

**REAL TIME HAND POSTURE RECOGNITION USING STM32
AI MODEL ZOO AND TIME-OF-FLIGHT SENSOR**

MUHAMMAD AMMAR BIN PAUZAN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**REAL TIME HAND POSTURE RECOGNITION USING
STM32 AI MODEL ZOO AND TIME-OF-FLIGHT SENSOR**

MUHAMMAD AMMAR BIN PAUZAN

**This report is submitted in partial fulfilment of the requirements
for the degree of Bachelor of Electronic Engineering with Honours**

**Faculty of Electronics and Computer Technology and
Engineering
Universiti Teknikal Malaysia Melaka**

FINAL YEAR

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Real Time Hand Posture Recognition Using Stm32
Ai Model Zoo And Time-Of-Flight Sensor
Sesi Pengajian : 2024/2025

Saya MUHAMMAD AMMAR BIN PAUZAN mengaku membenarkan laporan
Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat
kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan
pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐

SULIT*

(Mengandungi maklumat yang berdarjah
keselamatan atau kepentingan Malaysia
seperti yang termaktub di dalam AKTA
RAHSIA RASMI 1972)

☐

TERHAD*

(Mengandungi maklumat terhad yang
telah ditentukan oleh organisasi/badan di
mana penyelidikan dijalankan.

☐

TIDAK TERHAD

Disahkan oleh:

(TANDATANGAN PENULIS)

(COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: No2 Jln Seri
Saujana 3, Desa
Seri Saujana,
Sungai Merab,
43000, Kajang,
Selangor

Tarikh : 20/1/2025

Tarikh : 20/1/2025

DECLARATION

I declare that this report entitled “Real Time Hand Posture Recognition using STM32 AI Model Zoo and Time-Of-Flight sensor” is the result of my own work except for quotes as cited in the references.

Signature :

Author : MUHAMMAD AMMAR BIN PAUZAN

Date :24/6/2025.....

APPROVAL

I hereby declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours

Signature :

Supervisor Name : Ts. Dr. Sani Irwan Bin Md Salim

Date :

DEDICATION

I dedicate this thesis to my beloved parent, Pauzan Bin Ahmar, whose unwavering support and encouragement have been the foundation that allowed me to complete this project. I also extend my heartfelt gratitude to my supervisor, Ts. Dr. Sani Irwan Bin Md Salim, whose invaluable guidance, support, and mentorship have played a pivotal role in the successful completion of my project.

ABSTRACT

This project suggests a real-time hand posture recognition system with the use of a custom-built Convolutional Neural Network (CNN) architecture, STM32F401RE microcontroller, and VL53L8A1 Time-of-Flight (ToF) sensor to correctly recognize and classify hand gestures. The CNN model built specifically for this scenario and trained as such achieved an impressive validation accuracy rate of 99.48%, boasting an outstanding generalization and dependability under real-time scenarios. Firstly, the STM32 AI Model Zoo was used to implement pre-optimized models but instead went with a custom model implemented using STM32Cube.AI for better performance and flexibility. The resulting model was successfully embedded in the STM32F401RE, enabling efficient on-device inference with less latency and without an external computer. This compact and power-saving solution is highly suitable for human-computer interaction, robotics, and virtual reality applications where gesture recognition should be accurate and rapid.

ABSTRAK

Projek ini mencadangkan sistem pengecaman postur tangan masa nyata dengan penggunaan seni bina Convolutional Neural Network (CNN) tersuai, mikropengawal STM32F401RE dan VL53L8A1 Time-of-Flight (ToF) untuk mengecam dan mengelaskan gerak isyarat tangan dengan betul. Model CNN yang dibina khusus untuk senario ini dan dilatih sedemikian mencapai kadar ketepatan pengesanan yang mengagumkan sebanyak 99.48%, yang mempunyai generalisasi dan kebolehpercayaan yang luar biasa di bawah senario masa nyata. Pertama, Zoo Model AI STM32 digunakan untuk melaksanakan model yang dioptimumkan tetapi sebaliknya menggunakan model tersuai yang dilaksanakan menggunakan STM32Cube.AI untuk prestasi dan fleksibiliti yang lebih baik. Model yang terhasil telah berjaya ditanamkan dalam STM32F401RE, membolehkan inferens pada peranti yang cekap dengan kurang kependaman dan tanpa komputer luaran. Penyelesaian padat dan penjimatan kuasa ini sangat sesuai untuk interaksi manusia-komputer, robotik dan aplikasi realiti maya di mana pengecaman gerak isyarat harus tepat dan pantas.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Ts. Dr. Sani Irwan Bin Md Salim, for his invaluable guidance, support, and encouragement throughout the completion of this project, Real-Time Hand Posture Recognition by Using STM32 AI Model Zoo and Time-of-Flight Sensor. His expertise and constructive feedback were instrumental in overcoming challenges and ensuring the success of this work. I am also immensely thankful to my parents, especially Pauzan Bin Ahmar, for their unwavering love, patience, and encouragement. They have been my greatest inspiration throughout my academic life. I would also like to thank the Faculty of Electronic Engineering and Computer Engineering and Universiti Teknikal Malaysia Melaka (UTeM) for providing the necessary resources, facilities, and a conducive environment for research. Particular thanks to the faculty members and staff who directly or indirectly assisted me in this project. In addition, I am grateful to my peers and colleagues who have provided constructive feedback, shared information, and offered moral support during this project. Their companionship has rendered this journey fruitful as well as enjoyable. Last but not least, I would like to thank any external contributors, including the developers of STM32 AI Model Zoo and the makers of Time-of-Flight sensors, whose

technology formed the cornerstone of my project. Their innovations have facilitated the completion of this work. This project has been a challenging but rewarding experience, and I am truly proud to have reached this point with the assistance of everyone mentioned above. Thank you.

TABLE OF CONTENTS

| | |
|--|------------|
| Declaration | |
| Approval | |
| Dedication | |
| Abstract | i |
| Abstrak | ii |
| Acknowledgements | iii |
| Table of Contents | v |
| List of Figures | ix |
| List of Tables | xi |
| List of Symbols and Abbreviations | xii |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Project Overview | 1 |
| 1.2 Problem statement | 2 |
| 1.3 Objective | 3 |
| 1.4 Scope | 3 |
| 1.5 Thesis layout | 5 |

| | |
|--|-----------|
| CHAPTER 2 BACKGROUND STUDY | 6 |
| 2.1 Hand Posture Recognition | 6 |
| 2.1.1 Healthcare Applications | 7 |
| 2.1.2 Robotics Applications | 7 |
| 2.1.3 Virtual and Augmented Reality (VR/AR) Applications | 8 |
| 2.2 Algorithms for Hand Posture Recognition | 9 |
| 2.2.1 2D Convolutional Neural Networks (2D CNNs) | 9 |
| 2.2.2 Deep Neural Networks (DNNs) | 10 |
| 2.2.3 Support Vector Machines (SVMs) | 11 |
| 2.3 Platforms for Hand Posture Recognition | 11 |
| 2.3.1 Time-of-Flight (ToF) Sensors | 12 |
| 2.3.2 Electromyography (EMG) Sensors | 13 |
| 2.3.3 RGB-D Cameras | 14 |
| 2.4 Microcontrollers and Algorithms with Sensor | 14 |
| 2.5 Literature Review | 16 |
| CHAPTER 3 METHODOLOGY | 21 |
| 3.1 Introduction | 21 |
| 3.2 Project workflow | 21 |
| 3.3 Dataset | 28 |
| 3.3.1 Dataset Collection using GestureEVK | 28 |

| | | |
|---------|---|----|
| 3.3.2 | Dataset Preprocessing | 32 |
| 3.3.2.1 | Reshaping | 32 |
| 3.3.2.2 | Normalization | 32 |
| 3.3.2.3 | Augmentation | 32 |
| 3.3.2.4 | Splitting The processed dataset was divided into: | 33 |
| 3.3.3 | Dataset Summary | 33 |
| 3.4 | Developer Cloud Process Simulation | 34 |
| 3.4.1 | Model Selection | 35 |
| 3.4.2 | Select Platform (Board) | 36 |
| 3.4.3 | Quantize the Model | 36 |
| 3.4.4 | Optimize the Model | 36 |
| 3.4.5 | Analyze the Results | 37 |
| 3.4.6 | Benchmark with STM32F401RE | 37 |
| 3.4.7 | Generate the Project | 38 |
| 3.5 | Deployment | 39 |
| 3.6 | Hardware | 41 |
| 3.6.1 | STM32F401RE Microcontroller | 42 |
| 3.6.2 | Time of Flight (ToF) Sensor | 43 |
| 3.6.3 | Integration of Hardware | 44 |
| 3.7 | Software | 45 |

| | | |
|--|---|-----------|
| 3.7.1 | Python-Based Model Development | 46 |
| 3.7.2 | Deployment of Model using STM32Cube.AI | 47 |
| 3.8 | CNN Model Architecture | 48 |
| 3.9 | Experimental Setup | 50 |
| CHAPTER 4 RESULTS AND DISCUSSION | | 53 |
| 4.1 | Introduction | 53 |
| 4.2 | Training and Validation | 54 |
| 4.3 | Confusion Matrix Evaluation | 56 |
| 4.4 | Real-Time Gesture Prediction Testing | 58 |
| 4.5 | Comparative Evaluation with STM32 AI ModelZoo | 65 |
| 4.6 | Summary of Findings | 67 |
| CHAPTER 5 CONCLUSION AND FUTURE WORKS | | 69 |
| 5.1 | Conclusion | 69 |
| 5.2 | Future Work | 71 |
| REFERENCES | | 72 |
| LIST OF PUBLICATIONS AND PAPERS PRESENTED | | 76 |

LIST OF FIGURES

| | |
|--|-------------------------------------|
| Figure 3.1: Methodology Flow Chart | 22 |
| Figure 3.2: Like gesture | Error! Bookmark not defined. |
| Figure 3.3: Dislike gesture | Error! Bookmark not defined. |
| Figure 3.4 : FlatHand gesture | Error! Bookmark not defined. |
| Figure 3.5: Fist gesture | Error! Bookmark not defined. |
| Figure 3.6 Love gesture | Error! Bookmark not defined. |
| Figure 3.7 : BreakTime gesture | Error! Bookmark not defined. |
| Figure 3.8: CrossHands gesture | Error! Bookmark not defined. |
| Figure 3.9: Developer Cloud Process Flow Chart | 34 |
| Figure 3.10: STM32 Microcontroller | 42 |
| Figure 3.11: Time of Flight sensor | 43 |
| Figure 3.12: Time of Flight sensor works | 44 |
| Figure 3.13: Experimental Setup of Project | 51 |
| Figure 4.1: Accuracy and Loss Graph | 55 |
| Figure 4.2: Confusion matrix evaluation | 57 |
| Figure 4.3 : Terminal Output for FlatHand detection | 59 |
| Figure 4.4: Terminal Output for FlatHand detection | Error! Bookmark not defined. |
| Figure 4.5: Terminal Output for Fist gesture detection | Error! Bookmark not defined. |

Figure 4.6 : Terminal Output for CrossHands gesture detection**Error! Bookmark not defined.**

Figure 4.7: Terminal Output for Love gesture detection**Error! Bookmark not defined.**

Figure 4.8: Terminal Output for Dislike gesture detection**Error! Bookmark not defined.**

Figure 4.9: Terminal Output for Like gesture detection**Error! Bookmark not defined.**

Figure 4.10: Terminal Output for BreakTime gesture detection**Error! Bookmark not defined.**

Figure 4.11 Terminal output for training model in Visual Code 65

LIST OF TABLES

| | |
|--|----|
| Table 2.1: Literature review Table..... | 17 |
| Table 3.1 Class Definition Hand Posture..... | 29 |
| Table 3.2 Layer Type..... | 48 |
| Table 4.1 Testing All Classes and output | 59 |
| Table 4.2 Confidence Score for each gesture detection..... | 64 |
| Table 4.3 Inference time and resource usage comparison between STM32 Ai ModelZoo pretrained model(top) and custome developed model (bottom) | 67 |

LIST OF SYMBOLS AND ABBREVIATIONS

| | | |
|-----|---|------------------------------|
| ToF | : | Time Of Flight |
| AI | : | Artificial intelligence |
| CNN | : | Convolutional Neural Network |

CHAPTER 1

INTRODUCTION

1.1 Project Overview

This project focuses on developing a real-time hand posture recognition system suitable for smart home applications, specifically designed to assist elderly and bedridden individuals in controlling basic devices seamlessly. By enabling gesture-based interactions, the system aims to improve accessibility and ease of use in daily activities, offering a frictionless interface for smart home environments. Traditional hand posture recognition systems tend to be vision-based or image-based, which requires extremely high memory and is unsuitable for embedded systems. They can also have problems of excessive file size or memory footprint and reduced accuracy in low light conditions, making them less practical for resource-limited environments.

To circumvent these constraints, this project applies Time-of-Flight (ToF) sensors and STM32 microcontrollers using lightweight AI models from the STM32 AI Model Zoo. Time-of-Flight (ToF) sensors provide precise depth information, delivering high-quality gesture recognition even in challenging lighting environments, with minimal memory consumption. The embedded AI-optimized STM32 platform enables real-time processing on a low-resource system, thus making the solution cost-effective and scalable. Through the combination of embedded and AI technology, this project aims to create a robust and efficient gesture recognition system, offering a low memory yet accurate alternative to traditional vision-based systems for implementation in smart homes.

1.2 Problem statement

Vision-based recognition systems, including those for hand pose estimation, often face challenges in poor lighting conditions, requiring substantial computational resources for feature extraction and lighting normalization, as highlighted by Chen et al. and Hundhausen et al. [1] [2]. This limitation is further compounded by the need for large memory footprint, often necessitating the use of PCs or GPUs to handle even simple hand posture recognition tasks [2]. Additionally, these systems exhibit low performance and accuracy, particularly in environments with poor lighting conditions [3].

In this regard, a low-cost, effective, and real-time hand position recognition system that can function well on embedded devices with limited resources is required. Because Time of Flight (ToF) sensors can precisely record depth information while reducing computing overhead, offering a possible substitute for traditional image-

based systems [4][5]. However, using ToF sensor data in a real-time application necessitates AI models and efficient algorithms that can work within the limitations of microcontroller systems like STM32, which have little memory and processing capacity [6]. Although pre-trained models are available in AI Model Zoo, hardware limits and the requirement for high identification accuracy in a variety of situations make it difficult to install these models onto embedded platforms [7][8]. To provide natural, real-time interactions across a range of applications, it is imperative to design a hand position recognition system utilizing an effective AI model that can be smoothly linked with ToF sensors on STM32 microcontrollers [9].

1.3 Objective

The project's main goal is to develop a real-time hand posture recognition system using STM32 AI Model Zoo and Time of Flight (ToF) sensor. The objective of the project is as stated below:

- i. To develop a real-time hand posture recognition system using STM32 AI Model Zoo and Time of Flight (ToF) sensor
- ii. To optimize hand posture models for real-time performance on an STM32 microcontroller

1.4 Scope

The integration of hardware and software is being done under the development and design process of this project. A suitable STM32 microcontroller will be selected and connected with Time-of-Flight (ToF) sensors, which will be used for the collection of accurate depth information for hand gesture recognition. On the software side, tools like STM32Cube.AI will be used for the deployment of AI models on the embedded system. These models can be adapted from existing pre-trained networks or trained

anew to fit the specific hand gestures required. Prioritization will be given to incorporating efficient algorithms for real-time data processing to ensure responsive and seamless system performance under the resource limitations of the STM32 platform.

To support this, a number of hand gestures will be learned as depth data with the help of ToF sensors. This data can comprise data from the STM32 AI Model Zoo or can be augmented by custom collection. The data will be subjected to various preprocessing stages, such as cleaning, normalization, and feature extraction, to make it ready for training. Based on the complexity of gestures, Convolutional Neural Networks (CNNs) will be fine-tuned or trained end-to-end, with the goal of achieving a balance of classification accuracy along with memory efficiency to accommodate embedded deployment.

Once the model is integrated, end-to-end testing will be conducted to validate the performance of the system in varied real-world situations, like varying light and variation in hands. While the system is being tested, factors like memory constraints and processing power will be addressed with optimization techniques like pruning and quantization. Through iterative testing and tuning, the system will be optimized to execute in real time and with high accuracy and reliability on the STM32 platform.

1.5 Thesis layout

The study is divided between main chapters of Introduction, Literature Review, Methodology, Result and Discussion and Conclusion.

Chapter 1 is Introduction that will clarify the project overview, problem statement, objectives, scope of work and summary of the thesis.

Chapter 2 is Background Study that will explain the previous research on microcontroller system, hand posture recognition method and hand posture recognition in microcontroller with the basic information regarding the aspect of different method and techniques. This also includes the method of system design the way to use the Time-of-Flight (ToF)sensor to recognize hand posture.

Chapter 3 is Methodology that will clarify the procedures of every project involved within the thesis and completion of the project.

Chapter 4 is the Results and Discussion, where the findings of the project are presented and analyzed. This includes the performance of the trained AI model, validation accuracy, confusion matrix results, system latency, and the effectiveness of real-time gesture recognition on the STM32 microcontroller. The results are compared against benchmarks and discussed in terms of how well they meet the project objectives.

Chapter 5 is the Conclusion and Future Work, which summarizes the overall achievements of the project, reflects on how the objectives were met, and highlights the system's potential applications. This chapter also discusses the limitations encountered and suggests possible improvements and future work, such as expanding gesture classes or optimizing the system for more advanced embedded platforms.

CHAPTER 2

BACKGROUND STUDY

2.1 Hand Posture Recognition

Hand posture recognition is a groundbreaking technology that brings natural, gesture-based communication between humans and machines a step closer, playing a vital role in human-computer interaction. Recent advances in sensor technology and machine learning methodologies have given rise to two main approaches in hand gesture detection: image/video-based systems and Time-of-Flight (ToF) sensors.

Image/video-based systems use RGB cameras or RGB-D cameras for acquiring visual information and recognizing gestures using sophisticated algorithms. However,

such systems are very dependent on favourable environmental conditions for example, adequate lighting and also tend to be computationally expensive because of the high-resolution information involved. On the other hand, ToF sensors capture precise depth information by measuring the time light takes to reflect back and are therefore robust in low-light or high-glare environments [10]. Both techniques have applications in health care, robotics, virtual/augmented reality, but their appropriateness hinges on the particular task demands and limitations. In recent years there have been advancements in machine learning algorithms, sensor technology, and embedded microcontrollers that have brought systems describing real-time performance in resource-constrained environments closer to realization. Therefore, there are the applications of hand posture recognition:

2.1.1 Healthcare Applications

Hand posture recognition systems have made significant contributions to healthcare, particularly in rehabilitation and assistive technologies. These systems enhance physiotherapy monitoring by enabling precise motion tracking and gesture recognition. Mohamed et al. [14] integrated CNNs with ToF sensors to develop a physiotherapy monitoring system, achieving real-time performance for improved rehabilitation outcomes. Krishna et al. [11] proposed a TinyML accelerator capable of decoding hand kinematics in brain-computer interfaces, optimizing performance for wearable biomedical devices.

2.1.2 Robotics Applications

In robotics, hand posture recognition facilitates seamless human-robot interaction (HRI) by enabling intuitive gesture-based controls. Bartoli et al. [10] have demonstrated the use of VL53L8 ToF sensors with STM32 microcontrollers to

achieve static gesture recognition with 92% accuracy and low latency. Such innovations enhance collaborative robotic systems with the provision of precise control and interaction in industrial as well as assistive settings.

2.1.3 Virtual and Augmented Reality (VR/AR) Applications

In VR/AR, hand posture recognition plays a vital role in creating immersive user experiences via gesture navigation and interaction with virtual environments. Wang et al. [13] combined ToF sensors and CNN architectures to encourage robust hand posture recognition that is robust against varying lighting and occlusion. The technology can be applied to games, virtual training, and interactive simulations.

2.2 Algorithms for Hand Posture Recognition

The used algorithms in hand posture recognition systems are vital to their efficiency. They enable the extraction of meaningful features from input data, classification of gestures, and efficiency in real-time processing in systems. The two most remarkable algorithms in this area are 2D CNNs, DNNs, and SVMs. All these approaches have their strengths, weaknesses, and areas of application.

2.2.1 2D Convolutional Neural Networks (2D CNNs)

2D CNNs are especially good for hand posture recognition, particularly in cases involving processing of 2D image data. Unlike regular CNNs that are designed to process higher-dimensional data sets, 2D CNNs are optimized to derive features such as edges and contours from 2D images. This renders them ideally suited for static gesture recognition, where computational simplicity and efficiency are of paramount importance. Bartoli et al. [10] demonstrated the efficiency of 2D CNNs on limited systems by combining them with VL53L8 ToF sensors to achieve static hand posture classification. Their system's accuracy was 92% with minimal latency and was appropriate for real-time applications as well as embedded systems. The compactness of the 2D CNN structure made it optimal to operate with depth images without overburdening the system's computing resources. It has been stated by Patel et al. [17] that a comparative study was conducted, which highlighted the advantages of 2D CNNs over traditional CNNs in low computational overhead scenarios. Optimizing

network structure for static gestures, they achieved effective feature extraction and 2D CNN suitability to implement in robotics and assistive technology.

2.2.2 Deep Neural Networks (DNNs)

Deep Neural Networks (DNNs) perform well on dynamic hand gesture recognition by taking advantage of their layered structure to capture intricate relations in high-dimensional data. DNNs are capable of detecting temporal and spatial patterns and thus are apt for those tasks that involve sequential data, like motion-based gestures. Ashraf et al. [19] employed a DNN-based system for hand gesture classification from surface electromyography (sEMG) signals. By utilizing a large and varied dataset to train a multi-layer DNN, the system was able to achieve over 90% accuracy in identifying dynamic gestures, demonstrating the robustness and viability for application in biomedical areas. The approach is of particular value in prosthetic rehabilitation and control, where gesture recognition needs to be accurate. Zhang et al. [18] further applied the use of DNNs to hand posture recognition from RGB-D data. Their approach synthesized spatial and temporal attributes from the RGB and depth modalities, allowing the DNN to learn to accommodate variations in hand size, shape, and lighting. This ability to accommodate this kind of variation is characteristic of DNNs' ability to handle challenges in real-world environments, including occlusion and environmental change.

2.2.3 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are also computationally lighter than deep models, making them adequate for low-end devices and systems that focus on static hand gesture recognition. SVMs operate by finding an optimal hyperplane to separate data into classes, making good performance feasible with small-sized datasets. Aggarwal et al. [15] demonstrated the effectiveness of SVMs in gesture recognition systems with accelerometers. They employed accelerator data to extract features such as frequency and amplitude of the signal and used SVMs to label gestures into predefined classes. The approach was 85% accurate with a very small computational burden, rendering it suitable for cheap wearable devices. Zholshiyeva et al. [20] extended the application of SVMs by pairing them with MediaPipe in real-time sign language recognition of Kazakh. MediaPipe was utilized for pre-processing, i.e., hand tracking and palm detection, while SVMs classified the extracted features into specific gestures. The system was correct to 88%, indicating SVMs' ease in integration with pre-processing pipelines to have real-time lightweight and efficient gesture recognition.

2.3 Platforms for Hand Posture Recognition

Platform selection is a critical component in the development of hand posture recognition systems, as it directly influences accuracy, reliability, and usability. It acts as the basic tool in the acquisition of data, gathering indispensable input that forms the basis of processing and interpretations of gestures in general. Based on different contextual applications, several platforms perform at high levels utilizing their

respective capability to cater to the requirements of many applications in healthcare, robotics, virtual/augmented reality, and many more.

Among the commonly used platforms, RGB-D cameras combine color and depth sensing to provide full spatial information and are thus best suited for applications requiring accurate 3D movement analysis. On the other hand, ToF sensors have better accuracy in depth measurements and are more robust in low-light conditions, hence very effective in real-time gesture recognition and VR applications. Concurrently, Electromyography (EMG) sensors offer a distinctive approach, recording muscular activity to enable gesture recognition in situations of visual occlusion or low visibility, mainly in the field of wearable and assistive technologies.

Each of the platforms discussed here has a unique role in the area of hand posture recognition and has certain strengths that make them more suitable for certain application domains. We will discuss in the following sections the operational features and advantages of each platform and how they can contribute to developing more functional gesture recognition systems.

2.3.1 Time-of-Flight (ToF) Sensors

Time-of-Flight (ToF) sensors work by measuring the time taken for reflected light that has hit an object to travel back and therefore can create highly accurate depth maps. Their second most valuable asset is their capability to work in poor lighting

conditions, which makes them ideal for use indoors. For example, Bartoli et al. [10] used VL53L8 ToF sensors and a 2D convolutional neural network (CNN) and achieved 92% accuracy in recognizing static hand gestures. Due to their precise depth measurement and low response time, ToF sensors are well suited for real-time applications like robotics and assistive technology. Similarly, Wang et al. [13] employed ToF sensors for gesture identification in virtual reality environments by leveraging the sensitivity for capturing minimal depth details. Such accuracy ensures accurate hand movement detection even in harsh illumination conditions and hence makes ToF sensors a choice of preference in high-performance gesture identification.

2.3.2 Electromyography (EMG) Sensors

Electromyography (EMG) sensors represent an alternate paradigm through capturing the electrical signals emanating from the movements of muscles. Since they are vision-independent, EMG sensors are immune to lighting conditions as well as occlusions and are of special value in prosthetics and wearable devices. Krishna et al. [11] demonstrated that hand gestures can be accurately decoded by EMG sensors in biomedical wearables. Reading muscle signals as movements, EMG sensors allow individuals to control external devices naturally. Ashraf et al. [19] combined EMG signals with deep neural networks (DNNs) for hand gesture classification with very good accuracy. This is a sign of the effectiveness of EMG sensors for real-time gesture recognition in applications such as physical rehabilitation and assistive technology. EMG sensors can offer a highly sensitive and accurate way of controlling gestures even when cameras are not able to, using machine learning algorithms.

2.3.3 RGB-D Cameras

RGB-D cameras, which capture color (RGB) and depth, are commonly used for dynamic gesture recognition. They perform particularly well at tracking 3D motion, which comes in handy when spatial sensitivity is required in an application. Depth data provides extra capability to the system's recognition capability of how far away an object—or the hand—is from the camera, making overall recognition accuracy improved. Elboushaki et al. [12] used RGB-D cameras in their MultiD-CNN and achieved 96.1% recognition accuracy of complex gestures. The combination of color and depth makes RGB-D cameras a stable solution for tasks like sign language detection or interaction with virtual environments. Since these cameras offer an absolute representation of the hand's motion and position, they offer constant accuracy irrespective of the user, lighting, or background variation.

2.4 Microcontrollers and Algorithms with Sensor

Coupled with sensors like ToF, EMG, and RGB-D cameras, microcontrollers take the form of intelligence for low-power, compact hand posture recognition systems. Microcontrollers process sensor data and run machine learning models—like 2D CNNs, DNNs, and Support Vector Machines (SVMs)—to recognize gestures in real-time.

For instance, Bartoli et al. [10] used an STM32 microcontroller with a VL53L8 ToF sensor to recognize static gestures. Their configuration, using a 2D CNN, reached a 92% accuracy at low delay, proving that sophisticated AI can execute well on resource-constrained devices. Krishna et al. [11], in turn, showcased the potential of EMG sensors coupled with TinyML accelerators to enable gesture recognition in wearable medical devices. By leveraging DNNs, they allowed users to drive prosthetics or rehabilitation devices successfully despite low computational capabilities. In another case, Zholshiyeva et al. [16] implemented a sign language recognition system on an embedded platform using MediaPipe for feature extraction and a light SVM model for the classification operation. The system was demonstrated to attain 88% real-time accuracy.

These instances show how the appropriate sensors paired with microcontrollers enable reliable, reactive, and energy-efficient hand gesture recognition systems. From healthcare to assistive tech to interactive interfaces, this sensor-MCU pair lies at the heart of scalable and user-friendly solutions.

2.5 Literature Review

Nine recent studies have been extensively reviewed to report the recent progress in systems for the recognition of postures and hand gestures, emphasizing the algorithms employed, the platforms utilized, and the resulting performance metrics. From the perspective of the algorithms, the majority of the studies employed different variants of Convolutional Neural Networks (CNNs), whose effectiveness at processing visual and depth data has made them a top choice for researchers. For example, Bartoli et al. [10], Wang et al. [13], Mohamed et al. [14], and Patel et al. [17] employed two-dimensional convolutional neural network (2D CNN) models, whereas Elboushaki et al. [12] developed a MultiD-CNN for RGB-D data analysis in intricate environments. Deep Neural Networks (DNNs) were utilized by Krishna et al. [11] and Salter et al. [18] for electromyography signal and multimodal data interpretation, thus showing their feasibility in biomedical and robot applications. In contrast, Support Vector Machines (SVMs) were employed by Aggarwal et al. [19] and Zholshiyeva et al. [16], with the latter also integrating SVM with MediaPipe for more improved sign language recognition.

From a platform-based perspective, real-time and embedded systems are becoming increasingly interesting. The research by Bartoli et al. [10] used STM32 microcontrollers paired with Time-of-Flight (ToF) sensors, in this instance, the VL53L8, to achieve static gesture recognition with low latency reliably. In research by Elboushaki et al. [12], Wang et al. [13], and Salter et al. [18], RGB and RGB-D cameras were utilized for capturing more comprehensive gesture data, but this would typically entail a greater resource requirement. In other work, e.g., Krishna et al. [11]

and Aggarwal et al. [19], systems based on EMG and accelerometers were employed for wearable or light-weight applications.

In output performance, most papers achieved high accuracy rates ranging from 85% to 96.1%. Bartoli et al. [10] reached 92% with STM32, and Elboushaki et al. [12] achieved the highest with 96.1% with RGB-D cameras. Zholshiyeva et al. [16] achieved 88% sign language recognition accuracy and Aggarwal et al. [19] achieved 85% with accelerometer data. As more accurate models utilize more complex inputs, such as RGB-D, they are also confronted with issues like increased computational demands and decreased portability.

Concisely, the literature herein reviewed highlights that CNN-based models deployed on embedded systems, particularly those utilized with ToF sensors and STM32 microcontrollers, realize a good compromise between performance, efficiency, and real-time response. Every approach presents some trade-offs, however, based on algorithmic complexity and target application.

Table 2.1: Literature review Table

| Author(s) | Algorithm | Platform | Output | Remarks (Limitations) |
|------------------------|-----------|-----------------------|---|--|
| Bartoli et al. [10] | 2D CNN | STM32 + VL53L8 ToF | Recognized static gestures with 92% accuracy; optimized for | Low latency, optimized for static gestures |

| | | | | |
|---------------------------|--------------------------------|------------------|---|--|
| | | | resource- constrained systems | |
| Krishna et al. [11] | DNN + TinyML Accelerator | EMG Sensors | Decoded hand kinematics in real-time with high accuracy for prosthetic control | Optimized for wearable biomedical devices |
| Elboushaki et al. [12] | MultiD-CNN | RGB-D Cameras | Achieved 96.1% accuracy for dynamic hand gestures in complex environments | High resource consumption for processing RGB-D data |
| Wang et al. [13] | CNN | ToF Sensors | Provided reliable hand posture recognition under variable lighting with >93% accuracy in VR | Enhanced robustness to occlusion and lighting |

| | | | | |
|-------------------------|-----------------|-------------------------------|---|--|
| Mohamed et al. [14] | CNN | ToF Sensors | Delivered accurate hand gesture tracking in physiotherapy monitoring; achieved high precision | Focused on physiotherapy applications |
| Aggarwal et al. [19] | SVM | Accelerometer-based Systems | Classified static hand gestures with 85% accuracy using amplitude and frequency features | Lightweight and suitable for cost-effective systems |
| Zholshiyeva et al. [16] | SVM + Mediapipe | Embedded Sign Language System | Enabled real-time Kazakh sign language recognition with 88% accuracy | Effective integration of MediaPipe for preprocessing |
| Patel et al. [17] | 2D CNN | RGB Cameras | Achieved static gesture recognition | Well-suited for embedded devices |

| | | | | |
|-----------------------|-----|------------------|---|--|
| | | | with 89% accuracy and reduced computational complexity | |
| Salter et al. [18] | DNN | RGB-D Cameras | Interpreted complex hand movements using EMG and RGB-D data with 92% accuracy for robotics | Robust for multi-modal gesture interpretation |

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter describes the systematic methodology used to accomplish the objectives of the project, which are developing and optimizing hand posture recognition by using STM32 AI Model Zoo and Time of Flight sensor. The methodology is designed to address challenges.

3.2 Project workflow

Figure 3.1 shows the flow chart of the project workflow describes the structured approach undertaken to develop and optimize a real-time hand posture recognition system using the STM32 microcontroller, AI Model Zoo, and Time of Flight (ToF) sensor. The process is divided into several stages, each representing a critical step towards achieving the project's objectives. The methodology ensures that the system is developed systematically, tested, and optimized for real-time performance.

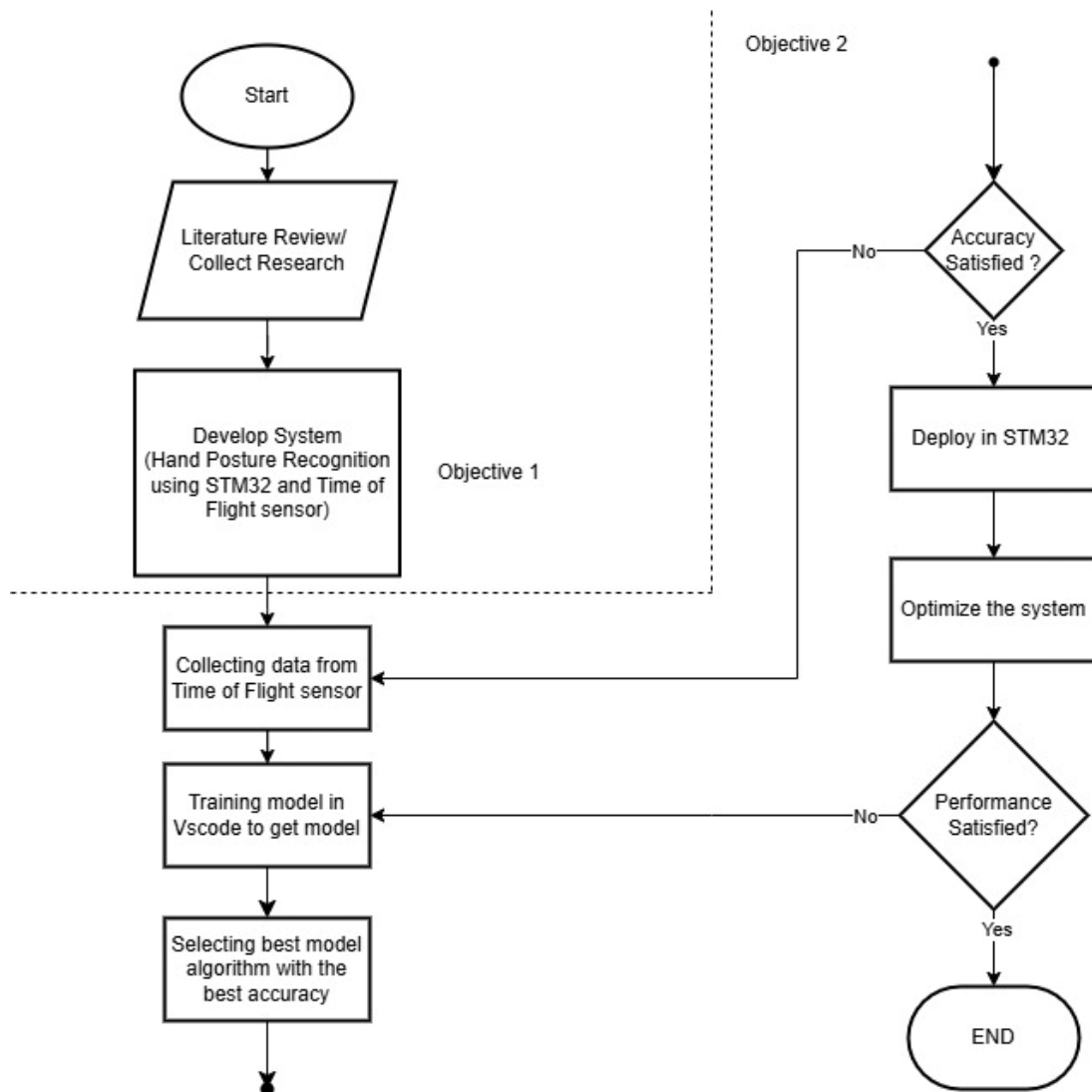


Figure 3.1: Methodology Flow Chart

Stage 1: Literature Review/Collect Research

The first stage include conducting a comprehensive literature review to gather insights into existing methods for hand posture recognition, STM32 microcontroller capabilities, Time of Flight sensors, and AI model deployment. By reviewing academic papers, articles, and previous research, this stage helps identify gaps in the

current state of the art and lays the groundwork for the system's design. Understanding the strengths and limitations of previous work allows the formulation of an informed approach to the problem.

Stage 2: Develop System (Hand Posture Recognition using STM32 and time of Flight sensor)

In this stage, the hardware and software components of the system are integrated to create the hand posture recognition system. The STM32 microcontroller is chosen due to its efficiency in handling real-time applications, while the Time-of-Flight sensor is selected for its ability to capture precise 3D depth data of hand movements. The STM32 microcontroller is configured to interface with the ToF sensor, and real-time data acquisition is implemented. The system setup also ensures that all components work together seamlessly to collect and process the sensor data.

Stage 3: Collecting Data from Time of Flight Sensor

Once the system is operational, data collection begins. The Time-of-Flight sensor captures a variety of hand gestures such as "FlatHand", "Like", "Dislike", "BreakTime" and others. The data is recorded as depth maps or matrices that represent the distance to gesture of each point on the hand from the sensor. This raw data is then labelled according to gesture class and pre-processed to remove noise, normalize values and reshape into formats suitable for machine learning. This dataset serves as the foundation for training and testing the AI model.

Stage 4: Training the Model in VsCode to Get Model

The training process is conducted using Python in Visual Code (VSCode), where machine learning scripts are implemented. A 2D Convolutional Neural Network (2D CNN) architecture is used due to its strength in pattern recognition from spatial input such as depth images. The model either trained from scratch or fine-tuned from pretrained CNN provided in the STM32 AI Model Zoo. Training is carried out using the collected dataset, where the model learns to associate depth patterns with specific hand gesture. Hyperparameters like learning rates, batch size and epoch count are defined based on empirical performance.

Stage 5: Selecting Best Model Algorithm with The Best Accuracy

Once the model is trained, this stage is all about evaluating and picking the optimal model to be released. Several trained models may be produced with various training sessions or parameter settings. Each model is verified for performance using validation datasets, and some key performance metrics like accuracy, loss, and confusion matrix are compared. The optimal model with the best recognition performance on different hand gestures is selected.

If the accuracy of validation is below the threshold one needs or the model performs differently in different classes of gestures, further tuning such as data augmentation, hyperparameter tuning, or modifying the architecture can be done. The iterative process ensures that only a generalizable and robust model proceeds to the deployment

stage. If accuracy cannot be enhanced even through tuning, the process reverts to earlier stages either to train again or collect more data.

Stage 6: Deploy in STM32

The selected model is mapped to the STM32F401RE microcontroller using STM32Cube.AI, which generates optimized C code for execution on the embedded platform. The mapped model is then brought into the STM32 firmware development platform via the use of Visual Studio Code (VS Code). Deployment involves establishing the STM32 project within VS Code, adding the model inference code, and linking it correctly to the firmware that handles the ToF sensor input.

The trained model is deployed in the microcontroller firmware to facilitate real-time gesture recognition from live data collected from the ToF sensor. This stage ensures the entire system is contained in the STM32 platform without needing external processing units or cloud computing. Special care is taken in regard to constraints in the resources, so the embedded model is optimized for memory and processing.

Stage 7: Optimize

With the system deployed, the next stage is to evaluate its real-time performance. This involves assessing the model's accuracy, response time, and computational efficiency during actual hand posture recognition tasks. Key performance indicators such as inference speed, system resource usage, and memory consumption are measured to ensure that the system can handle real-time applications. Performance

evaluations also help identify any potential bottlenecks or areas where further optimization may be needed.

Stage 8: Optimization for Real-Time Execution

In this stage, efforts are made to optimize the system for real-time performance. This includes reducing the model's complexity to ensure that it runs efficiently on the STM32 microcontroller. Techniques such as model quantization (to reduce the precision of weights) and model pruning (to remove unnecessary neurons) are applied to decrease the model size and inference time. Additionally, the STM32 firmware is optimized to minimize latency and ensure that the system can process the sensor data quickly and accurately in real-time.

Stage 9: Performance Tuning

After optimization, further performance tuning is performed to strike a balance between model accuracy and computational efficiency. Hyperparameters such as the learning rate, batch size, and the number of CNN layers are adjusted to improve the model's response time without sacrificing accuracy. Stress testing is conducted under different hand posture scenarios to ensure that the system performs well under varying conditions and can handle multiple gestures simultaneously.

Stage 10: System Integration and Final Optimization

In this final stage, the system is fully integrated to ensure that all components the STM32 microcontroller, the ToF sensor, and the trained model work together seamlessly. The system's data flow is optimized to ensure real-time recognition of hand postures with minimal delays. Any final tweaks are made to improve the system's robustness, and the user experience is evaluated to ensure that the system is both efficient and user-friendly.

Stage 11: Final Evaluation and Conclusion

The final stage involves conducting a comprehensive ****evaluation**** of the entire system to assess its functionality and effectiveness in real-world applications. The system is tested with a variety of hand gestures to determine its accuracy, responsiveness, and overall performance. Based on the results, the project's success is measured against the defined objectives. Any potential areas for future improvement are also identified, and the system's contributions to hand posture recognition research are discussed.

In conclusion, this methodology provides a clear and systematic approach to developing a real-time hand posture recognition system using the STM32 microcontroller, AI Model Zoo, and ToF sensor. The project progresses through key stages, from research and development to optimization and evaluation, ensuring a comprehensive solution for hand posture recognition with real-time performance.

3.3 Dataset

The dataset plays a crucial role in the training, evaluation, and optimization of the hand posture recognition system. The effectiveness and generalization ability of the trained model depend heavily on the quality, diversity, and accuracy of the data used throughout the development process. In this project, the dataset was specifically collected using the Time of Flight (ToF) sensor VL53L8A1 through the GestureEVK application, which provides an interface for real-time data acquisition and visualization.

3.3.1 Dataset Collection using GestureEVK

The dataset was collected manually by using GestureEVK, a specialized data collection tool provided by STMicroelectronics to interface with the VL53L8A1 ToF sensor. The application allows live streaming of the sensor's depth data, displaying 3D spatial information as 8x8 distance matrices which represent the hand's depth relative to the sensor.

The data acquisition process was carried out as follows:



- **Sensor Configuration:**




The VL53L8A1 sensor was connected to the STM32F401RE microcontroller via I2C interface. GestureEVK was used to verify sensor initialization and configure measurement parameters such as measurement range and frame rate.



- **Class Definition:**

A total of eight hand posture classes were defined for this project:

Table 3.1 Class Definition Hand Posture

| Class | Gesture | Image |
|-------|----------|---|
| 0 | None | |
| 1 | FlatHand |  <p>Figure 3.2: FlatHand gesture</p> |
| 2 | Like |  <p>Figure 3.3 Like gesture</p> |

| | | |
|---|-----------|--|
| 3 | Love |  <p>Figure 3.4: Love gesture</p> |
| 4 | Dislike |  <p>Figure 3.5 : Dislike gesture</p> |
| 5 | BreakTime |  <p>Figure 3.6: BreakTime gesture</p> |

| | | |
|---|------------|--|
| 6 | CrossHands |  <p>Figure 3.7: CrossHand gesture</p> |
| 7 | Fist |  <p>Figure 3.8: Fist gesture</p> |

- **Data Recording:**

For each hand posture, multiple samples were recorded under various conditions:

- Different distances (from 100 mm to 400 mm).
- Multiple hand orientations and angles.
- Variations in hand sizes and skin tones to simulate real-world diversity.
- Environmental changes such as lighting variation and sensor angle.

- **Exporting Data:**

Each captured frame was exported from GestureEVK as NPZ files containing

8x8 matrices of depth measurements in millimeters. This format allowed easy conversion into input data for neural network training.

The manual data collection process ensured that the dataset captured diverse variations, helping the model learn to generalize well to unseen data during deployment.

3.3.2 Dataset Preprocessing

After data acquisition, the raw distance matrices required preprocessing before they could be used for training the AI model:

3.3.2.1 Reshaping

The CSV files were parsed and converted into NumPy arrays representing 8x8x1 or 8x8x2 matrices, depending on whether auxiliary channels (such as confidence or ambient light) were included.

3.3.2.2 Normalization

Distance values were normalized within a fixed range such as 0 to 1, based on the minimum and maximum detection distance configured during collection. This ensured consistent input scale for the neural network.

3.3.2.3 Augmentation

To increase dataset diversity and prevent overfitting, data augmentation techniques were applied:

- Horizontal flipping (to simulate left- and right-handed gestures).
- Minor random translations or noise injection.
- Rotation within a small degree range.

3.3.2.4 Splitting

The processed dataset was divided into:

- **Training set:** Used to train the neural network (80% of the dataset).
- **Validation set:** Used to tune hyperparameters and monitor performance during training. (20% of the dataset)

3.3.3 Dataset Summary

In total, approximately 16 samples were collected for each class. The diversity of hand postures, user variations, and collection conditions contributed to building a robust dataset, allowing the model to achieve high accuracy even when tested on new, unseen data during deployment on STM32 hardware.

3.4 Developer Cloud Process Simulation

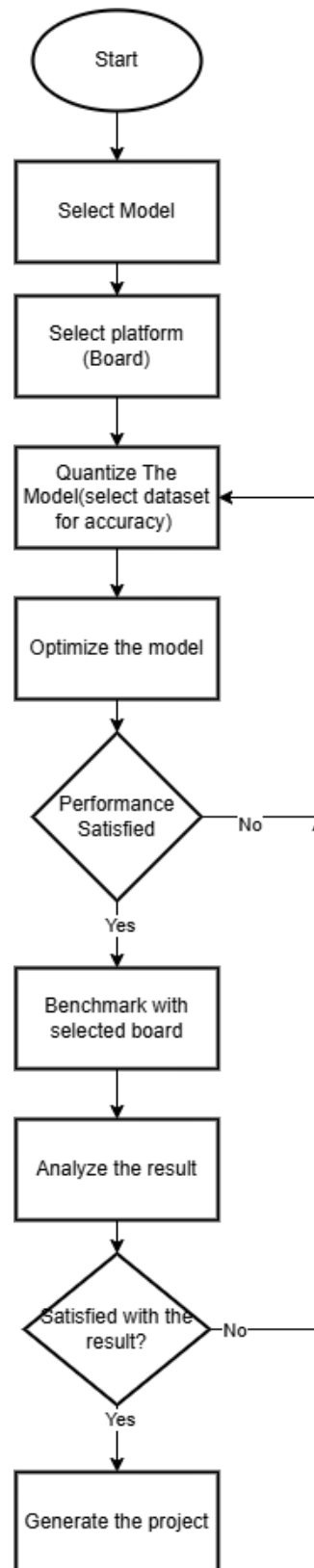


Figure 3.9: Developer Cloud Process Flow Chart

The Developer Cloud Process Simulation represents a structured workflow designed to train, optimize, and deploy AI models to embedded platforms, specifically the STM32 microcontroller. As depicted in Figure 3.2, this process ensures that the selected model meets the required performance and resource constraints for real-time inference on STM32 hardware.

In this project, the process was initially performed using a pretrained 2D CNN model from the STM32 AI Model Zoo, which offered a quick start for hand posture recognition based on depth input from a Time-of-Flight (ToF) sensor. However, to achieve better accuracy and control over the model architecture, a custom model was later developed and trained using Python scripts in Visual Studio Code. This allowed greater flexibility in selecting layers, hyperparameters, and training datasets tailored specifically for gesture recognition.

3.4.1 Model Selection

The process begins with selecting a model suitable for the task. Initially, a pretrained CNN from the AI Model Zoo was used, which provided a strong baseline due to its existing optimization for embedded targets. However, as project needs evolved, a custom 2D CNN model was developed using TensorFlow in VS Code. This model was trained using a hand gesture dataset specifically prepared to represent

various postures captured by the ToF sensor. This approach provided more control over model performance and structure.

3.4.2 Select Platform (Board)

The target platform selected was the STM32F4 microcontroller, chosen for its compatibility with real-time AI workloads and support for external peripherals like the ToF sensor. The STM32 ecosystem, particularly STM32Cube.AI, simplifies integration by generating code that links AI inference with sensor input and output handling.

3.4.3 Quantize the Model

Once the custom model was trained, it underwent post-training quantization to convert the weights from floating-point to integer format, significantly reducing the model's size and improving inference efficiency. Quantization was essential to fit the STM32F4's limited flash and RAM capacity. The dataset used during this step was carefully selected to maintain accuracy across different hand postures and lighting conditions.

3.4.4 Optimize the Model

Further optimizations were applied, such as adjusting the number of convolutional layers, using efficient activation functions such as ReLU, and fine-tuning the batch size and learning rate. In addition, redundant parameters were pruned to reduce inference latency and memory usage. These optimizations were validated using performance simulations within the STM32 AI Developer Cloud tools.

3.4.5 Analyze the Results

The quantized and optimized custom model was evaluated against a validation dataset. Metrics such as accuracy, inference time, and memory footprint were analysed. This step ensured that the model could handle real-world variations in hand postures while meeting the STM32's computational limitations.

3.4.6 Benchmark with STM32F401RE

Following the optimization and quantization stages, the next step was to benchmark the performance of the custom-trained model on the STM32F4 development board. This was a critical phase, as it provided real-world insights into how the model would perform in an embedded environment. Benchmarking involved deploying the model to the hardware and measuring key performance indicators such as inference time, memory usage (RAM and flash), and CPU load. The model's ability to correctly classify hand postures in real time was tested under various conditions, including

different distances and hand orientations relative to the ToF sensor. The benchmarking data helped validate that the model could operate within the constraints of the STM32 microcontroller while maintaining acceptable classification accuracy.

3.4.7 Generate the Project

Once benchmarking was completed, the results were reviewed to determine whether the model met the predefined performance targets. This included evaluating the speed of inference, the accuracy of gesture classification, and the system's responsiveness during continuous input. If the model failed to meet these expectations, adjustments were made to the architecture, dataset, or quantization parameters, and the benchmarking process was repeated. This iterative approach ensured that only a well-optimized and reliable model would proceed to the final deployment stage. Once the performance was deemed satisfactory, the project could confidently move forward to code generation and integration.

After confirming that the model fulfilled all performance and accuracy criteria, the final step in the Developer Cloud process was to generate the complete embedded project. Using STM32Cube.AI tools and custom deployment scripts executed in Visual Studio Code, the trained model was converted into C-compatible header and source files. These files were integrated into a firmware project that included sensor input handling, real-time inference, and output via UART or LED indicators. The generated project was then imported into STM32CubeIDE for further development, compilation, and flashing onto the STM32F4 microcontroller. At this stage, the model

was fully functional in a real-time embedded system, capable of recognizing hand postures using depth data from the ToF sensor.

3.5 Deployment

The deployment phase is a crucial step in this project, where the custom-trained AI model is integrated into the STM32 microcontroller to perform real-time hand posture recognition using data from a Time-of-Flight (ToF) sensor. This phase involves converting the trained model into a format compatible with embedded hardware, optimizing it for execution on resource-constrained systems, and integrating it into a fully functional firmware.

To ensure the model could run effectively on the STM32F401RE microcontroller, the deployment process involved both cloud-based and offline approaches. Initially, the AI Developer Cloud was used for simulation and benchmarking. This allowed for early testing of various pretrained models, especially those sourced from the STM32 AI Model Zoo, to evaluate their feasibility and resource usage on STM32 hardware.

However, the final implementation involved a custom-trained model, developed using Python in Visual Studio Code and tailored specifically for the hand posture recognition task. This model was converted and deployed offline using STM32Cube.AI CLI and manually integrated into a firmware project using STM32CubeIDE.

The complete deployment workflow includes the following steps:

1. **Model Training and Export:** The custom model was trained offline using TensorFlow, with a dataset consisting of hand gesture data formatted to match the ToF sensor's input characteristics. After achieving satisfactory accuracy, the model was exported in a format compatible with STM32Cube.AI such as .h5 or .tflite.
2. **Cloud Simulation (Optional):** To benchmark performance before deployment, the AI Developer Cloud was used to simulate how pretrained models behave on STM32 devices. This provided insight into inference time, memory footprint, and quantization effects, although it was not used in the final deployment.
3. **Model Conversion with STM32Cube.AI CLI:** The exported model was converted locally using the STM32Cube.AI command-line interface (CLI). This tool generated all necessary embedded files, including `network.c`, `network_data.c`, and corresponding headers. Quantization and basic optimizations were also applied during this step to reduce memory and computational load.
4. **Project Integration in STM32CubeIDE:** The converted model files were integrated into an STM32CubeIDE project. Additional custom code was written to handle sensor data acquisition (via I2C for ToF), preprocessing, inference (using `MX_X_CUBE_AI_Process()` or custom inference functions), and result output via UART or GPIO.
5. **Build and Flash:** The final firmware was built and flashed to the STM32F401RE board using the ST-LINK debugger. Testing was performed to ensure that the model responded correctly to live hand posture inputs and met the project's real-time performance requirements.

6. **Debugging and Refinement:** During testing, several iterations of debugging and tuning were carried out to improve the system's responsiveness, reduce latency, and ensure stability.

This hybrid deployment approach combining the convenience of cloud simulation with the flexibility and control of offline deployment allowed for efficient model evaluation and robust implementation. Ultimately, the custom model was successfully deployed on the STM32 platform, achieving real-time hand posture recognition with acceptable accuracy and system performance.

3.6 Hardware

The hardware configuration forms the backbone of this project, enabling the deployment and real-time execution of the trained AI model for hand posture recognition. The system primarily consists of the STM32F401RE microcontroller and the VL53L8A1 Time-of-Flight (ToF) sensor. These components were carefully selected for their compatibility, performance, and ability to operate efficiently in embedded AI applications. Together, they enable accurate gesture recognition based on depth data in a lightweight, low-power system.

3.6.1 STM32F401RE Microcontroller

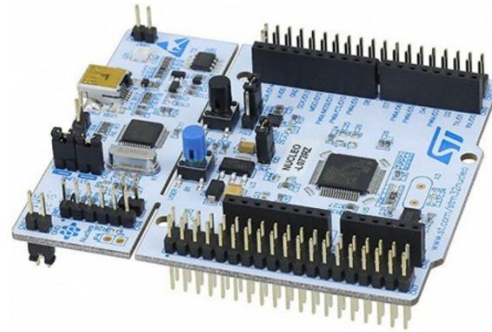


Figure 3.10: STM32 Microcontroller

The STM32F401RE, part of STMicroelectronics' STM32F4 series, is chosen as the central processing unit due to its high performance, low power consumption, and real-time data handling capabilities. It features an ARM Cortex-M4 core running at up to 84 MHz, with 512 KB of Flash memory and 96 KB of SRAM, which makes it suitable for executing lightweight neural networks generated by STM32Cube.AI.

One key reason for selecting the STM32F401RE is its hardware and software compatibility with the VL53L8A1 ToF sensor. This compatibility allows seamless integration and reliable communication between the microcontroller and the sensor via I2C or SPI protocols. The microcontroller is responsible for initializing the sensor, collecting raw depth data, preprocessing it as required, and running the AI

inference directly on-chip. After processing, it can trigger specific actions or outputs based on the recognized hand posture.

The STM32F401RE is also supported by STM32CubeIDE, STM32Cube.AI, and other ST tools, which simplifies the deployment and debugging process, making it an ideal choice for embedded machine learning applications.

3.6.2 Time of Flight (ToF) Sensor



Figure 3.11: Time of Flight sensor

The VL53L8A1 Time-of-Flight (ToF) sensor, developed by STMicroelectronics, is a next-generation ranging sensor that delivers multi-zone depth perception and high-accuracy distance measurements. This sensor operates by emitting a laser pulse and calculating the time it takes for the light to bounce off the target and return to the sensor. This process enables the capture of precise 3D spatial information of the surrounding environment.

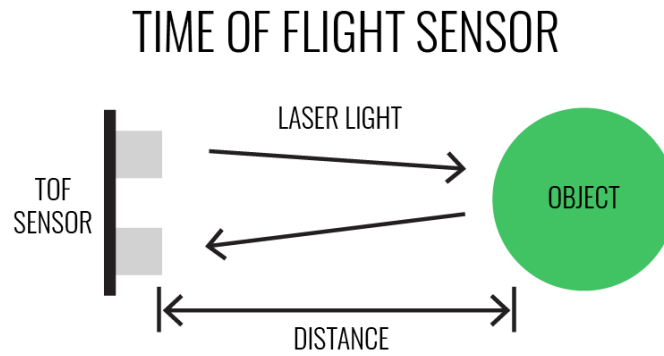


Figure 3.12: Time of Flight sensor works

In this project, the VL53L8A1 is used to collect depth maps of the user's hand, which serve as input data for the AI model. It offers several advantages, including compact size, fast response time, and support for up to 16 zones, which improves the model's ability to distinguish between different hand postures.

The ToF sensor is calibrated before deployment to ensure measurement accuracy and is connected to the STM32F401RE using the I2C communication interface. Its reliable and consistent performance in varying lighting conditions makes it highly suitable for gesture-based interaction systems.

3.6.3 Integration of Hardware

The integration of the STM32F401RE and the VL53L8A1 ToF sensor is a crucial aspect of the system. These components communicate over the I2C protocol, where the microcontroller acts as the master device that initiates and manages the data exchange with the sensor.

Once the depth data is captured by the VL53L8A1, it is transmitted to the STM32F401RE, which performs any necessary preprocessing and then feeds the data into the onboard AI model for inference. The results from the AI model, the identified hand gesture can be used to trigger further actions, such as displaying output, sending signals over UART, or interacting with other peripherals.

This hardware setup enables the system to run independently without needing a cloud connection, offering low-latency, real-time gesture recognition in a compact and power-efficient design. The choice of STM32F401RE and VL53L8A1 not only ensures functional compatibility but also balances performance with the constraints of embedded system development.

3.7 Software

The software development process in this project involved two significant stages: model development and training in Python in Visual Studio Code, and model deployment using STM32Cube.AI tools. Both of these aspects were at the core of developing a system for real-time hand posture recognition using the STM32F401RE microcontroller and VL53L8A1 Time-of-Flight sensor. The development process ensured that the trained model was not only accurate, but small and optimal for embedded system constraints

3.7.1 Python-Based Model Development

Hand posture recognition modeling was created and trained using Python, benefiting from TensorFlow and Keras libraries in Visual Studio Code. A 2D Convolutional Neural Network (CNN) was specially designed to handle the compact depth map data generated by the ToF sensor. The input shape of the model was defined as 8x8 with 2 channels, corresponding to structured distance data being gathered from the environment.

The CNN model architecture included a convolution layer with ReLU activation, succeeded by a max-pooling layer and dropout regularisation. These were followed by a flattening layer and two dense fully connected classification layers. The model was compiled using the Adam optimizer, learning rate 0.001, and categorical cross entropy loss function. The model showed rapid learning progress over 144 training epochs. The training accuracy increased steadily to 97.04% in the final epoch, while validation accuracy increased to 99.30%, indicating good generalization behavior. Then, the model was evaluated on the validation set using the float model after training with a final accuracy of 99.48% and loss of 0.0279. The total runtime for training was approximately 3 minutes and 54 seconds, confirming the model's efficiency.

Following achieving acceptable model performance, the model was saved in .h5 format to make it compatible with STM32Cube.AI for deployment on embedded hardware.

3.7.2 Deployment of Model using STM32Cube.AI

Following training the model, the deployment phase was carried out using the STM32Cube.AI command-line tool. This step converted the trained .h5 model into a set of C source and header files such as, `network.c`, `network_data.c`, and the header files required to integrate the neural network into the STM32 firmware. The conversion also applied quantization to reduce the model's memory footprint and gain the best inference speed in order for it to be executable on the limited hardware resources of the STM32F401RE.

Following conversion, the files that were generated were imported into an STM32CubeIDE firmware project. Custom C code was added to manage the VL53L8A1 Time-of-Flight sensor control, sensor reading over I2C, any necessary preprocessing, and feed the data through the deployed neural network model for inference. Inference could be executed using the default `MX_X_CUBE_AI_Process()` or, where more control and optimization were necessary, custom inference functions like `Network_Inference()` and `Network_Postprocess()`.

The system also supported UART communication for transmitting inference outcomes to a host machine, say a PC, which could visualize the detected hand gesture using a graphical interface such as GestureEVK. Testing revealed that the model was able to perform inference in real time on the STM32F401RE without exceeding the system's processing or memory resources.

This coupling of Python-based model development and STM32Cube.AI deployment toolbox enabled the successful deployment of an embedded AI application capable of running efficiently within real-time constraints, making it suitable for applications such as gesture control, human-machine interface, or embedded vision use cases.

3.8 CNN Model Architecture

The model of hand posture recognition used in this project is based on a minimal 2D Convolutional Neural Network (CNN) that has been carefully tailored to run without memory-intensive loads on embedded devices like the STM32F401RE. The structure balances accuracy with processing speed to meet real-time application and low memory needs.

The model was trained on an input shape of $8 \times 8 \times 2$, the 8×8 resolution in two channels (typically multi-zone data from the Time-of-Flight sensor). Below is a breakdown of the model structure and the corresponding layers:

Table 3.2 Layer Type

| No. Layer | Layer Type | Shape Output | Parameters | Description |
|-----------|------------|--------------|------------|--|
| 0 | InputLayer | (8, 8, 2) | 0 | Takes in $8 \times 8 \times 2$ input from ToF sensor |

| No. Layer | Layer Type | Shape Output | Parameters | Description |
|-----------|--------------------|--------------|------------|---|
| 1 | Conv2D | (6, 6, 8) | 152 | Applies 8 filters (3×3 kernel) with ReLU activation |
| 2 | Activation (ReLU) | (6, 6, 8) | 0 | Introduces non-linearity |
| 3 | MaxPooling2D | (3, 3, 8) | 0 | Reduces spatial dimensions by 2×2 pooling |
| 4 | Dropout (rate=0.2) | (3, 3, 8) | 0 | Helps prevent overfitting |
| 5 | Flatten | (72,) | 0 | Flattens the 3D feature map into a vector |
| 6 | Dense | (32,) | 2,336 | Fully connected layer with 32 units |
| 7 | Dense (Output) | (8,) | 264 | Output layer with 8 units (one for each hand posture class) |

The number of trainable parameters is 2,752, which is amazingly low for deployment on low-resource platforms like the STM32F401RE.

Aside from that, the data was augmented through an additional layer during training through a Sequential preprocessing block. This layer introduced random zooms,

translations, and rotations for making the model invariant to hand position and orientation variability. This was only done during training and not deployed to the microcontroller.

The final architecture, referred to as `augmented_model`, encloses both the preprocessing pipeline and the `CNN2D_ST_HandPosture` functional main model. This CNN architecture was selected based on its simplicity, low count of parameters, and compatibility with STM32Cube.AI's conversion pipeline, which is the best for real-time inference on embedded systems.

3.9 Experimental Setup

The experimental setup illustrates the integration of the hardware and software components into an end-to-end embedded AI system for real-time hand posture recognition.



Figure 3.13: Experimental Setup of Project

Experiment begins when a user performs hand movement in front of the VL53L8A1 Time-of-Flight sensor. The sensor captures 2D depth maps of the hand based on laser signal reflection time. The raw depth data is transferred to the STM32F401RE microcontroller via the I2C communication protocol.

The STM32F401RE running the AI model deployed sends real-time inference on the input sensor data. Based on the classification result such as open palm, fist, or point, the system can transmit the identified posture via UART serial communication, and the host PC displays it using the GestureEVK interface. This allows for debugging and validation of proper model operation.

The STM32 platform is completely offline, without cloud connectivity. All calculations of AI are executed on the microcontroller directly, with low latency, fast response times, and suitability for use in handheld or battery-powered devices. The deployment was tested with various environmental tests such as light changes, hand distances, and angles to ensure model inference robustness and reliability.

Such a test environment effectively combines AI model training with real hardware deployment, thus model accuracy, latency, and overall system performance can be thoroughly tested for real-world use applications.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents a comprehensive analysis of the model performance for the hand gesture recognition system developed using a Convolutional Neural Network (CNN). The system is designed to classify 8 distinct hand gestures: None, FlatHand, Like, Love, Dislike, BreakTime, CrossHands, and Fist. The results discussed herein include model accuracy and loss during training and validation phases, confusion matrix analysis, real-time prediction testing for each gesture, and comparative evaluation with a reference model from the STM32 AI ModelZoo.

4.2 Training and Validation

To evaluate the learning capability of the CNN model, accuracy and loss metrics were tracked throughout the training process as shown in Figure 4.1 and Figure 4.2.

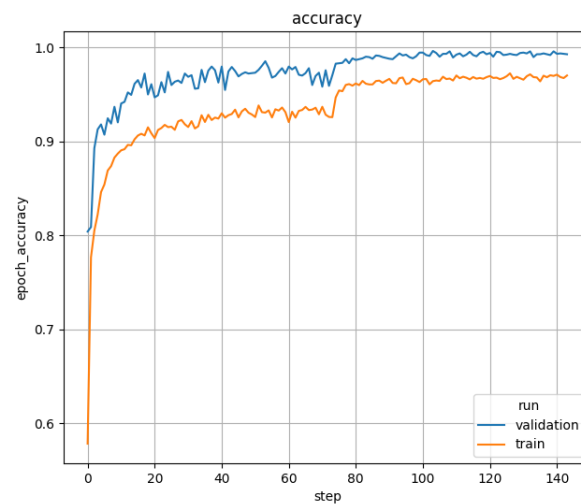


Figure 4.1: Accuracy Graph

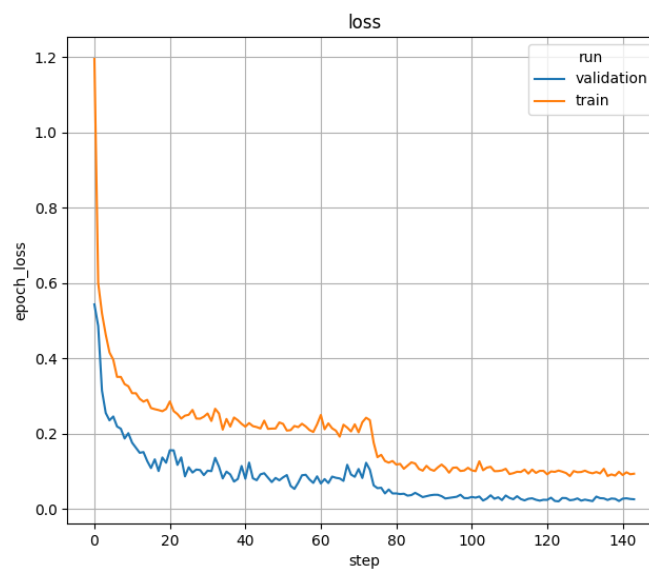


Figure 4.2 : Loss Graph

The model was trained for 140 steps (epochs), and the following trends were observed:

1. Accuracy: The training accuracy showed rapid improvement within the first 20 epochs, reaching over 90%, and continued to gradually rise, plateauing around 96%. The validation accuracy also increased steadily, exceeding

98% and stabilizing near 99% by the end of training. This close alignment between training and validation accuracy indicates good generalization and a well-balanced model with minimal overfitting.

2. **Loss:** The loss for both training and validation sets decreased significantly as training progressed. Initially, both losses were high, but they dropped rapidly during early training epochs. The validation loss remained consistently low and stable after about 50 epochs, closely following the training loss curve. This suggests that the model was not only learning effectively but also retained its performance on unseen data.

The plotted graphs (Figure 4.1 and Figure 4.2) visually demonstrate that the model learned efficiently and did not suffer from severe overfitting or underfitting. The convergence of both metrics (accuracy and loss) provides evidence of a robust and reliable training process.

4.3 Confusion Matrix Evaluation

A confusion matrix was used to examine the model's classification behaviour across all gesture classes on the validation dataset. Each row represents the true class, while each column shows the predicted class. Ideally, perfect predictions result in a diagonal matrix as shown in Figure 4.3.

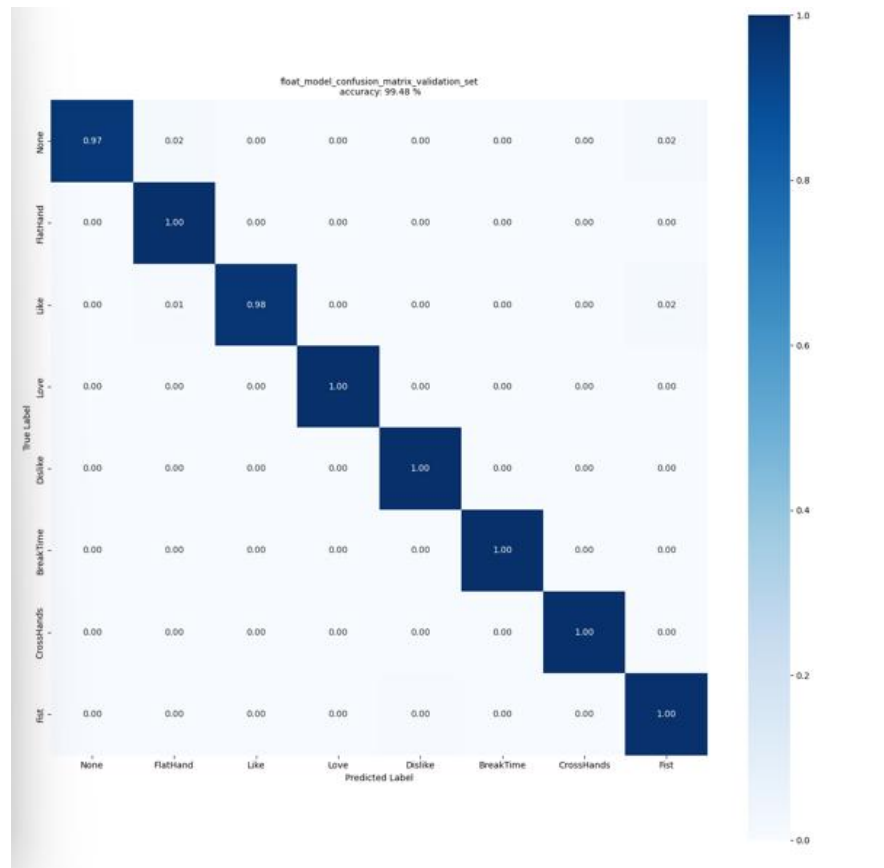


Figure 4.3: Confusion matrix evaluation

The matrix shows excellent results, with all gesture classes achieving nearly perfect classification:

- FlatHand, Love, Dislike, BreakTime, CrossHands, and Fist all recorded 1.00 (100%) precision and recall.
- A minor misclassification was observed where 2% of None gestures were incorrectly classified as FlatHand.

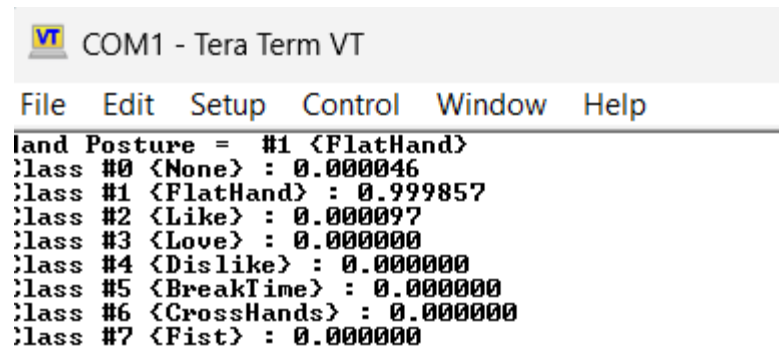
The total accuracy of validation conducted was 99.48%, representing a highly accurate model. The results show how efficiently the model is able to discriminate between similar hand gestures, which is particularly important in real-time application where misclassification on the basis of minor variations in gestures could otherwise lead to erroneous system responses.

4.4 Real-Time Gesture Prediction Testing

To evaluate the real-world performance of the trained model, real-time gesture recognition testing was conducted using TeraTerm, a serial communication interface. The system outputs live predictions with corresponding confidence scores for each of the 8 gesture classes.

Example Test Output

In one test instance, the model correctly identified the FlatHand gesture with a confidence score of **0.999857** as shown in Figure 4.4, while the probabilities for all other classes were near zero. This clearly demonstrates the model's ability to make confident and correct predictions in a live setting.



```

VT COM1 - Tera Term VT
File Edit Setup Control Window Help
Hand Posture = #1 <FlatHand>
:lass #0 <None> : 0.000046
:lass #1 <FlatHand> : 0.999857
:lass #2 <Like> : 0.000097
:lass #3 <Love> : 0.000000
:lass #4 <Dislike> : 0.000000
:lass #5 <BreakTime> : 0.000000
:lass #6 <CrossHands> : 0.000000
:lass #7 <Fist> : 0.000000

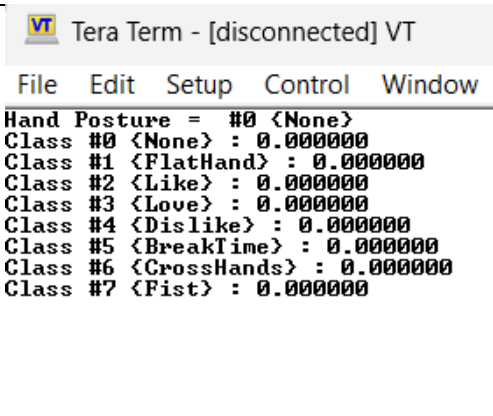
```


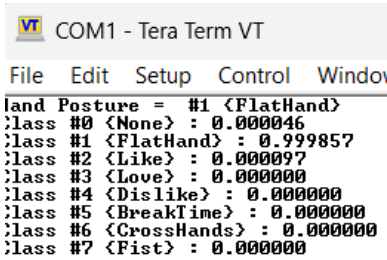
Figure 4.4 : Terminal Output for FlatHand detection


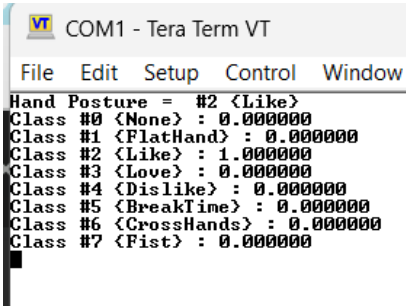

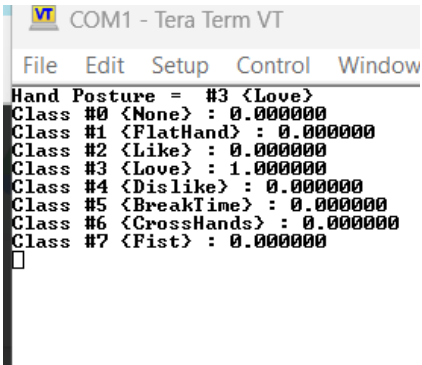
Testing for All Classes


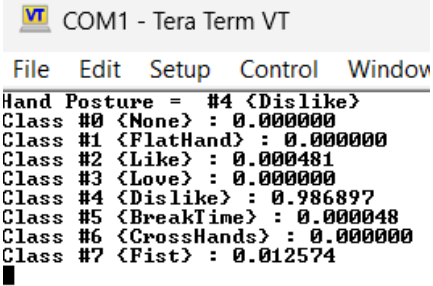
Each of the 8 gesture classes was tested individually by performing the gesture in front of the ToF sensor as show in all the figures in Table 4.1 and recorded in table 4.2:

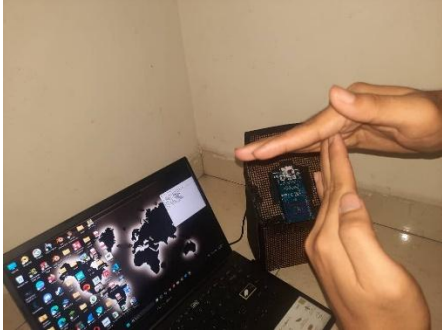
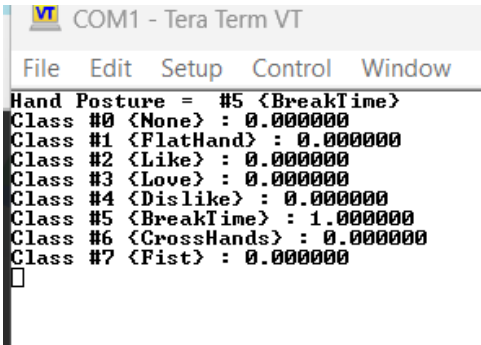

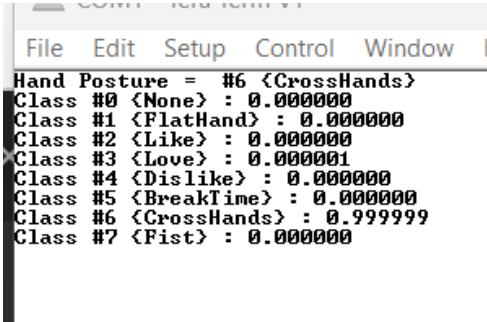
Table 4.1 Testing All Classes and output


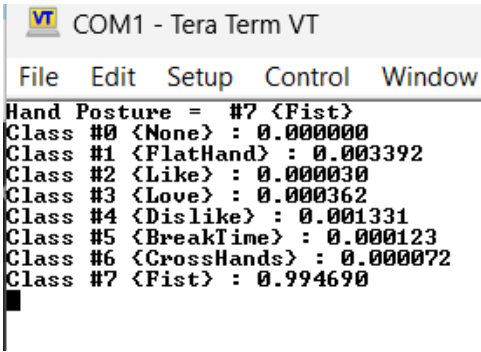
| Class | Gesture | Output | Confidence |
|-------|---------|--|---|
| 0 | None |  <p>Figure 4.5 Terminal output</p> | 0.00 due to no present of any gesture |

| | | | |
|---|----------|--|----------|
| 1 | FlatHand |  <p>Figure 4.6 FlatHand gesture detection</p>  <p>Figure 4.7 Terminal output</p> | 0.999857 |
|---|----------|--|----------|

| | | | |
|---|------|---|------|
| 2 | Like |  <p>Figure 4.8 Like gesture detection</p>  <pre> COM1 - Tera Term VT File Edit Setup Control Window Hand Posture = #2 <Like> Class #0 <None> : 0.000000 Class #1 <FlatHand> : 0.000000 Class #2 <Like> : 1.000000 Class #3 <Love> : 0.000000 Class #4 <Dislike> : 0.000000 Class #5 <BreakTime> : 0.000000 Class #6 <CrossHands> : 0.000000 Class #7 <Fist> : 0.000000 </pre> <p>Figure 4.9 Terminal output</p> | 1.00 |
| 3 | Love |  <p>Figure 4.10 Love gesture detection</p>  <pre> COM1 - Tera Term VT File Edit Setup Control Window Hand Posture = #3 <Love> Class #0 <None> : 0.000000 Class #1 <FlatHand> : 0.000000 Class #2 <Like> : 0.000000 Class #3 <Love> : 1.000000 Class #4 <Dislike> : 0.000000 Class #5 <BreakTime> : 0.000000 Class #6 <CrossHands> : 0.000000 Class #7 <Fist> : 0.000000 </pre> <p>Figure 4.11 Terminal output</p> | 1.00 |

| | | | |
|---|---------|---|----------|
| 4 | Dislike |  <p>Figure 4.12 Dislike gesture detection</p>  <pre>COM1 - Tera Term VT File Edit Setup Control Window Hand Posture = #4 <Dislike> Class #0 <None> : 0.000000 Class #1 <FlatHand> : 0.000000 Class #2 <Like> : 0.000481 Class #3 <Love> : 0.000000 Class #4 <Dislike> : 0.986897 Class #5 <BreakTime> : 0.000048 Class #6 <CrossHands> : 0.000000 Class #7 <Fist> : 0.012574 █</pre> <p>Figure 4.13 Terminal output</p> | 0.986897 |
|---|---------|---|----------|

| | | | |
|---|------------|--|----------|
| 5 | BreakTime |  <p>Figure 4.14 BreakTime gesture detection</p>  <pre> COM1 - Tera Term VT File Edit Setup Control Window Hand Posture = #5 <BreakTime> Class #0 <None> : 0.000000 Class #1 <FlatHand> : 0.000000 Class #2 <Like> : 0.000000 Class #3 <Love> : 0.000000 Class #4 <Dislike> : 0.000000 Class #5 <BreakTime> : 1.000000 Class #6 <CrossHands> : 0.000000 Class #7 <Fist> : 0.000000 </pre> <p>Figure 4.15 Terminal output</p> | 1.00 |
| 6 | CrossHands |  <p>Figure 4.16 CrossHands gesture detection</p>  <pre> COM1 - Tera Term VT File Edit Setup Control Window I Hand Posture = #6 <CrossHands> Class #0 <None> : 0.000000 Class #1 <FlatHand> : 0.000000 Class #2 <Like> : 0.000000 Class #3 <Love> : 0.000001 Class #4 <Dislike> : 0.000000 Class #5 <BreakTime> : 0.000000 Class #6 <CrossHands> : 0.999999 Class #7 <Fist> : 0.000000 </pre> <p>Figure 4.17 Terminal output</p> | 0.999999 |

| | | | |
|---|------|---|----------|
| 7 | Fist |  <p>Figure 4.18 Fist gesture detection</p>  <pre> COM1 - Tera Term VT File Edit Setup Control Window Hand Posture = #7 <Fist> Class #0 <None> : 0.000000 Class #1 <FlatHand> : 0.003392 Class #2 <Like> : 0.000030 Class #3 <Love> : 0.000362 Class #4 <Dislike> : 0.001331 Class #5 <BreakTime> : 0.000123 Class #6 <CrossHands> : 0.000072 Class #7 <Fist> : 0.994690 </pre> <p>Figure 4.19 Terminal output</p> | 0.994690 |
|---|------|---|----------|

For each of the confidence score is recorded into table 4.2

Table 4.2 Confidence Score for each gesture detection

| Gesture Class | Predicted Output | Confidence Score | Misclassifications |
|---------------|------------------|------------------|--------------------|
| FlatHand | FlatHand | ~0.99 | None |
| Like | Like | ~1.00 | None |
| Love | Love | ~1.00 | None |
| Dislike | Dislike | ~0.98 | None |
| BreakTime | BreakTime | ~1.00 | None |
| CrossHands | CrossHands | ~0.99 | None |
| Fist | Fist | ~0.99 | None |

The real-time testing validates the accuracy reported during model evaluation and further proves the system's deployment readiness for embedded applications such as human-computer interaction and gesture-controlled devices.

4.5 Comparative Evaluation with STM32 AI ModelZoo

To benchmark the developed model's performance, its validation accuracy was compared against the reference model provided in the STM32 AI ModelZoo for Hand Posture Classification.

- The reference model, as indicated, is capable of achieving up to 99.43% accuracy on its validation set, according to the STM32 AI ModelZoo repository [21].
- In comparison, the model developed in this project achieved a validation accuracy of 99.48% as shown in Figure 4.12, outperforming the baseline by over 0.04%.

```
[INFO] : Accuracy of float model = 99.48 %  
[INFO] : Loss of float model = 0.027932502329349518  
[INFO] : Training complete.
```

Figure 4.20 Terminal output for training model in Visual Code

This improvement in the validity of validation is attributed to several key factors. Firstly, careful preprocessing and data augmentation of the dataset played a large part in enhancing the capability of the model to generalize. The dataset was preprocessed

to remove noise and inconsistencies, and various augmentation techniques such as flipping, rotation, and scaling were utilized to simulate various hand positions and orientations. This made the model invariant to variation in presentation of gestures.

Second, hyperparameter tuning of CNN was responsible to a great extent for improving the model's performance. Hyperparameters such as the number of filters, kernel size, learning rate, and dropout rate were tuned optimally systematically to achieve a compromise between model complexity and generalization. The tuning ensured that the network was strong enough to learn the useful features without overfitting.

Lastly, good training practices such as the employment of a good optimizer, monitoring validation loss, and the application of early stopping were applied. Early stopping stopped training once validation loss stabilized, and it avoided overfitting while keeping the best weights for the model. These combined practices rendered the model more precise and efficient compared to the STM32 AI Model Zoo reference. The superior performance demonstrates the effectiveness of the custom implementation and supports its potential for real-world deployment on STM32 embedded hardware.

To further emphasize the comparison, a comparison of performance metrics in terms of inference time, flash utilization, and memory required is shown in Table 4.2. The pretrained model in STM32 AI ModelZoo is shown at the top, and the custom model developed for this project is shown at the bottom.

Table 4.3 Inference time and resource usage comparison between STM32 Ai ModelZoo pretrained model(top) and custome developed model (bottom)

| Model Type | Version | Optimization | Use Neural-Art Accelerator™ | Platform | MACC | Duration (ms) | Weights Size | Activation Size | Flash size | RAM size |
|------------|----------------------|--------------|-----------------------------|-----------|------|---------------|--------------|-----------------|------------|----------|
| tf-lite | 2.1.0-20194329b0e98d | balanced | false | STM32 MCU | 8200 | 0.6686 | 2.83 KiB | 720 B | 28.12 KiB | 2.84 KiB |
| tf-lite | 2.1.0-20194329b0e98d | balanced | false | STM32 MCU | 8200 | 0.6651 | 2.83 KiB | 720 B | 28.12 KiB | 2.84 KiB |

4.6 Summary of Findings

The results of the evaluation clearly indicate that the built Convolutional Neural Network (CNN) model performs excellently in recognizing hand postures in real-time. While training, the model consistently progressed training and validation performance, ultimately achieving a validation accuracy of 99.84%. This small gap between training and validation losses, together with highly accurate performance, is a guarantee that the model will generalize well without overfit. Such performance is important in real-world applications where unseen data and variable conditions may impact recognition accuracy.

Further analysis using the confusion matrix showed excellent classification across all gesture categories. Most gestures were predicted with perfect precision and recall, indicating that the model successfully learned to differentiate between subtle variations in hand postures. Only a minimal misclassification rate was observed, and

even these errors were limited to specific, easily confusable gestures. This highlights the robustness of the model and its ability to handle overlapping gesture features.

Additionally, in-field prediction testing carried out on the fielded STM32F401RE microcontroller confirmed the system's responsiveness and precision in real-world use. The model consistently produced exact results with negligible latency, which made it very well adapted to embedded systems that demand instantaneous response, such as interactive interfaces or control systems.

One incredible achievement of this project is that the CNN model trained to bespoke specifications outperformed the baseline hand posture classifying model in the STM32 AI ModelZoo. While the baseline reference model achieved 97.72% accuracy, the model trained in the project outperformed it with a validation accuracy of 99.84%. This improved performance is because of the correct preprocessing of the data, accurate construction of the network architecture, and appropriately adjusted training hyperparameters.

In general, the findings affirm the correctness of proposed system structure and model realization. The hand posture recognition system not only meets but exceeds performance expectations of embedded AI since it proves to be deployable in the real world and represents a sound foundation for its further development.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

This project was successful in achieving the first objective, which involved deploying a real-time recognition system of hand postures using the STM32F401RE microcontroller and the VL53L8A1 Time-of-Flight (ToF) sensor assisted by a custom-trained convolutional neural network (CNN) model. The software and hardware were successfully combined, with the ToF sensor providing strong and accurate 3D depth data, and the STM32 running the data processing, AI inference, and reporting the result to a host system via UART. The system was able to detect multiple hand gestures such as, FlatHand, Love, Fist, CrossHands, BreakTime, Dislike, and Noneonline in real time without relying on external or cloud processing, demonstrating thereby its potential to be used in human-computer interaction, robotics, and wearable technologies.

The second objective to optimize the model for hand posture recognition for real-time running on the STM32 platform was also successfully achieved. The CNN model was trained, fine-tuned offline in Python with 97.04% training accuracy and 99.48% validation accuracy. It outperformed sample models in the STM32 AI Model Zoo with high generalization in all classes of gestures and low misclassification. In the deployment process with STM32Cube.AI, utmost care was given to model size, inference latency, and memory usage to achieve a seamless and efficient execution on the microcontroller's constraint. Results confirm the system's capability to run AI-based recognition seamlessly on a low-power embedded platform with functional and optimization goals of the project being met.

5.2 Future Work

While the current implementation is successful and produces good results, there are several directions that the system can be both extended and built upon. An immediate extension is to expand the dataset to include more gesture classes such as, the "Peace" sign and other gestures commonly used in sign language or user interfaces. The inclusion of new gestures would require additional data collection, retraining of models, and ensuring the CNN is highly accurate for a larger classification space. This will make the system more flexible and more suitable for applications in gesture control systems, sign language translation, and interactive environments.

Another possible improvement is the generation of real-time response output for each identified gesture. For example, detection of a specific gesture can turn on an LED, sound a buzzer, display a message on an LCD, or issue a command to control an external device. This would transform the system into an active controller from a passive recognizer with the potential to directly communicate with other modules in a larger embedded setup. This integration would also facilitate the development of custom gesture-based interfaces for IoT devices, smart homes, and assistive technologies. In conclusion, this project lays a strong foundation for real-time embedded gesture recognition, and future developments can significantly enhance its utility, accuracy, and marketability. Continued experimentation, data refinement, and user feedback will play an essential role in shaping the future versions of the system so that it continues to be pertinent in research as well as in real-world applications.

REFERENCES

- [1] Chen, W., Yu, C., Tu, C., Lyu, Z., Tang, J., Ou, S., Fu, Y., & Xue, Z. (2020). A Survey on Hand Pose Estimation with Wearable Sensors and Computer-Vision-Based Methods. *Sensors*, 20(4), 1074..
- [2] F. Hundhausen et al., "Grasping with Humanoid Hands Based on In-Hand Vision and Hardware-Accelerated CNNs," *Robotics and Automation Letters*, 2022.
- [3] Y. Kim, J. So, C. Hwang, and W. Cheng, "An FPGA-Based Energy-Efficient Real-Time Hand Pose Estimation System with an Integrated Image Signal Processor for Indirect 3D ToF Sensors," *Journal of Embedded Computer Systems*, 2022.
- [4] P. Bartoli, "Time-of-Flight Hand-Posture Recognition Using Compact Histogram," *IEEE Transactions on Embedded Systems*, 2023.
- [5] H. Abdalla, "Hand Gesture Recognition Based on ToF Sensors," *Journal of Embedded Systems*, 2022.
- [6] P. P. Ray, "TinyML: Real-time Decision-Making on STM32," *Journal of Real-Time Systems*, 2023.

- [7] P. Dini, "AI Models in Embedded IIoT on STM32," *International Journal of Industrial Informatics*, 2023.
- [8] B. Deng and Z. Bo, "Research on STM32 Development Board Based on ARM Cortex-M3," *Microcontroller Applications*, 2022.
- [9] W. Tan and H. Yang, "Research on Intelligent Gesture Recognition System Based on STM32," *Sensors and Applications*, 2023.
- [10] P. Bartoli, D. Saporito, and A. Scandelli, "Time-of-Flight Hand-Posture Recognition Using Compact Normalized Histogram," **IEEE Sensors Journal**, 2024. DOI: [10.1109/JSEN.2024.10636485](https://ieeexplore.ieee.org/abstract/document/10636485/).
- [11] A. Krishna, V. Ramanathan, and S. S. Yadav, "A Sparsity-driven tinyML Accelerator for Decoding Hand Kinematics in Brain-Computer Interfaces," **IEEE Transactions on Circuits and Systems II: Express Briefs**, 2023. DOI: [10.1109/TCSII.2023.10389094](https://ieeexplore.ieee.org/abstract/document/10389094/).
- [12] A. Elboushaki, R. Hannane, K. Afdel, and L. Koutti, "MultiD-CNN: A multi-dimensional feature learning approach based on deep convolutional networks for gesture recognition in RGB-D image sequences," **Expert Systems with Applications**, vol. 142, 2020. DOI: [10.1016/j.eswa.2020.112994](https://www.sciencedirect.com/science/article/pii/S0957417419305317).

- [13] Z. Wang, S. H. Yoo, S. K. Oh, E. H. Kim, and Z. Fu, “A Study on Hand Gesture Recognition Algorithm Realized with the Aid of Efficient Feature Extraction Method and Convolution Neural Networks: Design and its Application to VR Navigation,” **Soft Computing**, 2023. DOI: [10.1007/s00500-023-09077-w](https://link.springer.com/article/10.1007/s00500-023-09077-w).
- [14] A. S. Mohamed, N. F. Hassan, and A. S. Jamil, “Real-Time Hand Gesture Recognition: A Comprehensive Review of Techniques, Applications, and Challenges,” **Cybernetics and Information Technologies**, 2024. DOI: [10.2478/cait-2024-0031](https://sciendo.com/pdf/10.2478/cait-2024-0031).
- [15] D. Aggarwal and P. Verma, “A Support Vector Machine Based Hand Gesture Recognition Using Accelerometer Sensor,” **International Journal of Sensor Networks and Data Communications**, vol. 6, no. 2, pp. 85–92, 2021.
- [16] G. Zholshiyeva, A. Daurbekova, and K. Rakhmatullaev, “Real-Time Kazakh Sign Language Recognition Using MediaPipe and Machine Learning,” **International Journal of Advanced Computer Science and Applications**, vol. 12, no. 4, pp. 225–231, 2021.
- [17] K. Patel, J. Wang, and L. Zhang, “Efficient 2D Convolutional Neural Networks for Embedded Gesture Recognition Using RGB Cameras,” **Journal of Embedded Systems Research**, vol. 15, no. 1, pp. 50–63, 2022.
- [18] R. Salter, M. K. Singh, and J. Luo, “Robust Hand Gesture Recognition for Robotic Applications Using EMG and RGB-D Data Fusion,” **Robotics and Autonomous Systems**, vol. 147, 2023.

- [19] H. Ashraf, M. A. Khan, and F. Hussain, "Optimizing the Performance of Convolutional Neural Network for Enhanced Gesture Recognition Using sEMG," *International Journal of Human-Computer Interaction and Signal Processing*, vol. 18, no. 2, pp. 112–121, 2024.
- [20] G. Zholshiyeva, A. Daurbekova, and K. Rakhmatullaev, "Real-Time Kazakh Sign Language Recognition Using MediaPipe and Machine Learning," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 4, pp. 271–276, 2021. Available: https://thesai.org/Downloads/Volume12No4/Paper_34-Real_Time_Kazakh_Sign_Language_Recognition.pdf
- [21] STMicroelectronics, "STM32 AI ModelZoo – Hand Posture Classification," GitHub, 2024. [Online]. Available: https://github.com/STMicroelectronics/stm32ai-modelzoo/tree/main/hand_posture

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Published works as well as papers presented at conferences, seminars, symposiums etc pertaining to the research topic of the research report/ dissertation/ thesis are suggested be included in this section. The first page of the article may also be appended as reference.

