensive healthcare delivery. This project introduces a groundbreaking solution leveraging Tiny Machine Learning (TinyML)to create a Sign Language Interpreter tailored for medical consultations. Traditional interpreters may not always be readily available, leading to delays and potential misunderstandings in conveying critical healthcare information. By incorporating TinyML into the interpreter, we aim to provide a portable, efficient, and real-time solution that enhances accessibility for individuals with hearing impairments during medical interactions. The integration of TinyML in the realm of sign language interpretation for medical consultations holds immense promise. This project addresses the critical need for enhanced accessibility in healthcare, promoting inclusivity and patient-centered care. By leveraging the power of machine learning on edge devices, we not only empower individuals with hearing impairments but also contribute to the broader advancement of assistive technologies in the healthcare domain.

# Chapter 2

# Existing Methods

## 2.1 Review of Literature

The literature review section of this project aims to provide an overview of existing approaches to TinyML-based sign language interpretation, with a particular focus on systems designed for medical consultations. This will include an examination of their methodologies, strengths, and limitations, setting the foundation for understanding the unique benefits of the proposed solution.

### 2.1.1 Sign language recognition and translation method based on VTN(1)

The paper introduces a novel approach to sign language recognition and translation using a Video Transformer Net (VTN). The researchers developed a lightweight network model to improve real-time sign language translation and communication for deaf individuals. The method involves training isolated sign language recognition and continuous sign language translation frameworks using the CSL BS dataset, which includes data from the "National Common Sign Language Common Vocabulary List" and integrates the public CSL dataset.The proposed method shows improvements in recognition accuracy and speed compared to I3D-based approaches.

The paper mentions shortcomings in the recognition of similar sign language actions, indicating a limitation in distinguishing between closely related gestures.The extraction of long-sequence key frames is identified as an area needing improvement, suggesting a challenge in capturing detailed information from extended sign language sequences.

### 2.1.2 TinyML: Current progress, research challenges, and future roadmap.(2)

The paper delves into the burgeoning field of tiny machine learning (TinyML), which encompasses machine learning technologies and applications designed to perform sensor data analytics on low-power devices. This rapidly evolving domain is paving the way for on-device processing, enabling always-on functionalities in battery-operated devices. The document outlines the current advancements, challenges, and research opportunities in TinyML, emphasizing the significance of benchmarking methodologies and the emergence of Edge-AI applications facilitated by TinyML technology. Furthermore, the paper underscores the momentum in technical advancements, ecosystem development, and the need for robust evaluation frameworks in the realm of TinyML.

Moreover, the paper underscores the cross-layer design flow in TinyML, showcasing the integration of hardware and software optimizations to accelerate deep neural network (DNN) inference. Researchers have proposed specialized accelerators at the hardware level for efficient data processing, while software-level techniques such as Neural Architecture Search (NAS), pruning, and quantization have been introduced to craft compact and efficient models suitable for resource-constrained Internet of Things (IoT) devices. By synergistically connecting various optimization techniques, the cross-layer design flow presented in the paper aims to build ultra-efficient DNN-based systems, reflecting the interdisciplinary nature of TinyML research and development.

### 2.1.3 Survey on implementation of TinyML for real-time sign language recognition using smart gloves.(3)

The paper explores the utilization of TinyML technology in conjunction with smart gloves for real-time recognition of sign language gestures. The study aims to enhance communication for individuals who are deaf and dumb by developing a system that can accurately interpret sign language gestures and convert them into text or speech. By leveraging machine learning algorithms and hardware components, the proposed system offers a portable and efficient solution for bridging the communication gap between individuals with hearing or speech impairments and those without.

One limitation of the proposed model is the accuracy of the recognition system, as it may not always correctly interpret all sign language gestures, leading to potential miscommunication. Additionally, the reliance on hardware components and machine learning algorithms introduces complexities in maintenance and troubleshooting, especially in real-world scenarios where environmental factors such as lighting conditions or background noise can impact the system's performance.

### 2.1.4 Gesture-based sign language recognition system.(4)

The paper presents a novel approach to enhancing communication for individuals with hearing or speech impairments by utilizing hand gestures for sign language recognition. The system incorporates cutting-edge technologies such as MobileNet, Xception, ResNet 101, and DenseNet 121 to achieve high accuracy in recognizing sign language gestures. Additionally, the system integrates text-to-speech technology and a Telegram bot for real-time sign language detection and communication, providing a user-friendly interface for both individuals with impairments and those without.

One drawback of the proposed system is the reliance on complex deep learning models, which requires significant computational resources and expertise to train and deploy effectively. Additionally, the real-time detection using a web camera introduces latency issues, impacting the system's responsiveness and usability in certain scenarios. Addressing these challenges through optimization and simplification of the system's architecture would enhance its practicality and accessibility for a wider range of users.

### 2.1.5 Deaf and mute sign language translator on static alphabets gestures.(5)

The paper presents a vision-based approach for recognizing static American Sign Language alphabet gestures using MobileNet, a lightweight convolutional neural network architecture. By focusing on static gestures and utilizing efficient depthwise separable convolutions, the model achieves a high accuracy of 99.93 % with a low standard deviation of 0.3 %. The dataset used for training and testing consists of 27,455 training images and 7,172 testing images, sourced from the Kaggle ASL Letter dataset. The proposed system demonstrates the potential for real-time sign language translation, offering a cost-effective and efficient solution for communication between deaf/mute individuals and non-signers.

However, despite its merits, the proposed system also has limitations. One disadvantage is the limited application scope of MobileNet due to its small size and depthwise convolution architecture. This may result in reduced accuracy when dealing with complex models or datasets. Additionally, the system's reliance on static gestures excludes dynamic gestures, which are also integral to sign language communication. Future work could focus on addressing these limitations by incorporating dynamic gestures and enhancing the model's performance on more diverse datasets.

### 2.1.6 A cost-effective design and implementation of Arduino-based sign language interpreter.(6)

The paper presents a novel approach to converting sign language gestures into speech using Arduino-based technology. The system utilizes Bluetooth-enabled gloves equipped with sensors to capture hand movements, enabling the recognition of finger spelling for effective communication. By leveraging the ATMEGA32 microcontroller and Arduino Leonardo board, the device processes the captured gestures and translates them into speech, thereby facilitating communication between individuals with voice disabilities and the general population. This cost-effective solution offers a promising avenue for enhancing the communication abilities of those with listening impairments.

One potential drawback of the system is the accuracy and reliability of the finger spelling recognition, as variations in hand movements or sensor malfunctions lead to errors in translation. Additionally, the dependency on Bluetooth connectivity for communication poses challenges in environments with high interference or limited range, affecting the real-time effectiveness of the system. Addressing these limitations through further research and development could enhance the overall performance and usability of the Arduino-based sign language interpreter.

### 2.1.7 Classification of sign language using MobileNet-deep learning.(7)

The paper delves into the utilization of deep learning techniques, particularly MobileNet, for the classification of sign language. It evaluates the performance of the pretrained MobileNet model on a dataset comprising 29 classes of sign language signs, achieving an accuracy rate of 95.41 % after 20 epochs. The study also compares the results with a previous paper that used VGG16, which attained a higher accuracy of 100 percent. The research highlights the potential of MobileNet in effectively predicting and classifying different sign languages, emphasizing its significance for enhancing accessibility and user-friendly access in sign language recognition systems.

However, a notable disadvantage of the paper lies in its limited exploration of the scalability and generalizability of the MobileNet algorithm in practical sign language recognition applications. Further discussion on the adaptability of MobileNet to diverse sign language datasets and real-world scenarios could enhance the paper's relevance and applicability. Additionally, while the comparison with VGG16 provides valuable insights into different deep learning models, a more comprehensive analysis of the trade-offs between accuracy, computational efficiency, and model complexity would offer a deeper understanding of the strengths and limitations of each approach.

### 2.1.8 A sign language recognition system applied to deaf-mute medical consultation.(9)

The paper presents the development of the Heart-Speaker system, a novel device designed to facilitate sign language recognition for deaf individuals during medical consultations. The system integrates real-time and fast machine vision technology, utilizing a MobileNet-YOLOv3 network model for efficient sign language image recognition. The compact size of the device, comparable to a mobile phone, enhances portability and user experience. Additionally, the system's innovative design features, such as external sensing of the radar module and a 2.4-inch LCD display, demonstrate a thoughtful approach to addressing the communication barriers faced by the deaf community in healthcare settings.

However, despite the promising advancements showcased in the paper, there are certain limitations and disadvantages that warrant consideration. One notable drawback is the reliance on complex machine vision algorithms and deep learning models, which may pose challenges in terms of computational resources and processing power, especially on low-power embedded platforms. Additionally, the paper does not extensively discuss the scalability and generalizability of the Heart-Speaker system beyond sign language recognition, raising concerns about its adaptability to diverse communication contexts and potential limitations in real-world applications. Further research and development efforts may be needed to address these drawbacks and enhance the overall effectiveness and usability of the system.

## 2.2 MOTIVATION

The current methods for sign language interpretation have a number of limitations as discussed above.These limitations led us to the following motivation:

- Deployment on Edge Devices: Utilizing TinyML on edge devices allows for real-time processing and privacy preservation, as data is handled directly on the patient's or doctor's device without reliance on the cloud.
- Ease of Use: The system is designed to be lightweight, low-power, and user-friendly, making it accessible for a wide range of users and ensuring efficient operation.
- Real-world Testing: The interpreter is evaluated in medical consultation settings to optimize its performance for practical use, addressing real-world challenges and opportunities.

# Chapter 3

# Problem Statement and Objectives

Individuals who are deaf or hard-of-hearing often encounter significant communication obstacles within the healthcare system. These barriers can lead to misdiagnosis, delayed treatment, and heightened anxiety, ultimately compromising the quality of care. This project tackles this critical issue by developing a TinyML-based sign language interpreter tailored specifically for the medical consultation setting. The overarching goal is to bridge this communication gap and promote inclusivity by providing healthcare providers with an accessible and efficient tool for understanding patients who rely on sign language. To achieve this, the project will encompass several key steps: meticulous collection and preprocessing of a comprehensive sign language dataset, development of an accurate and resource-efficient machine learning model, optimization techniques for edge device deployment, and rigorous testing and evaluation in realistic medical environments

## 3.1 Problem Statement

To develop a lightweight, low-power, easy-to-use system that allows healthcare providers to communicate with mute or hard-of-hearing patients.

## 3.2 Objectives

The project is centered around the creation of a TinyML-based sign language interpreter for medical consultations. The overarching goal is to facilitate seamless communication between healthcare providers and patients who are mute or hard-of-hearing. To this end, the project is divided into two distinct yet interconnected phases, each with its own set of objectives.

During the first phase, the emphasis is on assembling a robust dataset of American Sign Language (ASL) gestures. This involves not only the collection of a wide array of ASL signs but also the meticulous preprocessing of this data to ensure it is of high quality and suitable for model training. The development of the machine learning model follows, with a focus on achieving high accuracy in translating sign language gestures into text. This model must be efficient enough to operate on an edge device, making it accessible and portable for use in various medical environments.

The second phase is dedicated to refining the machine learning model to meet the constraints of an IoT device. The aim is to minimize the model's size without sacrificing its accuracy. This will involve exploring and applying different optimization techniques to create a version of the model that is both compact and effective. The subsequent conversion of the model to a format that is compatible with the IoT device is crucial, as is the selection of the IoT device itself. The chosen device must support the model's requirements while remaining energy-efficient and cost-effective.

Finally, the deployment of the model onto the IoT device marks the culmination of the project's development stage. The system will then undergo extensive testing and evaluation within a medical consultation setting to ensure it meets the necessary standards of performance, usability, and reliability. The successful completion of these objectives will significantly enhance the quality of patient care and the overall consultation experience for individuals who communicate through sign language.

# Chapter 4

# Design and Implementation

In this chapter, we will explore the meticulous process of designing and implementing the project, from model selection to development, optimization, and deployment. We'll explore the methods for choosing the most appropriate model for the task, developing it effectively, optimizing its size without sacrificing accuracy, and finally deploying it onto a resource-constrained IoT device.

In this section, we will delve into the design principles and implementation that were used in our project and also discuss the strengths and weaknesses of our platform.

## 4.1 SOFTWARE REQUIREMENTS SPECIFICATION

### 4.1.1 Introduction

#### 4.1.1.1 Purpose

The project's primary objective is to develop an efficient and accurate ML model and deploy it to an IOT device that can enable effective communication between sign language users and medical professionals during medical consultations.

#### 4.1.1.2 Intended Audience

The project is intended to be used by the mute community to have effective communication with medical professionals regarding their illnesses and difficulties.

#### 4.1.1.3 Project Scope

Sign Language is mainly used by the mute community to exchange information between their community and with other people. It is a language where people use their hand gestures to communicate as they can't speak or hear. But these people face difficulties in communicating with ordinary people as they can't understand sign language. These difficulties are more prominent in the medical field. This project aims to implement a machine learning (ML) model and then deploy it into an edge device for medical consultation with sign language users.

### 4.1.2 System Workflow

Figure 1.1 (Entire workflow) is the overview of the project. The model uses sign language from the patients and converts it into text which is then used by the medical professionals.

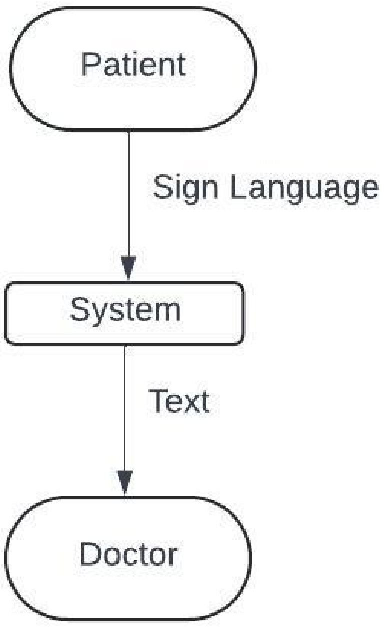


**Figure 4.1:** Entire work-flow

### 4.1.3 System Overview

The TinyML-based Sign Language Interpreter is a system designed to help patients with hearing or speech impairments communicate more effectively with medical professionals. The system uses machine learning algorithms to interpret sign language gestures made by the patient and convert them into text that can be understood by the medical professional. The system will use computer vision to capture sign language gestures and convert them into text, allowing sign language users to communicate effectively with their medical professionals. Furthermore, a comprehensive module for doctor-to-patient communication is implemented, leveraging speech recognition to convert the doctor's speech into text. To enhance the communication experience, relevant images of the disease or medical conditions are displayed alongside the translated text, providing visual context and facilitating a better understanding

### 4.1.4 System Requirements

#### 4.1.4.1 Functional Requirements

The following are the functional requirements of the system:

- Sign Language Interpretation: The system should be able to accurately interpret the signs made by the patients and convert them into text. The system should be trained using a pre-existing dataset of sign language gestures.
- Real-time Processing: The system should be able to work in real time and be able to process the data on the edge device without requiring a connection to the internet.

#### 4.1.4.2 Non Functional Requirements

The following are the nonfunctional requirements of the system:

- Low computation power: The developed ML model should be deployable on low-power devices.
- Offline Functionality: The system should work without using the Internet.
- Accuracy: The system should be able to accurately interpret the signs made by the patients.
- Performance: The system should be able to process data in real time without any lag or delay.

### 4.1.5 Software and Hardware Requirements

#### 4.1.5.1 Software Requirements

The following are the software requirements of the system:

- Machine Learning model: A Convolution Neural Network based model ie Mobilenet is developed using Tensorflow, Keras, Numpy, and scikit-learn libraries in Python.
- Computer Vision Libraries: OpenCV is used to capture sign language gestures.

- TinyML Framework: We choose a TinyML-compatible framework, such as TensorFlow Lite to deploy the above ML model into an edge device.
- Model Optimization Tools: This may include techniques such as Knowledge Distillation and Quantization.
- Deployment:We need Raspberry Pi Imager and Raspberry Pi OS (64 bit) installed to deploy the ML model into Raspberry Pi.

#### 4.1.5.2 Hardware Requirements

The following are the hardware requirements of the system:

- Raspberry Pi 4 Model B with 1 GB RAM
- MicroSD card (minimum of 8GB)
- Webcam
- Monitor

## 4.2 SOFTWARE DESIGN DOCUMENT

### 4.2.1 Introduction

#### 4.2.1.1 Purpose

This software design document describes the architecture and system design of the TinyML based sign language interpreter to establish an efficient communication between the sign language users and medical professionals.

### 4.2.2 System Architecture Design

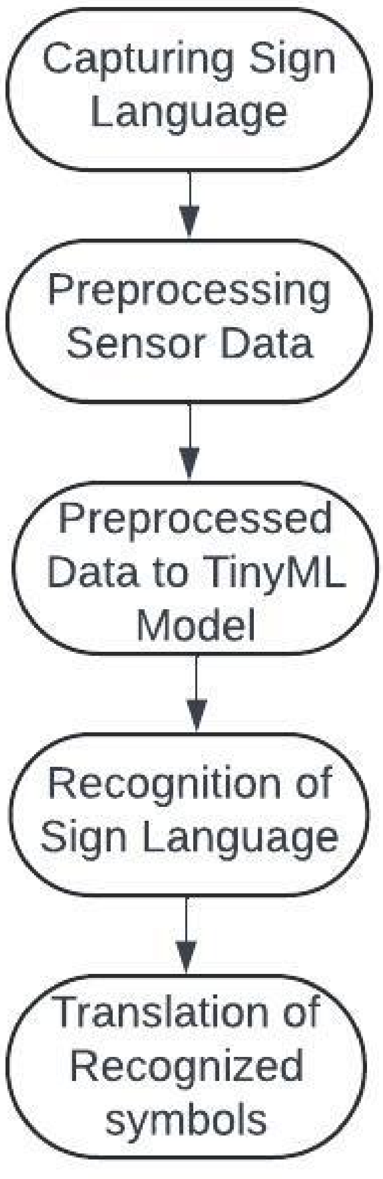

Capturing Sign Language
Preprocessing Sensor Data
Preprocessed Data to TinyML Model
Recognition of Sign Language
Translation of Recognized symbols

**Figure 4.2:** Activity diagram

#### 4.2.2.1 Capturing Sign Language

Capturing sign language gestures using a camera involves the use of technology to detect and recognize hand gestures made by sign language users. Here we use computer vision techniques to process image input from a camera, which can detect and track hand movements in real time.

#### 4.2.2.2 Preprocessing Captured Data

Preprocessing captured data is an essential step in developing machine-learning models for TinyML applications. Preprocessing involves transforming raw captured data into a suitable format that can be used by machine learning algorithms. By preprocessing captured data effectively, our models can achieve higher accuracy, improved performance, and lower power consumption, making them ideal for edge computing applications.

#### 4.2.2.3 Input Preprocessed Data to TinyML Model

After preprocessing the captured data, the next step is to input the data into the TinyML model for inference. TinyML models typically require input data to be in a specific format, such as a tensor, which is a multidimensional array of values. Once the input data is in the correct format, it can be passed through the layers of the TinyML model to generate a prediction or classification.

#### 4.2.2.4 Recognition of Sign Language

During the consultation, the patient could use sign language to communicate with the doctor, and the TinyML model could recognize the gestures in real time, providing a text or speech translation that the doctor can understand. This would enable effective communication between the patient and the doctor, without the need for an interpreter, and could help to improve the quality of healthcare for patients with speech and hearing disabilities.

#### 4.2.2.5 Translation of Recognized Symbols

After recognizing sign language gestures using a TinyML model, the next step is to translate the recognized symbols into text.

### 4.2.3 Technology Stack

- Machine Learning model: Tensorflow, Keras, Numpy, and scikit-learn libraries in Python.
- Computer Vision Libraries: OpenCV
- TinyML Framework: TensorFlow Lite

### 4.2.4 Application Architectural Design

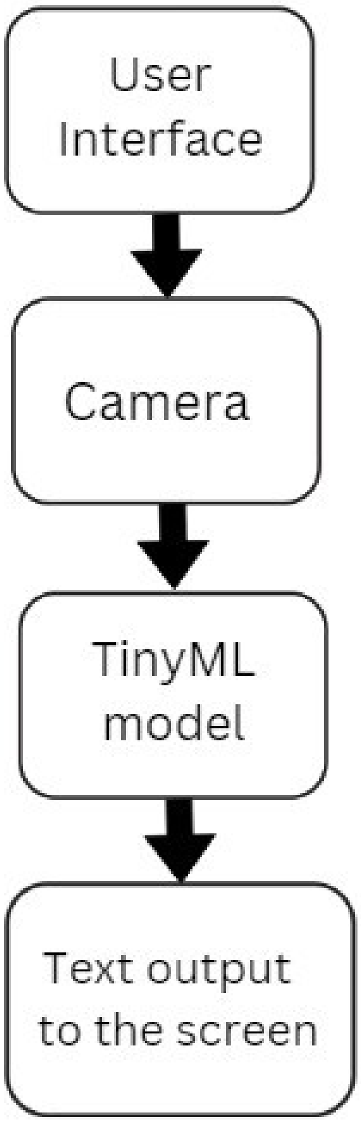


**Figure 4.3:** Component Level Design

- User Interface:allows users to input sign language gestures and view the translated text output on a PC screen.
- Camera: Used to capture the sign language gestures made by the user, which are then processed by the tinyML model
- TinyML model: A machine learning model deployed on a PC that processes sign language gestures from the camera feed and translates them into text.
- Text output to the screen: displays the translated text output on the PC screen for the user to read.

## 4.3 DESIGN

Figure 4.1 shows the proposed design diagram

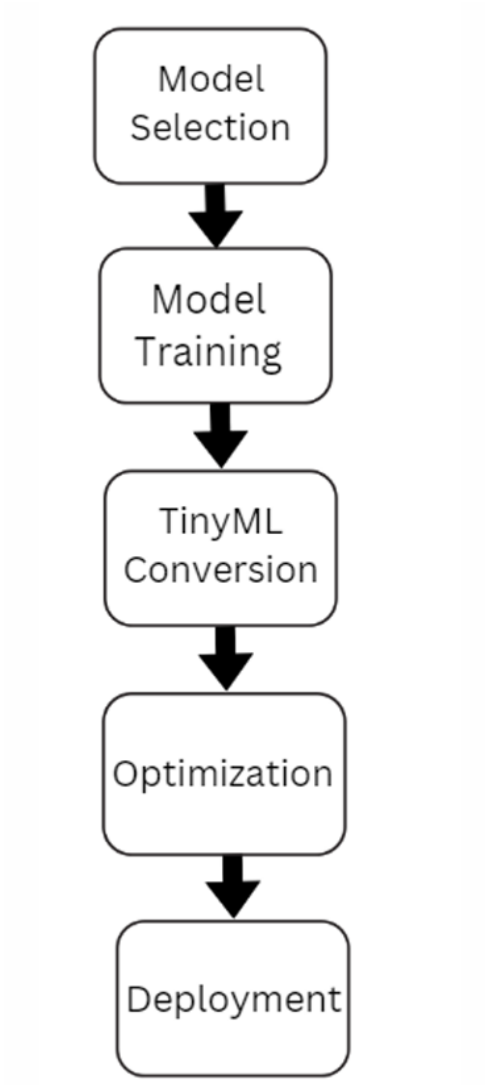


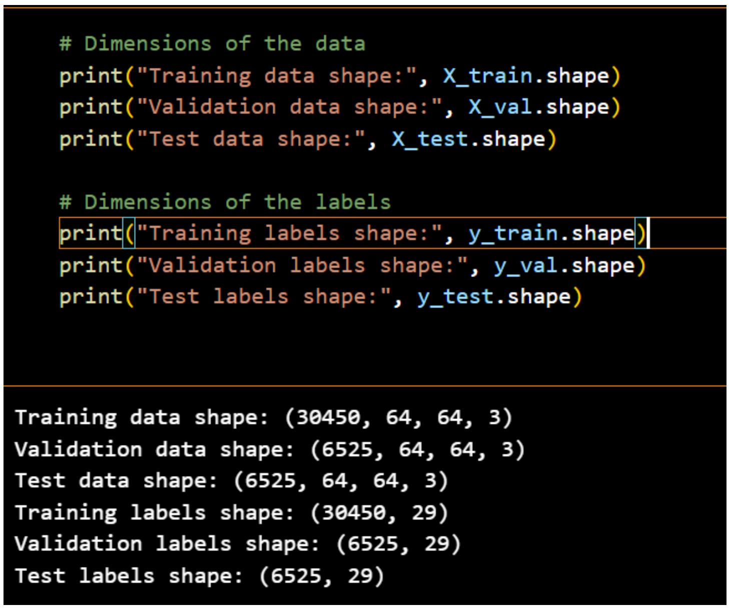
**Figure 4.4:** Design Diagram

### 4.3.1 Dataset

The proposed Sign Language Recognition System leverages a dataset of American Sign Language (ASL) images sourced from Kaggle. This dataset comprises 87,000 JPEG images, each with dimensions of 200x200 pixels, representing 29 distinct classes. These classes encompass the 26 letters of the alphabet from A to Z, along with three additional classes denoting SPACE, DELETE, and NOTHING. The inclusion of these three supplementary classes enhances the system's utility in real-time applications and classification tasks. To mitigate the computational burden, the decision was made to utilize only 43,500 images from the original dataset. This subset retains the 29 classes, ensuring comprehensive coverage of ASL gestures while effectively reducing the computational load.

#### 4.3.1.1 Datasplit

To accurately train and evaluate the model, the dataset will be divided into three sets: a training set (70%), a validation set (15%), and a testing set (15%).



```
# Dimensions of the data
print("Training data shape:", X_train.shape)
print("Validation data shape:", X_val.shape)
print("Test data shape:", X_test.shape)

# Dimensions of the labels
print("Training labels shape:", y_train.shape)
print("Validation labels shape:", y_val.shape)
print("Test labels shape:", y_test.shape)
```

```
Training data shape: (30450, 64, 64, 3)
Validation data shape: (6525, 64, 64, 3)
Test data shape: (6525, 64, 64, 3)
Training labels shape: (30450, 29)
Validation labels shape: (6525, 29)
Test labels shape: (6525, 29)
```

**Figure 4.5:** Dataset-Split

### 4.3.2 Model Selection and Model Training

After conducting thorough evaluations on four different models—namely YOLO V5, Inception, CNN, and MobileNet—it became evident that MobileNet exhibited the most promising performance for gesture recognition tasks. Despite having a significantly lower number of parameters compared to the other models, with only 3.25 million parameters as per Figure 5.1, MobileNet achieved an exceptional accuracy rate of 99.04%. Additionally, its inference time of 0.1400 seconds showcased its efficiency in real-time applications.

The MobileNet model's superior performance can be attributed to its efficient architecture, which strikes a balance between accuracy and computational efficiency. By utilizing depthwise separable convolutions, MobileNet reduces the computational complexity without compromising on accuracy. This design choice makes it particularly well-suited for resource-constrained devices like the Raspberry Pi, where computational resources are limited.

Furthermore, MobileNet's favorable resource requirements make it an attractive option for deployment in practical scenarios. Its relatively low memory footprint and computational demands ensure smooth operation on devices with limited processing power and memory capacity.

In conclusion, MobileNet emerges as the optimal choice(Table 5.1) for gesture recognition tasks due to its impressive accuracy, efficient architecture, and suitability for deployment on resource-constrained devices. Its robust performance makes it a reliable solution for real-world applications requiring gesture recognition capabilities.

### 4.3.3 Tiny-ML Conversion

The decision to adopt TinyML for our project was driven by its exceptional capabilities in delivering real-time, low-latency, and privacy-enhanced machine learning on edge devices. By leveraging TinyML, we can ensure cost-effective and offline performance, making our system ideal for deployment in resource-constrained environments. Furthermore, our choice of TensorFlow Lite as the framework for our TinyML model was deliberate. TensorFlow Lite is a specialized version of TensorFlow tailored specifically for running lightweight machine-learning models on small, low-power devices. Its optimized architecture enables efficient execution of our model on edge devices, without compromising on accuracy or performance. This strategic combination of TinyML and TensorFlow Lite equips our system with the necessary tools to achieve our objectives effectively and efficiently.

### 4.3.4 Optimization

In pursuit of enhancing the efficiency and deployability of our chosen MobileNet model, which initially occupied a substantial size of 37MB, we have implemented a series of optimization techniques. These techniques, include knowledge distillation and quantization, which are meticulously employed to reduce the model's size while preserving its accuracy. By delving into these optimization strategies, we have streamlined the model's footprint without compromising its ability to deliver accurate results, thus facilitating its integration into resource-constrained environments and enhancing overall system performance.

#### 4.3.4.1 Knowledge distillation

Knowledge distillation, a powerful optimization technique, plays a pivotal role in transforming a large "teacher" model into a more compact "student" model tailored for deployment on edge devices. In this process, the student model learns from the teacher model's soft labels, which encapsulate the rich knowledge and insights gained from the teacher's extensive training. By distilling this knowledge into a smaller and more efficient form, knowledge distillation enables us to significantly reduce the size of the student model while retaining its ability to make accurate predictions. However, it is important to note that this reduction in size may come at a slight cost to the model's accuracy. In our case, employing knowledge distillation resulted in the creation of a student model with a substantially reduced size of 12.6MB, compared to the initial size of 37MB (refer to Table 5.2). Additionally, while the model's accuracy decreased from 99% to 95%, and the inference time was reduced from 0.14 to 0.12 seconds. Despite these minor trade-offs, the reduction in size and inference time achieved through knowledge distillation significantly enhances the model's deployability on resource-constrained devices, contributing to improved system performance and efficiency. The incorporation of knowledge distillation underscores our commitment to leveraging advanced optimization techniques to maximize the effectiveness and practicality of our Sign Language Recognition System.

#### 4.3.4.2 Quantization and tflite conversion

Quantization, an essential optimization technique, involves reducing the bit precision of numerical data within machine learning models. By employing quantization, we aim to decrease the model's memory footprint and computational requirements while preserving its accuracy to the greatest extent possible. In our approach, we utilized dynamic range quantization, which involves converting numerical data from float 32-bit precision to 8-bit precision. This reduction in precision allows for more efficient storage and computation of model parameters, thereby facilitating deployment on resource-constrained edge devices.

Additionally, TensorFlow Lite conversion played a crucial role in the optimization process. TensorFlow Lite conversion transforms the optimized MobileNet model into a format compatible with TensorFlow Lite, a specialized framework designed for running lightweight machine-learning models on small, low-power devices. This conversion process ensures seamless integration of the optimized model into the target deployment environment, enabling efficient execution and inference on edge devices.

Following the application of quantization and TensorFlow Lite conversion, significant improvements were observed in the model's size, accuracy, and inference time. The model's size was notably reduced to 3.25MB(refer to Table 5.2), representing a substantial decrease from the previous size of 12.6MB. While there was a slight reduction in accuracy from 95% to 91%, the model's inference time was dramatically improved, dropping from 0.12 seconds to a mere 0.02 seconds. These enhancements highlight the effectiveness of the optimization techniques employed in enhancing the efficiency and deployability of the Sign Language Recognition System. By incorporating quantization and TensorFlow Lite conversion, we have successfully transformed the MobileNet model into a highly optimized and lightweight solution, suitable for deployment on resource-constrained edge devices.

### 4.3.5 Deployment

Following the optimization process, the highly refined MobileNet model was seamlessly deployed onto a targeted edge device, specifically the Raspberry Pi 4 Model B with 1 GB of RAM. This device was carefully selected for its compatibility with the requirements of our Sign Language Recognition System, which necessitates support for Python, TensorFlow Lite, OpenCV, and relevant libraries essential for model execution and inference.

The Raspberry Pi 4 Model B's 1 GB of RAM provides adequate resources for running the optimized MobileNet model efficiently, ensuring smooth operation without compromising performance. Additionally, the device's support for Python, TensorFlow Lite, and OpenCV, along with other essential libraries, enables seamless integration of the optimized model into the device's environment.

By deploying the optimized MobileNet model onto the Raspberry Pi 4 Model B, we are able to harness the power of edge computing, bringing real-time gesture recognition capabilities directly to the user's fingertips. This deployment represents a significant milestone in the development of our Sign Language Recognition System, as it demonstrates the feasibility and effectiveness of deploying sophisticated machine learning models on resource-constrained edge devices for practical applications.

## 4.4 STRENGTHS

- **Optimized Model**: The successful reduction of the MobileNet model size from 37.47MB to 3.25MB demonstrates the effectiveness of our optimization techniques, making it suitable for deployment on resource-constrained edge devices.

- **Practical Application**: The Sign Language Recognition System addresses a real-world need by facilitating communication between healthcare professionals and patients with hearing impairments during medical consultations, thereby enhancing accessibility and inclusivity in healthcare settings.

- **Compatibility with Edge Devices**:The deployment of the system on the Raspberry Pi 4B underscores its compatibility with edge devices, enabling its integration into existing healthcare infrastructure with minimal additional hardware requirements.

- **Future Development Potential**: The identification of future work avenues, such as incorporating two-way communication features and exploring audio channels, provides a roadmap for further enhancing the system's capabilities and addressing emerging user needs.

## 4.5 WEAKNESSES

- **Reduction in Accuracy**: Despite optimization efforts, there was a slight reduction in model accuracy from 95% to 91%. While still acceptable for many applications, further improvements may be necessary to ensure robust performance in all scenarios.

# Chapter 5

# Results and Discussion

The results and discussion chapter of our project report delves into the outcomes and implications of our endeavor to develop a TinyML-based Sign Language Recognition System tailored for medical consultations. Throughout this chapter, we will analyze the performance, efficacy, and practical implications of our system, which is designed to facilitate communication between healthcare professionals and patients with hearing impairments. By examining the results of our model optimization efforts, deployment onto edge devices, and real-world testing scenarios, we aim to shed light on the viability and impact of our Sign Language Recognition System within the context of medical consultations.

## 5.1 Performance Metrics of Trained Models

| Model | Parameters | Accuracy | Inference time (s) |
|---|---|---|---|
| Inception | 22 Million | 94% | 0.173 |
| YOLO V5 | 7.08 Million | 92% | 0.93 |
| CNN | 5.59 Million | 94% | 0.134 |
| MobileNet | 3.25 Million | 99.04% | 0.1400 |

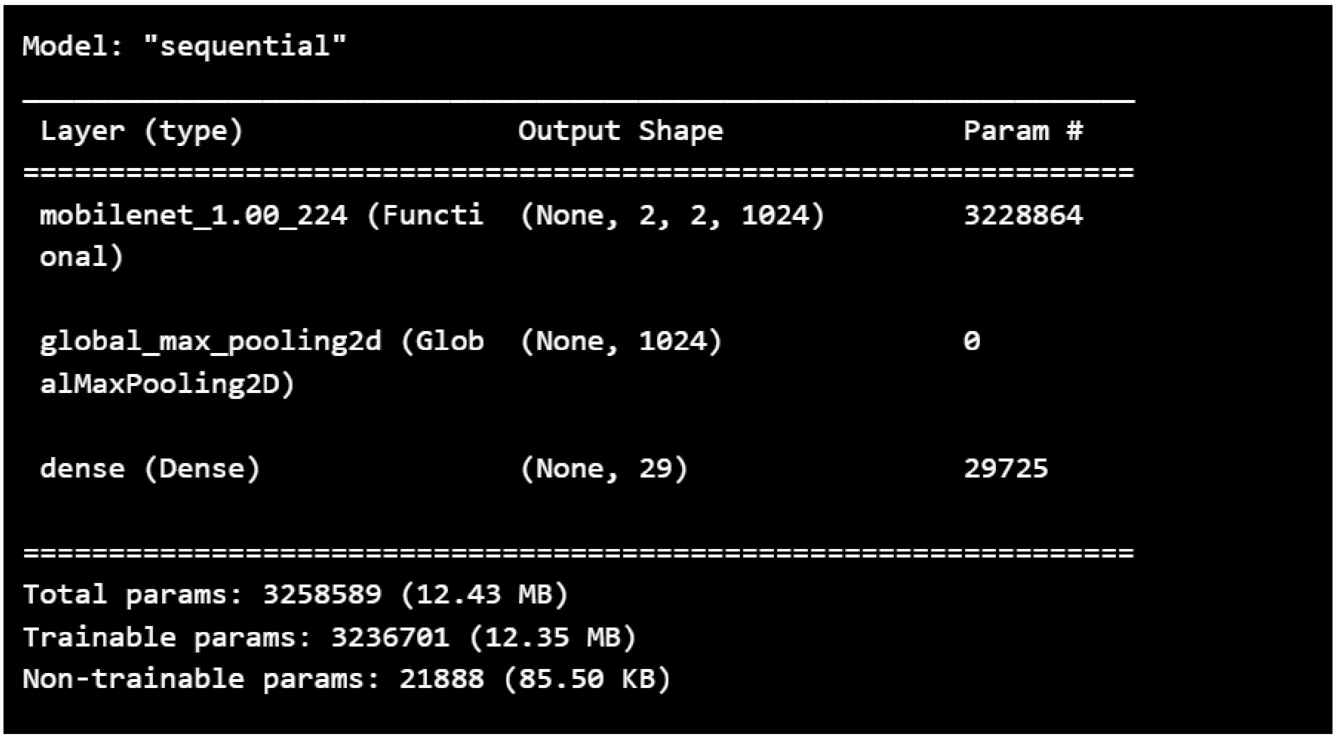
**Table 5.1:** Performance Comparison of Different Models

## 5.2 OPTIMIZATION TECHNIQUES-RESULT

| Model | Size | Accuracy | Inference time |
|---|---|---|---|
| Mobilenet | 37.47 MB | 99% | 0.14s |
| Knowledge distillation | 12.60 MB | 95% | 0.12s |
| Quantization | 3.25 MB | 91% | 0.02s |

**Table 5.2:** Optimization Results

## 5.3 MOBILENET-ARCHITECTURE



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenet_1.00_224 (Functi  (None, 2, 2, 1024)        3228864
 onal)

 global_max_pooling2d (Glob  (None, 1024)              0
 alMaxPooling2D)

 dense (Dense)               (None, 29)                29725

=================================================================
Total params: 3258589 (12.43 MB)
Trainable params: 3236701 (12.35 MB)
Non-trainable params: 21888 (85.50 KB)
```

**Figure 5.1:** Mobile-Net Architecture

## 5.4 MobileNet- Training and validation Accuracy

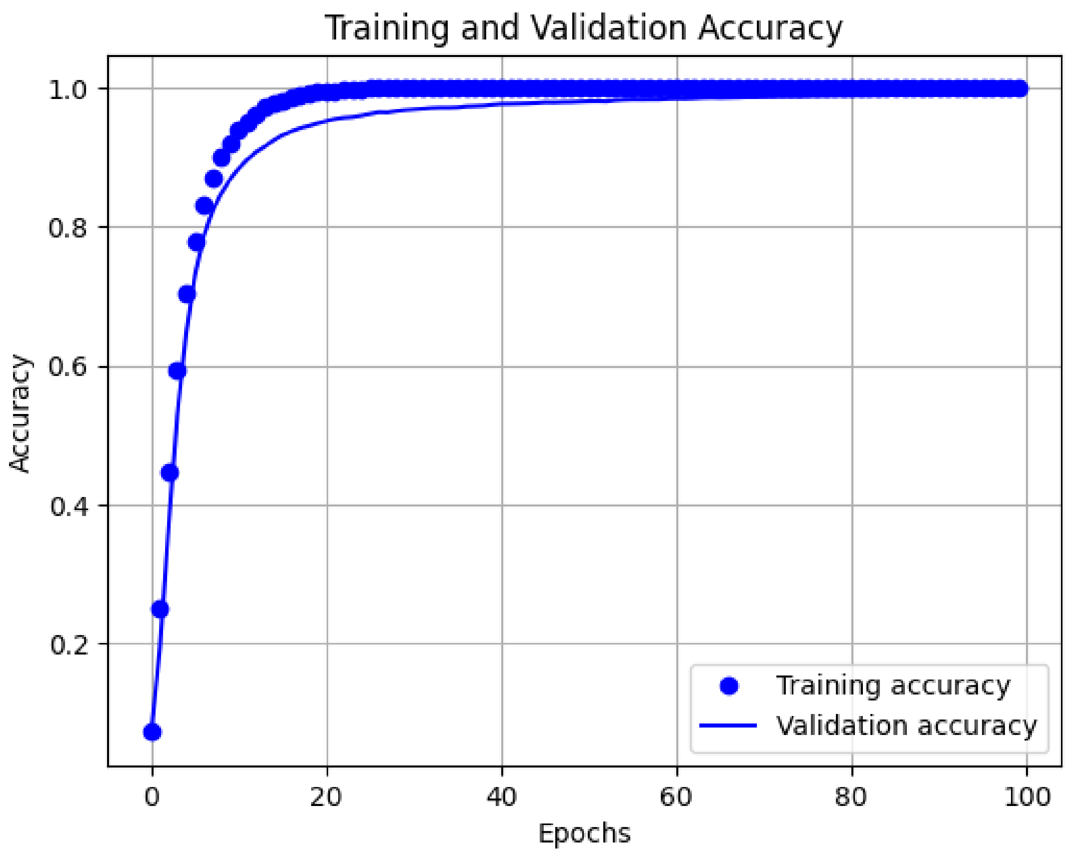


**Figure 5.2:** Accuracy vs epoch

## 5.5 MobileNet- Training and validation Loss

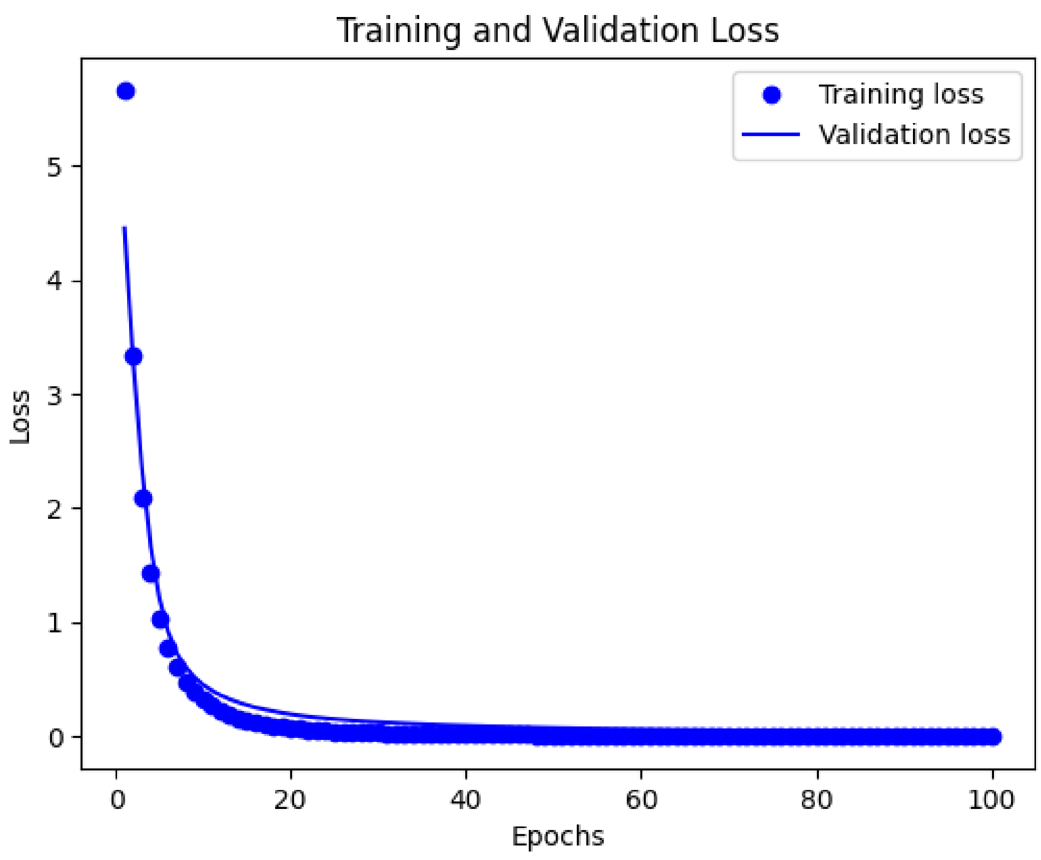
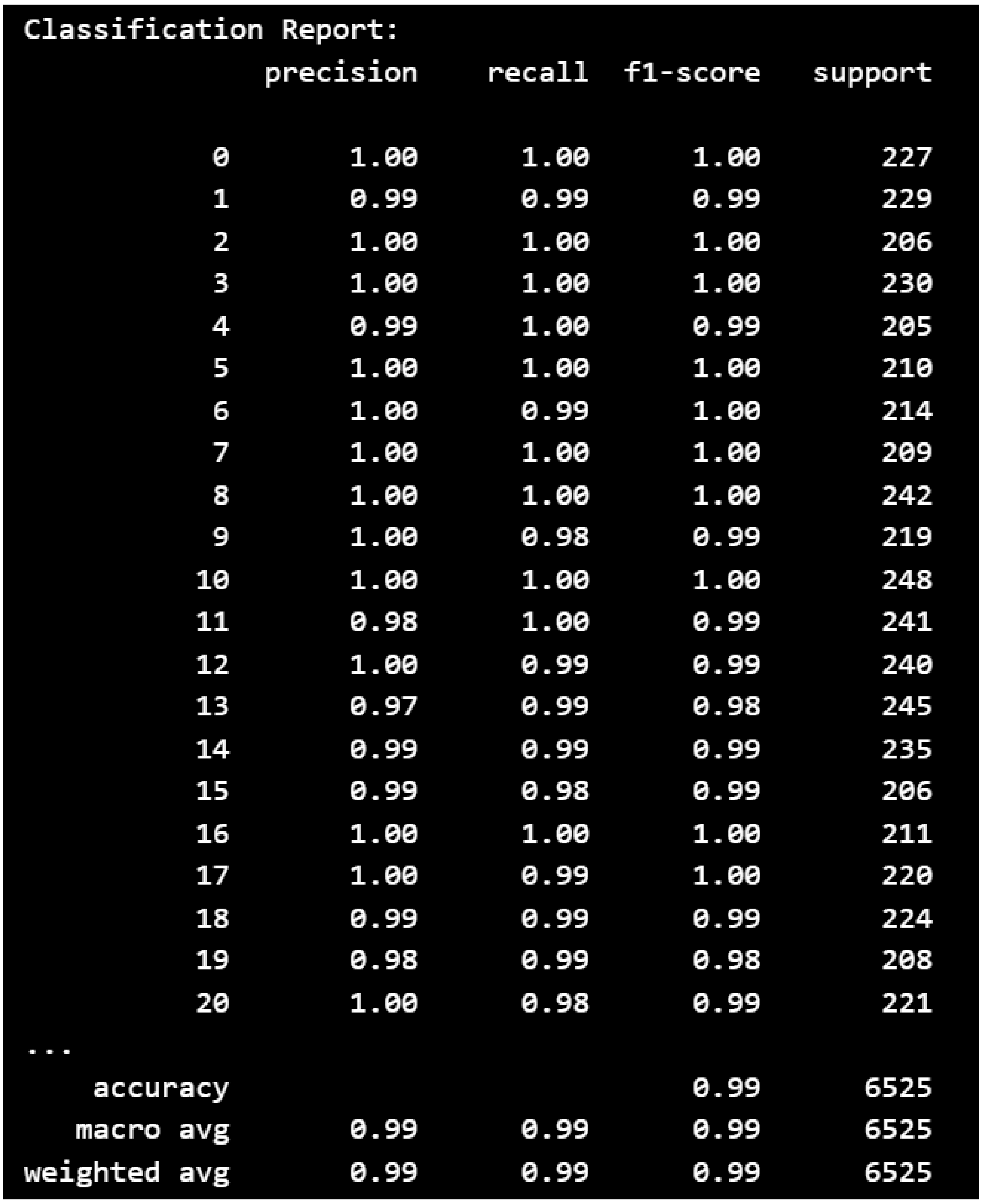


**Figure 5.3:** Loss vs epoch

## 5.6 MobileNet- Classification Report

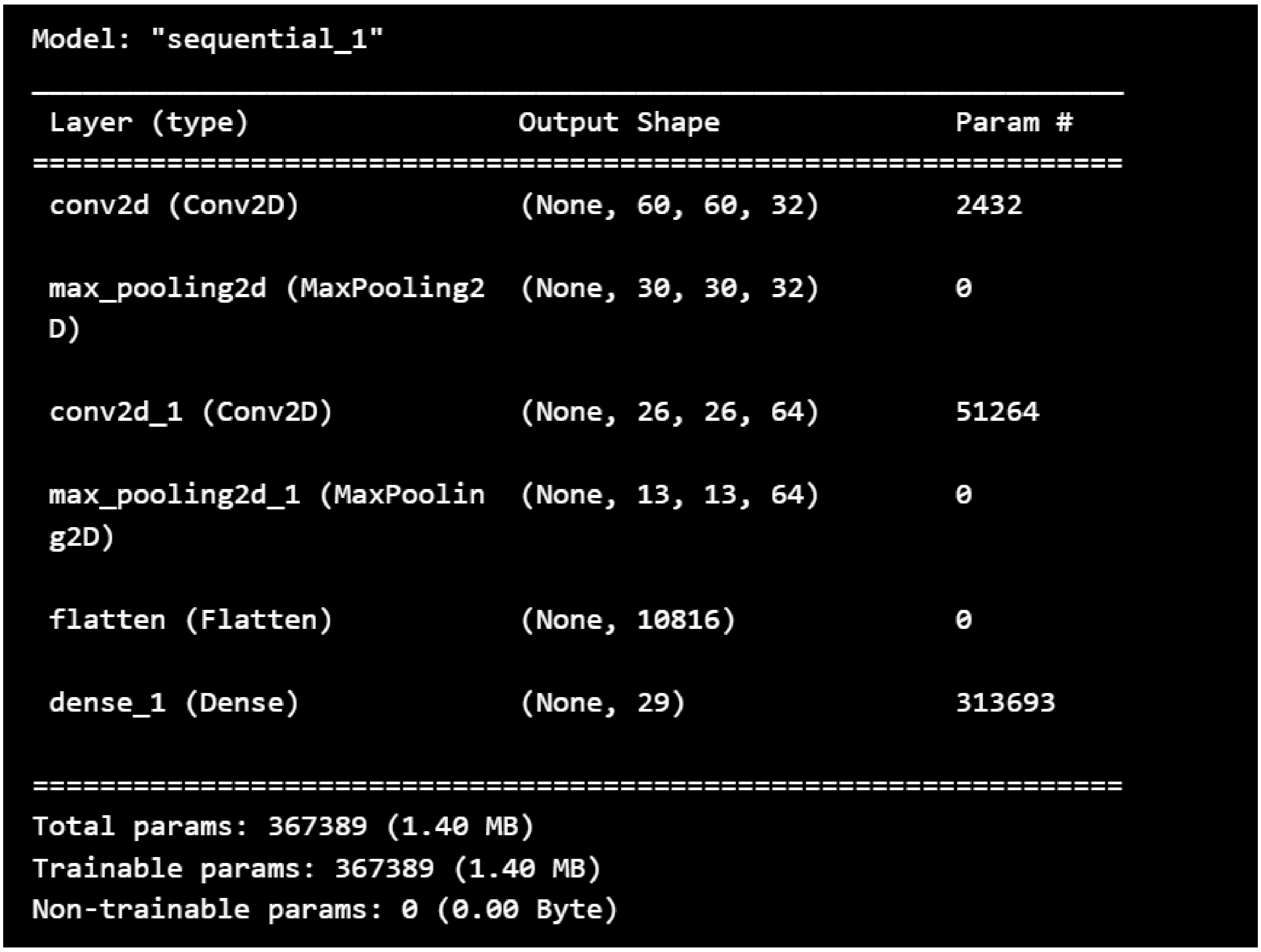


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       227
           1       0.99      0.99      0.99       229
           2       1.00      1.00      1.00       206
           3       1.00      1.00      1.00       230
           4       0.99      1.00      0.99       205
           5       1.00      1.00      1.00       210
           6       1.00      0.99      1.00       214
           7       1.00      1.00      1.00       209
           8       1.00      1.00      1.00       242
           9       1.00      0.98      0.99       219
          10       1.00      1.00      1.00       248
          11       0.98      1.00      0.99       241
          12       1.00      0.99      0.99       240
          13       0.97      0.99      0.98       245
          14       0.99      0.99      0.99       235
          15       0.99      0.98      0.99       206
          16       1.00      1.00      1.00       211
          17       1.00      0.99      1.00       220
          18       0.99      0.99      0.99       224
          19       0.98      0.99      0.98       208
          20       1.00      0.98      0.99       221
...
    accuracy                           0.99      6525
   macro avg       0.99      0.99      0.99      6525
weighted avg       0.99      0.99      0.99      6525
```

**Figure 5.4:** Classification Report

## 5.7 Student Model- Architecture



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 60, 60, 32)        2432

 max_pooling2d (MaxPooling2  (None, 30, 30, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 26, 26, 64)        51264

 max_pooling2d_1 (MaxPoolin  (None, 13, 13, 64)        0
 g2D)

 flatten (Flatten)           (None, 10816)             0

 dense_1 (Dense)             (None, 29)                313693

=================================================================
Total params: 367389 (1.40 MB)
Trainable params: 367389 (1.40 MB)
Non-trainable params: 0 (0.00 Byte)
```

**Figure 5.5:** Student Model- Architecture

## 5.8 Student Model- Training and validation Accuracy

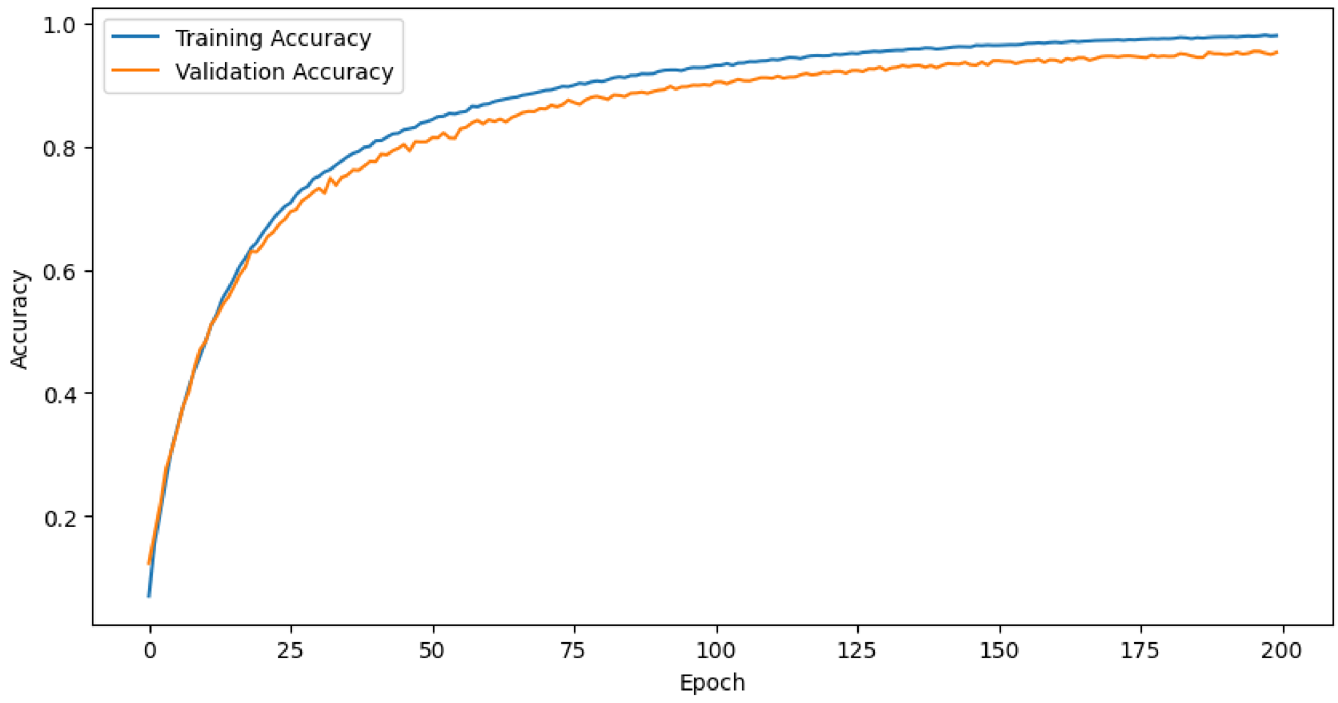
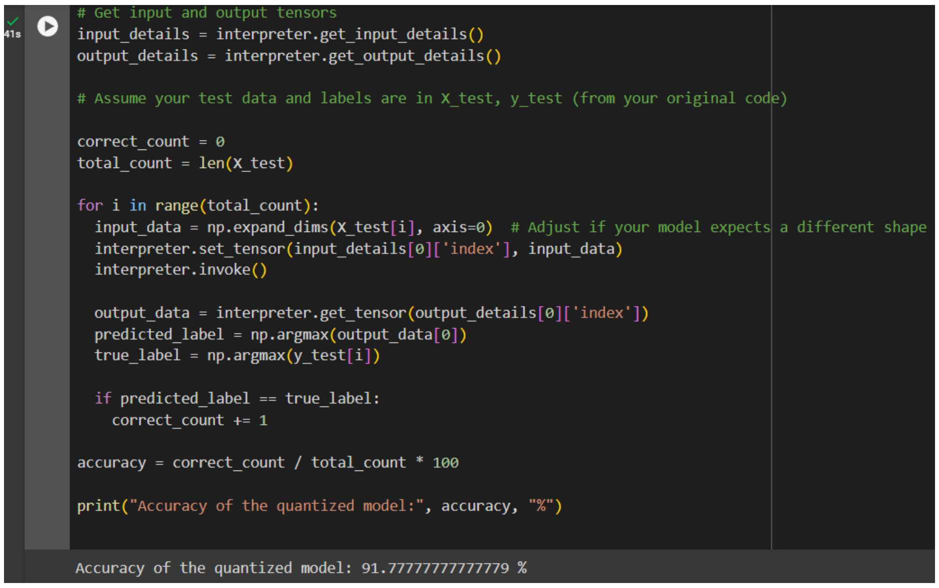


**Figure 5.6:** Student Model- Training and validation Accuracy

## 5.9 Quantized tflite mode-Accuracy



```
# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Assume your test data and labels are in X_test, y_test (from your original code)

correct_count = 0
total_count = len(X_test)

for i in range(total_count):
  input_data = np.expand_dims(X_test[i], axis=0)  # Adjust if your model expects a different shape
  interpreter.set_tensor(input_details[0]['index'], input_data)
  interpreter.invoke()

  output_data = interpreter.get_tensor(output_details[0]['index'])
  predicted_label = np.argmax(output_data[0])
  true_label = np.argmax(y_test[i])

  if predicted_label == true_label:
    correct_count += 1

accuracy = correct_count / total_count * 100

print("Accuracy of the quantized model:", accuracy, "%")
```

```
Accuracy of the quantized model: 91.77777777777779 %
```

**Figure 5.7:** Quantized tflite model-Accuracy

## 5.10 Hardware setup

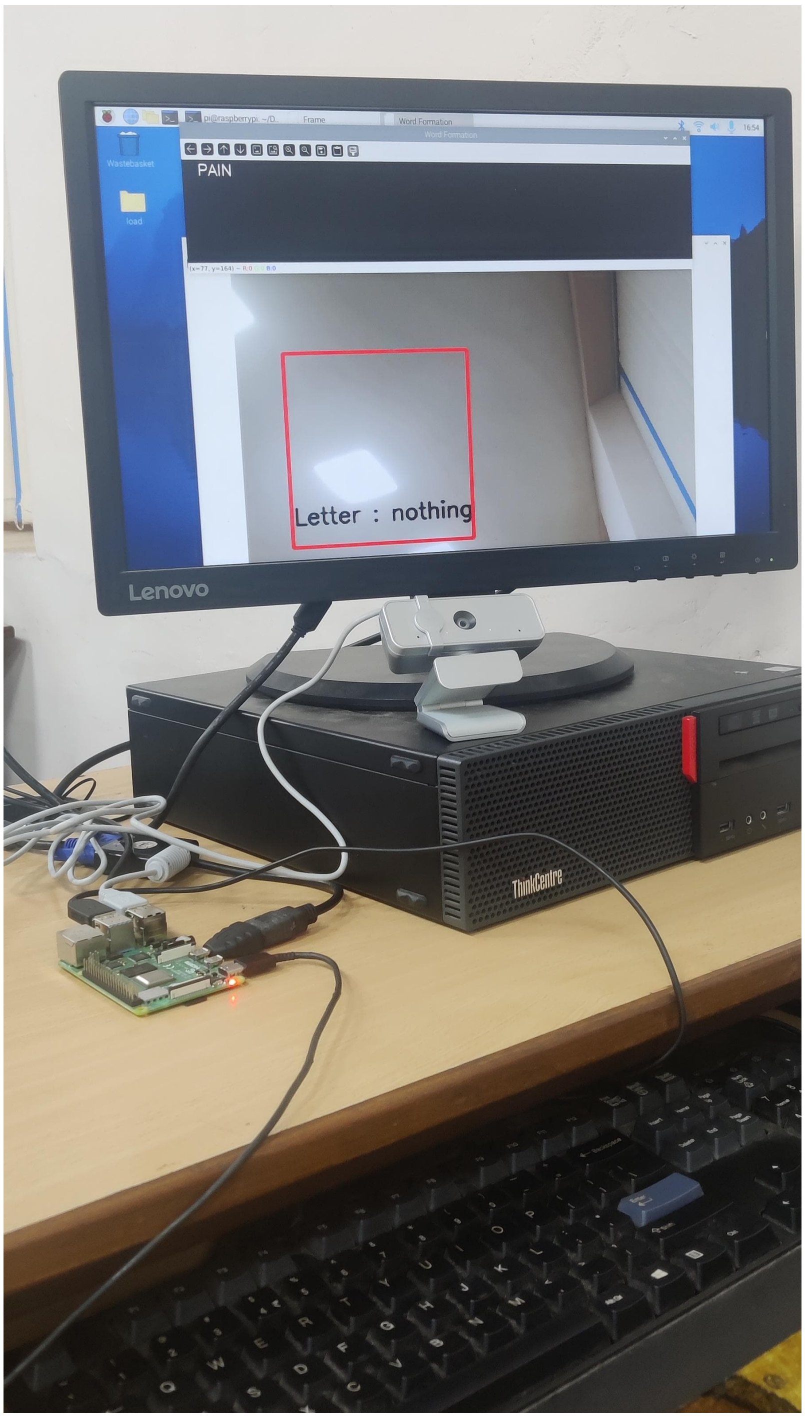


**Figure 5.8:** Hardware setup

## 5.11 REALTIME RESULT

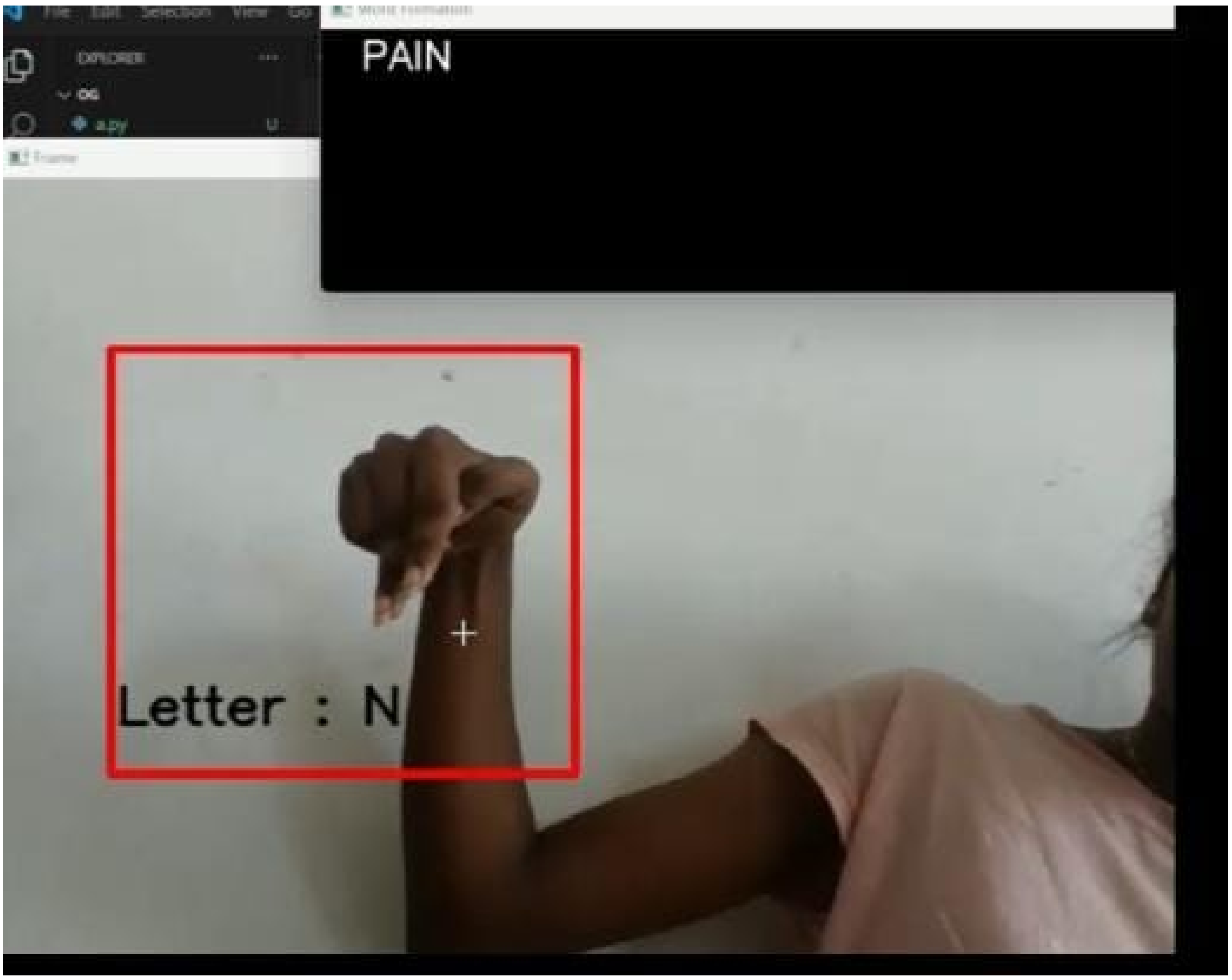


**Figure 5.9:** Realtime Result

# Chapter 6

# Conclusion and Future Scope

In conclusion, our project focused on developing a TinyML-based Sign Language Recognition System tailored for medical consultations, aimed at facilitating communication between healthcare professionals and patients with hearing impairments. Initially, the challenge of deploying the system onto edge devices was hindered by the large model size of 37.47MB. However, through meticulous optimization efforts, we successfully reduced the size of our MobileNet model. Leveraging techniques such as knowledge distillation, we achieved a model size of 12.60MB, further optimized to 3.25MB through quantization and TensorFlow Lite conversion(refer to Table 5.2). The Raspberry Pi 4B was selected as the designated edge device for deployment, given its compatibility with project requirements.

Looking ahead, future work will focus on enhancing the system's capabilities. One avenue involves incorporating efficient doctor-to-patient communication features, enabling two-way interaction and seamless exchange of information. Additionally, exploring the integration of audio channels alongside sign language could further enhance accessibility and inclusivity. Community engagement initiatives will also be vital, fostering collaboration and feedback from stakeholders to ensure the system meets diverse user needs. By pursuing these avenues, we aim to advance the field of assistive technologies for healthcare, promoting improved communication and accessibility for individuals with hearing impairments in medical settings.

# References

[1] Yuming Chen, Yanyin Yao, Shi Hu, Wuyang Qin, Xue Mei, & Qihang Zhang. Sign language recognition and translation method based on VTN.

[2] Vijay Janapa Reddy, Boris Murmann, Mohd. Shafique, & Theocharis Theocharides. TinyML: Current progress, research challenges, and future roadmap.

[3] Ritika Basavaraj Hiremath, Vanshika Sai Ramadurgam, Deepak Kumar Shaw, Santhosh Kumar B, & Rachna P. Survey on implementation of TinyML for real-time sign language recognition using smart gloves.

[4] R. Naveen Karthik, A. Kaliselvi, K. Anitha, & K. C. Varsheni. Gesture-based sign language recognition system.

[5] Srujal Reddy Gundedi, Vishal Satpute, Venaktesh Kandukuri, & Vipin Kamble. Deaf and mute sign language translator on static alphabets gestures.

[6] Abhijit Das, Anirbit Sengupta, & Md. Tausif Mallick. A cost-effective design and implementation of Arduino-based sign language interpreter.

[7] Abu-Jamie Tanseem N, & Samy S Abu-Naser Prof. Dr. Classification of sign language using MobileNet-deep learning.

[8] Hongliang Fan, Kun Xia, Weiwei Lu, & Qiang Zhao. A sign language recognition system applied to deaf-mute medical consultation.

[9] Hongliang Fan, Kun Xia, Weiwei Lu, & Qiang Zhao. A sign language recognition system applied to deaf-mute medical consultation.