FINAL YEAR PROJECT REPORT

FYP02-DS-T2510-0066

Malaysian Sign Language (BIM) Recognition Using Machine Learning

1211101888
SHAHNAZ BINTI HUSAIN SUKRI

BCS. DATA SCIENCE

JUNE 2025

FYP02-DS-T2510-0066
Malaysian Sign Language (BIM) Recognition Using Machine Learning

BY

SHAHNAZ BINTI HUSAIN SUKRI 1211101888

FINAL YEAR PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENT FOR THE DEGREE OF

BCS. DATA SCIENCE

in the

Faculty of Computing and Informatics

MULTIMEDIA UNIVERSITY
MALAYSIA

JUNE 2025

Copyright

Declaration

I hereby declare that the work has been done by myself and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institution of learning.

Name of candidate: Shahnaz Binti Husain Sukri

Faculty of Computing & Informatics

Multimedia University

Date: 29: 06: 2025

Acknowledgements

I would like to thank my supervisor, Dr. Alandoli Mohammed Nasser Mohammed, for their guidance, support, and helpful advice throughout this project. Their feedback has been very valuable in improving my work. I am also grateful to the Faculty of Computing & Informatics, Multimedia University, for providing the resources needed to complete this project. A special thanks to the FYP Committee Members for their help in organizing and managing the project process.

A big thank you to my family and friends for their constant encouragement and support. Their motivation helped me stay focused and overcome challenges. Lastly, I appreciate the deaf and hard-of-hearing community in Malaysia for inspiring this project. I hope this system can help improve communication and make a positive impact.

Abstract

This project presents the development of a real-time recognition system for Malaysian Sign Language (BIM) gestures, aimed at bridging communication barriers for the deaf and hard-of-hearing community. The system leverages MediaPipe Holistic for comprehensive keypoint extraction from video frames and employs multiple deep learning models—LSTM, BiLSTM, and GRU—to classify dynamic sign language gestures. A custom dataset comprising 15 BIM gestures was collected and preprocessed, with each gesture represented as a sequence of keypoint features. The models were systematically trained, tuned, and evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrices. An ensemble approach was adopted to further enhance recognition performance, with the best-performing model integrated into a user-friendly real-time application utilizing OpenCV for video capture and Tkinter for the graphical interface. User feedback indicated high satisfaction with the system's accuracy, usability, and responsiveness. The results demonstrate that the proposed system is effective, robust, and accessible, offering significant potential for supporting BIM learning and communication in real-world scenarios.

Table of Contents

List of Tables

List of Figures

List of Abbreviations/Symbols

Abbreviations		Symbols	
BIM	Bahasa Isyarat Malaysia	x	input feature vector
ASL	American Sign Language	y_i	class labels (+1 or -1)
BSL	B*ritish Sign Language*	α_i	the Lagrange multipliers
Libras	Brazilian Sign Language	$K(x_i, x)$	kernel function that maps input data into a higher-dimensional space
LSTM	Long Short-Term Memory	b	the bias term
CNN	Convolutional Neural Network	f_t	forget gate
IMU	Inertial measurement units		
HMM	Hidden Markov Model	i_t	input gate
SVM	Support Vector Machine	o_t	output gate
RNN	Recurrent Neural Network	C_t	memory cell
TTS	Text-to-Speech	h_t	hidden state
STT	Speech-to-Text	TP	true positives
UI	User Interface	TN	true Negatives
OpenCV	Open Source Computer Vision Library	FP	false Positives
		FN	false Negatives

List of Appendices

Chapter 1: Introduction

1.1 Project Overview

The Malaysian Sign Language (MSL) or Bahasa Isyarat Malaysia (BIM) is recognized as the official language of the Deaf in Malaysia through the Persons with Disabilities Act 2008 or Act 685 (*LAWS of MALAYSIA*, 2014). The language uses a combination of hand gestures, facial expressions and body movements to convey information. Despite its official status, a significant communication barrier persists between BIM users and non-signers, often limiting their communication to the majority.

Advancements in machine learning and computer vision offer promising solutions to bridge this communication gap between BIM users and non-signers. This project proposes the development of a real-time BIM gesture recognition system that detects sign gestures and translates them into their corresponding meanings. This system will be vision based and capable of recognizing different types of gestures and identifying their associated actions. In real world applications, the recognized gestures will be translated and displayed in real time, enabling smoother and more inclusive communication between the deaf and hearing communities.

1.2 Problem Statement

In Malaysia, there is a communication barrier between the deaf or hard of hearing and hearing members of the society (Md, 2022). While numerous sign language recognition systems have been developed, the majority are based on American Sign Language (ASL), which differs linguistically and culturally from BIM (Han Lin & Murli, 2022). As a result, these systems are not directly applicable to the Malaysian context. Additionally, existing machine learning based solutions often focus on static hand gestures and overlook the dynamic gestures and facial expressions that are fundamental components of BIM (Vaishnavi Karanjkar et al.,

2023). This limitation reduces their effectiveness in recognizing complete and meaningful sign language expressions.

Previous research and related applications, including gesture recognition systems have shown promising results in controlled environments but struggle with real time implementation due to accuracy issues and limited gesture vocabularies. These challenges highlight the need for a more robust, real time recognition system tailored specifically for BIM. Therefore, this project aims to address the gap by developing a machine learning based sign language recognition system that supports dynamic hand gestures and real-time performance, thereby facilitating more inclusive communication for the deaf or hard of hearing community in Malaysia.

1.3 Project Objectives

The goal of this project is to develop a real time BIM recognition system. This will assist the deaf community by using computer vision and deep learning techniques. The main objectives are:

1. To study and analyze existing machine learning techniques in the context of dynamic hand gestures, focusing on ASL and BIM.
2. To develop a real time BIM recognition system.
3. To evaluate the system’s performance.

1.4 Project Scope

The scope of this project is defined to ensure a focused and achievable implementation of a real time BIM recognition system. A working prototype will be developed that will convert the BIM gestures into text translation. The project scope is limited to recognizing 15 dynamic gestures. The performance of the system will be validated using relevant metrics. Additionally, the project will

implement an ensemble learning approach, where multiple recurrent neural network models are trained and evaluated. However, only the best model will be deployed and integrated into the real-time system to ensure optimal accuracy and efficiency during live gesture recognition.

1.5 Project Limitations

The project is limited to recognizing a certain set of BIM gestures. Thus, it cannot recognize a full range of expressions from BIM. As data is collected in controlled environments, it may not capture the variability of real-world conditions, such as differences in lighting and background. The performance of the system largely relies on the accuracy of the training dataset, which may not fully represent all potential users or scenarios. Due to resource and real time performance constraints, only the best-performing model from the ensemble approach will be selected and implemented in the final prototype. As the output will be a functional prototype rather than a full application, the system may offer limited functionality during the final presentation.

1.6 Methodology

The project methodology is designed to accomplish the aims of this project. The methodology followed consists of literature study, data collection, preprocessing, model development, real-time testing, and evaluation of the system. The approach looks towards developing a fully working prototype of BIM recognition.

Different types of machine learning models, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) are studied for static and dynamic gesture recognition. The data collection phase is conducted under varied light and background conditions to mimic real-world scenarios. For each of the 15 dynamic gesture classes, 150 video samples are recorded. MediaPipe Holistic is used to extract keypoints from each video frame, capturing hand joints,

face landmarks, and pose landmarks. These keypoints are formatted as numpy arrays to serve as input features for model training.

Multiple neural network architectures, including LSTM, BiLSTM, and GRU are developed to classify dynamic BIM gestures. The model is designed to handle dynamic gestures which are portrayed by a sequence of inputs. An ensemble learning approach is employed, where the outputs of these models are combined and evaluated.

OpenCV is integrated for real-time video capture and processing. The best-performing model from the ensemble is deployed in the real-time system to ensure optimal accuracy and efficiency. The system translates recognized BIM gestures into text, which is displayed on the user interface.

At the end, the model evaluation asses the performance of the system using key performance metrics like accuracy, precision, recall, F1-score and confusion matrix that shows the misclassifications. The robustness testing assesses the model's performance across variations in user, environment, and camera setup. These assessments are conducted to ensure that the system is accurate, reliable, and effective for real-time BIM gesture recognition.

1.7 Target Audience

This project is targeted at deaf and hard-of-hearing people in Malaysia. They could benefit from this project, as communication and accessibility will improve. This system will also help non-signers to communicate with BIM users. Accommodating both groups allows this project to promote inclusivity while contributing to a more connected and understanding society.

1.8 Summary

This project focuses on developing a real time BIM recognition system using machine learning to bridge the communication gap between the deaf and hard of hearing community and non-signers. The real time video processing and capturing of the system is done through OpenCV and the gestures are classified through an ensemble learning approach of multiple recurrent neural networks. The project is only limited to the recognition of 15 specific dynamic BIM gestures. The aim of this project is to provide a working prototype that is accurate, which is measured through performance metrics and usability tests. In the next chapter, a comprehensive literature review will explore existing sign language recognition systems, examine the different machine learning approaches, and highlight the gaps that this project aims to address.

Chapter 2: Literature Review

2.1 Introduction

Spoken languages mainly use the "vocal-auditory" channel, produced through the mouth and interpreted by the ear. In contrast, sign languages utilizes the “corporal-visual” channel, being produced with the body and perceived with the eyes (Adaloglou et al., 2021). Because sign languages evolve naturally wherever deaf communities can congregate and communicate, they are widely used around the world but are not universal (Adaloglou et al., 2021). ASL, British Sign Language (BSL), and BIM are distinct languages with their own vocabularies and grammatical structures.

Sign language recognition is an interdisciplinary research area at the intersection of pattern matching, computer vision, machine learning, linguistics, natural language processing and human-computer interaction (Wadhawan & Kumar, 2019). Its core aim is to develop automated systems that can interpret sign languages (Wadhawan & Kumar, 2019), capturing the visual and gestural information (from the hands, face, and body) and converting it into a machine-interpretable form. Ultimately, sign language recognition technologies strive to generate meaningful linguistic output, such as text or speech, from signed input.

2.1.1 Manual and Non-Manual Features

Sign languages consist of manual and non-manual elements, each contributing essential linguistic information (Adaloglou et al., 2021). Manual features include hand shape, position, movement, and orientation of the palm or fingers (Adaloglou et al., 2021). These features can be expressed through one-handed signs, in which a single dominant hand is used (either in a static gesture or with motion), or through two-handed signs, where both the dominant and non-dominant hands are involved. Two-handed signs can be further grouped into type 0 as shown in Figure 2.2, whereby both hands are equally active, and type 1 as

shown in Figure 2.3, where the dominant hand is more active than the non-dominant hand (Wadhawan & Kumar, 2019). A full hierarchical classification of different sign types is shown in Figure 2.1. In addition to these manual components, non-manual features, such as eye gaze, head nods or shakes, shoulder orientation, and facial expressions (including mouthing or mouth gestures), also play a critical role (Adaloglou et al., 2021). They can modify or emphasize the meaning of a sign, similar to intonation in spoken languages, and are thus essential for conveying the full grammatical and semantic range of sign languages. Understanding and accurately capturing both manual and non-manual features is key to developing robust sign language recognition systems, as they collectively form the linguistic symbols that enable clear, nuanced communication.

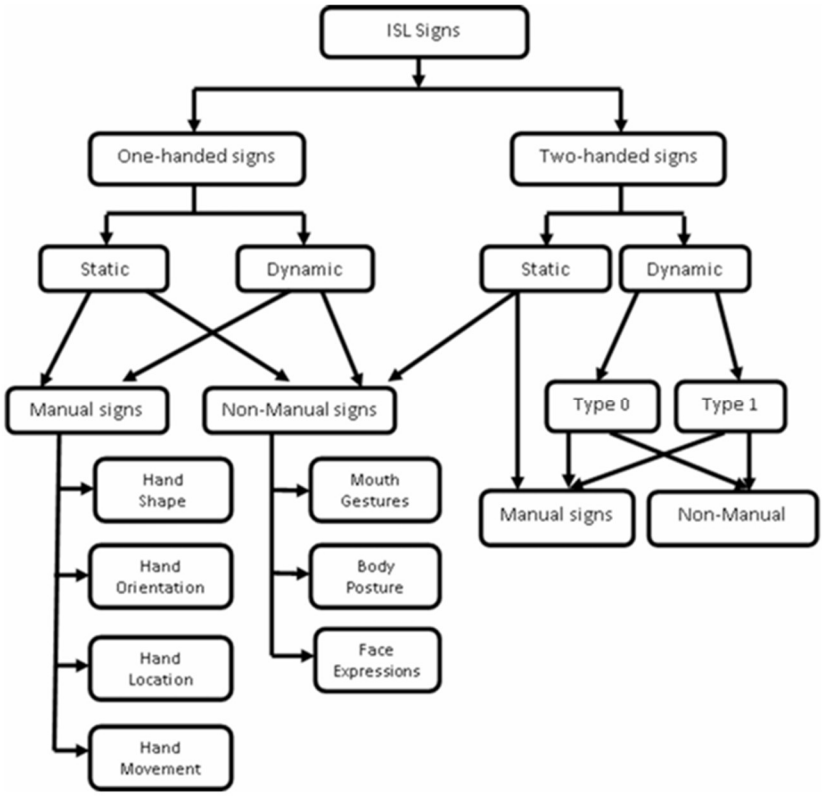


Figure 2.1: Hierarchical Classification of Indian Sign Language (ISL) Signs (Wadhawan & Kumar, 2019)

Figure 2.1: Two handed type 0 sign (both the hands are active) (Wadhawan & Kumar, 2019)

Figure 2.2: Two handed type 1 sign (only dominant hand is active) (Wadhawan & Kumar, 2019)

2.1.2 Static vs Dynamic Gesture Recognition

Sign language gestures are typically divided into two categories, static and dynamic, each requiring different computational approaches for recognition. Static gestures are single, stationary hand postures captured in a single frame, representing fixed meanings like letters or numbers. Recognizing static gestures is quite simple and based on image processing techniques. Furthermore, some use machine learning models like Convolutional Neural Networks (CNN). They are suitable for this application as they extract spatial features like contours and edges. This is efficiently done in static images. Nonetheless, a challenge in recognition of static gestures is due to segmentation issues and resemblance in

the general shape of the gesture which can trigger misclassification. Dynamic gestures, unlike static gestures, involve the movement of the user, making them much more complicated. For instance, dynamic gestures comprise the transfer, movement, or contact of the hand with the body or face. Because of the differences in gesture execution speed and orientation and signer styles, these gestures are more difficult to do (Samaan et al., 2022; Rastgoo et al., 2021). The temporal modeling techniques such as Recurrent Neural Networks (RNN), LSTM, or Bi-directional LSTMs are required to recognize dynamic gestures. Capturing the dependencies between sequences of data is done well by these models, which makes it easier to handle the complex spatial and temporal structure of dynamic gestures (Samaan et al., 2022).

2.1.3 Sensor and Vision Based Recognition

Sign language recognition has seen significant advancements in recent years, primarily through two main approaches: sensor-based and vision-based systems. Figure 2.4 shows below the two gesture recognition approaches.

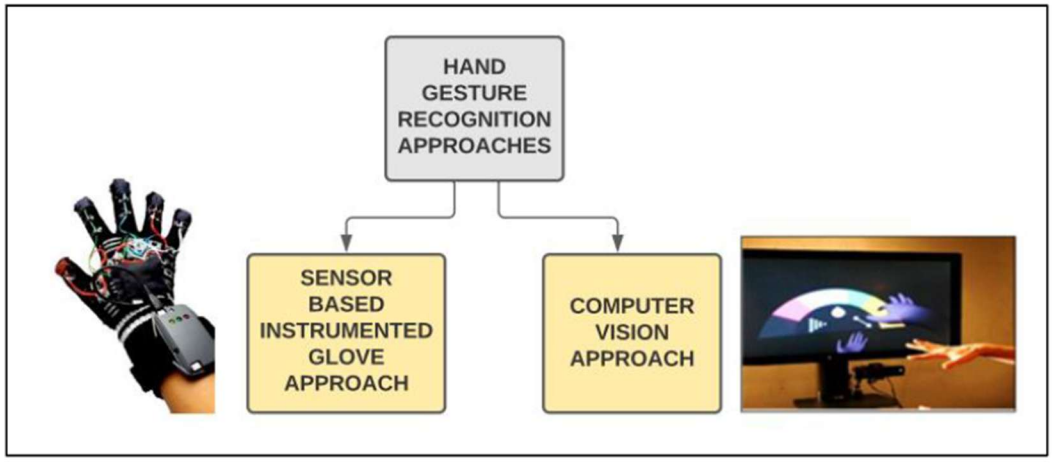


Figure 2.4: Hand Gesture Recognition Approaches (Jain et al., 2022)

Sensor-based systems utilize wearable devices such as data gloves, accelerometers, and inertial measurement units (IMUs) to capture the motion and orientation of the hands and fingers. These systems are known for their high accuracy and robustness to environmental changes, as they are less affected by lighting or background variations. For example, (Papastratis et al., 2021) discuss the use of sensor-based gloves and IMUs, highlighting their effectiveness in capturing dynamic gestures and their reliability in real-world applications. However, the need for users to wear specialized hardware can limit the practicality and user acceptance of these systems, especially for daily or widespread use (Galván-Ruiz et al., 2020).

Vision-based systems uses cameras and computer vision algorithms to interpret gestures from video streams. The introduction of deep learning and advanced keypoint detection frameworks, such as MediaPipe and OpenPose, has greatly improved the performance of vision-based systems (Lugaresi et al., 2019). These systems are non-intrusive and more user-friendly, as they do not require any wearable devices. (Madhiarasan & Roy, 2022) provide a comprehensive review of vision-based sign language recognition, noting the rapid progress enabled by large-scale datasets and deep neural networks. Despite these advances, vision-based systems can be sensitive to changes in lighting, background, and camera positioning, which may affect recognition accuracy.

Recent research trends indicate a growing preference for vision-based approaches due to their non-intrusive nature and the rapid advancements in computer vision. However, sensor-based systems still offer superior robustness in challenging environments. Hybrid approaches that combine both sensor and vision data are also being explored to leverage the strengths of both methods (Madhiarasan & Roy, 2022).

2.2 Machine Learning Techniques in Gesture Recognition

Machine learning is the heart of modern sign language recognition systems that allows the automatic classification of complex gestures in diverse conditions. In the initial stages, algorithms such as Support Vector Machines (SVM) and Hidden Markov Models (HMM), were often used, which worked well for small datasets of specific signs. However, due to the increase in complexity and size of datasets over the years, the use of deep learning techniques has increased with CNN used for spatial feature extraction and LSTM used for temporal modeling. These different techniques must balance accuracy, speed, and reliability to create solutions that are easy to use and work well for different signers.

2.2.1 Support Vector Machine (SVM)

SVM is a supervised learning algorithm commonly used for sign language recognition, particularly in static gesture classification. SVM works by finding the optimal hyperplane that separates different classes of gestures with the largest margin, making it effective for high-dimensional feature spaces. The decision function of an SVM if defined as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \quad (1)$$

where:

- x is the input feature vector
- y_i are the class labels (+1 or -1)
- α_i are the Lagrange multipliers
- $K(x_i, x)$ is the kernel function that maps input data into a higher-dimensional space

- b is the bias term.

SVM has shown high accuracy for static gesture tasks due to its robustness and capability to handle small datasets effectively. However, SVM struggles with dynamic gestures because it cannot model temporal dependencies, which are crucial for sequential data processing.

2.2.2 Hidden Markov Model (HMM)

HMM are widely used for dynamic gesture recognition in sign language systems. They model temporal dependencies by capturing sequential data patterns, making them ideal for recognizing gestures that evolve over time. HMMs classify gestures by analyzing transitions between states (e.g., hand movements and orientations) based on observed data (Al-Qurishi et al., 2021). Although HMMs are effective for modeling dynamic data, their reliance on assumptions of conditional independence between observations and states can oversimplify complex relationships in gestures. This limitation has led to their gradual replacement by advanced deep learning methods, such as LSTM networks, which offer greater flexibility in temporal modeling (Al-Qurishi et al., 2021).

2.2.3 Convolutional Neural Networks (CNN)

CNNs are widely used in sign language recognition for their ability to extract spatial features from images and video frames. CNNs are particularly effective for static gesture recognition, where each sign is represented as a single image (Srivastava et al., 2022). CNNs utilize layers such as convolution, pooling, and fully connected layers to automatically extract and classify features, reducing the reliance on manual feature engineering (Bhatia & Ankita Wadhawan, 2021). The CNN architecture typically involves an input layer, one or more convolutional layers for feature extraction, max pooling layers for dimensionality reduction, and

dense layers for classification. This process is shown in Figure 2.5, where an input image is passed through the different layers of the network to produce an output prediction. However, CNNs alone are less effective for dynamic gesture recognition, which requires temporal modeling across multiple frames.

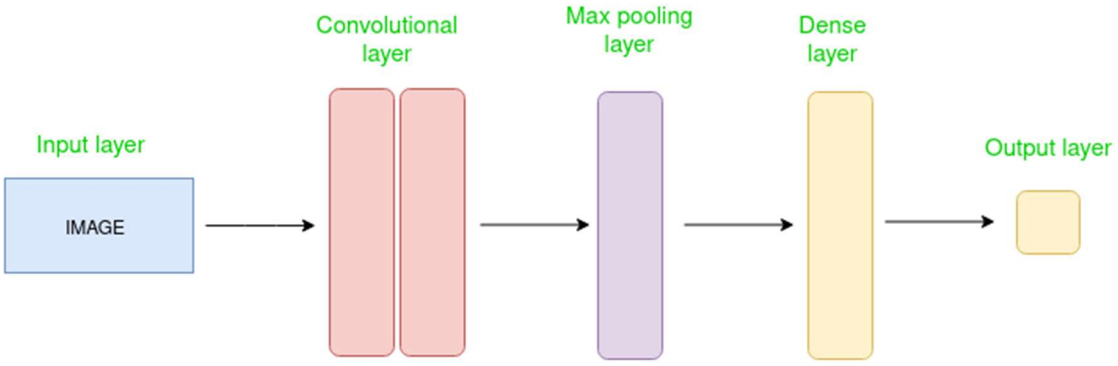


Figure 2.5: Simple CNN Architecture (GeeksforGeeks, 2017)

2.2.4 Long-Short Term Memory (LSTM)

LSTM networks, a type of RNN, are ideal for dynamic gesture recognition due to their ability to process sequential data. They address the challenges of vanishing gradients, allowing the model to retain information over long sequences. LSTM models have been successfully applied to dynamic sign language recognition, achieving high accuracy by analyzing temporal dependencies in gesture trajectories. For example, LSTM-based systems have been shown to effectively classify dynamic gestures by combining features extracted from sequences of video frames (Al-Qurishi et al., 2021). Bi-directional LSTMs (Al-Qurishi et al., 2021) further enhance performance by considering both past and future context in sequences, making them suitable for complex, continuous sign language recognition tasks.

2.2.5 MediaPipe

MediaPipe is a framework developed for real-time multi-modal feature extraction, including hand, body, and face landmarks. The MediaPipe hand landmark detector can track 21 hand landmarks, as shown in Figure 2.6 and 33 pose landmarks, as shown in Figure 2.7. It simplifies the preprocessing pipeline by providing accurate keypoint detection, enabling dynamic gesture recognition systems to track motion trajectories and spatial configurations efficiently. When combined with RNN models like LSTMs, MediaPipe has been used to achieve high accuracy in dynamic sign language recognition. By extracting features such as hand position, orientation, and facial expressions, MediaPipe reduces the complexity of dataset preprocessing and enhances the robustness of recognition systems (Samaan et al., 2022).

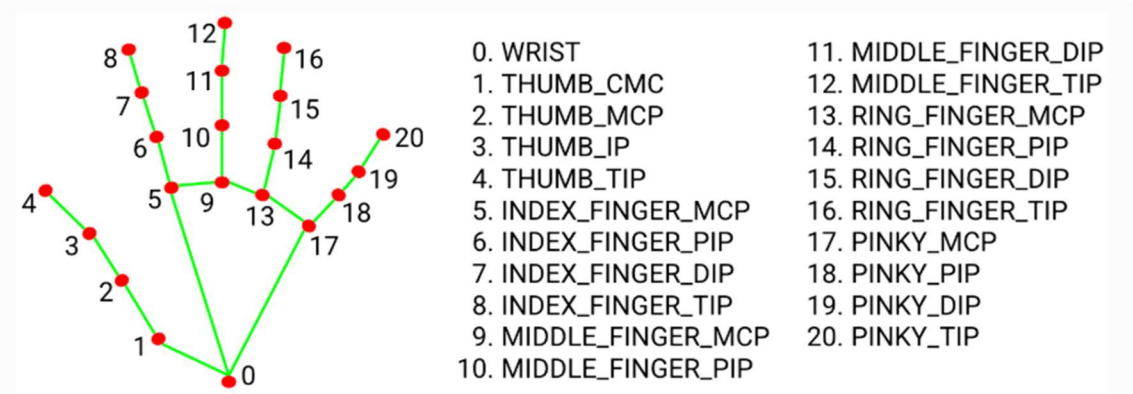


Figure 2.6: MediaPipe Hand Landmarks (MediaPipe, n.d)

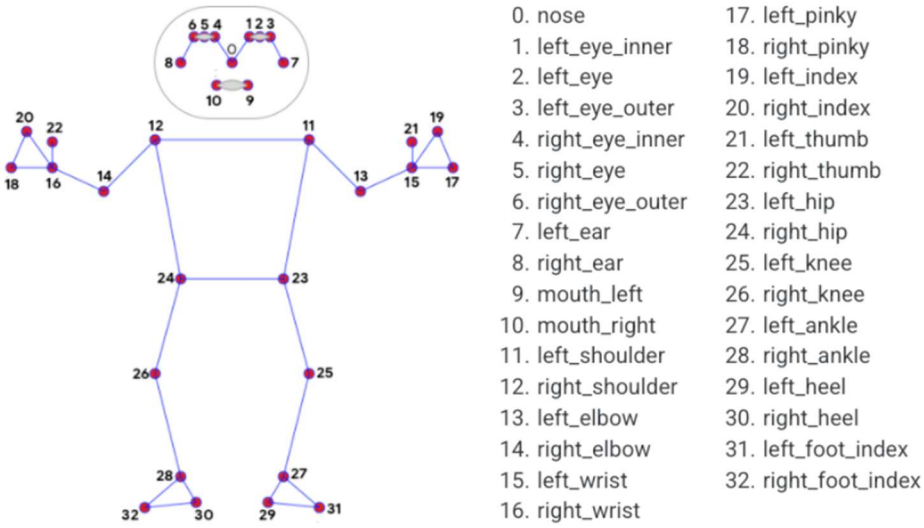


Figure 2.7: MediaPipe Pose Landmarks (MediaPipe, n.d)

2.3 Malaysian Sign Language Recognition

BIM is the official sign language of Malaysia, recognized under the Persons with Disabilities Act in 2008 (LAWS of MALAYSIA, 2014). It combines manual signals, such as hand and finger gestures, with non-manual signals, including facial expressions and body postures, to convey meaning. Unlike universal sign languages, BIM is linguistically and culturally distinct, incorporating elements specific to the Malaysian context (Han Lin & Murli, 2022).

In BIM, both static and dynamic gestures are essential for expressing a wide range of vocabulary and grammatical structures. Static signs involve hand configurations held in a fixed position and are typically used for simple concepts such as alphabets shown in Figure 2.8 (a) and numbers Figure 2.8 (b). However, dynamic signs, which involve continuous hand motion, changes in orientation, and sometimes non-manual elements like facial expressions or head movements, form the majority of BIM vocabulary. Examples of dynamic signs are seen in Figure 2.9 (a) and Figure 2.9 (b)

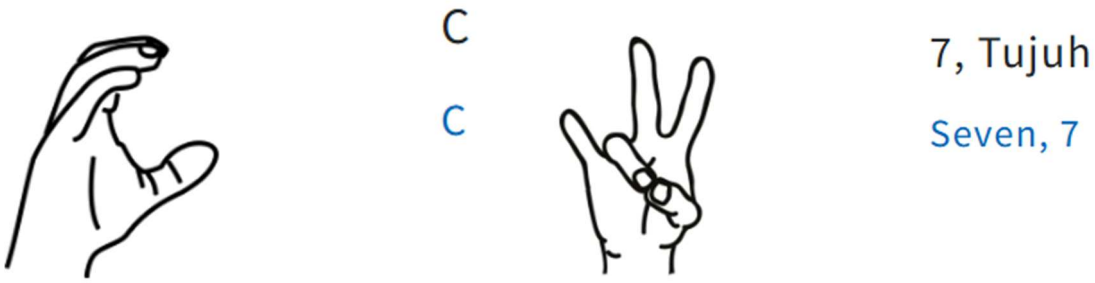


Figure 2.8 (a): Static Sign for "C". (b): Static Sign for "7, Seven"

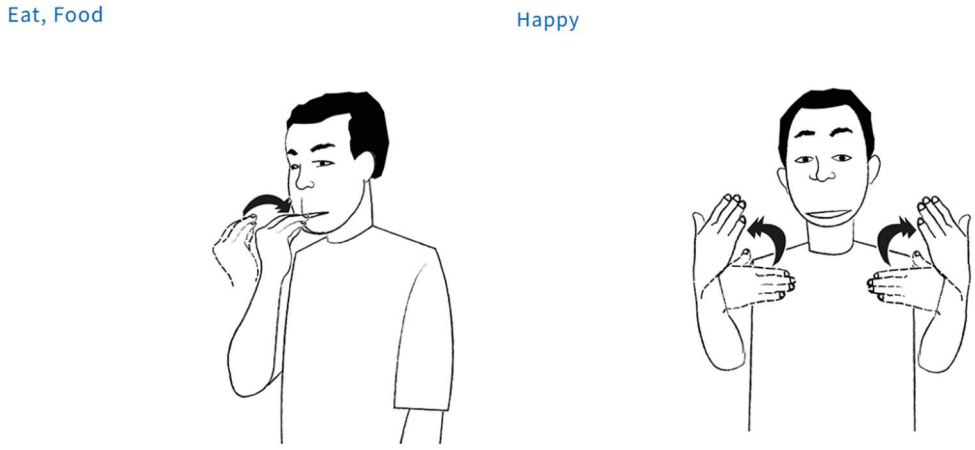


Figure 2.9 (a): Dynamic Sign for "Eat, Food". (b): Dynamic Sign for "Happy"

Based on the BIM Sign Bank, there are approximately 2,000 dynamic signs compared to only 99 static signs, highlighting the predominance of motion-based gestures in real communication. These dynamic signs are especially important for conveying actions, verbs, and more complex linguistic structures. Furthermore, many of these signs require two-handed coordination, increasing the challenge of accurate recognition. As such, a recognition system focused solely on static signs would be insufficient to capture the full expressive capacity of BIM. Therefore, to build an inclusive and effective recognition model, it is crucial to consider both static and dynamic gestures, with particular emphasis on the temporal modeling required for dynamic sign interpretation.

To address the communication gap between the deaf community and non-signers, various systems have been developed for BIM recognition. Vision-based methods using cameras have been employed to capture gestures, providing a cost-effective alternative to hardware like data gloves or Kinect cameras. Early approaches faced challenges such as poor image segmentation and centroid estimation, but advancements with frameworks like MediaPipe have significantly improved performance (Md, 2022).

Table 2.1: Summarized Review of BIM Recognition Systems

Author	Acquisition Mode	Single/ Double Handed	Static/ Dynamic	Techniques Used	Multilingual Support	Recognition Rate
Imran Md Jelas	Webcam	Both	Both	MediaPipe + LSTM	No	Not specified
Han Lin & Murli	HD Webcam	Single	Static	TensorFlow	Yes	Not specified
Khan et al.	Webcam	Single	Static	CBAM-Resent	No	More than 90%

Table 2.1 presents a comparative overview of several recent studies on BIM recognition systems, focusing on acquisition methods, gesture characteristics, techniques used, multilingual capabilities, and recognition performance. (Md, 2022) employed both static and dynamic gestures using single and double-handed signs captured via a webcam, integrating MediaPipe with LSTM for recognition. However, the recognition rate was not reported, and the system lacked multilingual support. (Han Lin & Murli, 2022) used a high-definition webcam for capturing static, single-handed gestures and implemented a TensorFlow-based approach, with some level of multilingual support, though the recognition accuracy was also not specified.

In contrast, (Khan et al., 2021) adopted CBAM-ResNet for static, single-handed gesture recognition using webcam input, achieving over 90% accuracy, though without multilingual support. These studies demonstrate varying approaches and highlight gaps such as limited gesture types, lack of real-time evaluation, and insufficient coverage of dynamic, two-handed, and multilingual recognition, underscoring the need for more inclusive and robust solutions, particularly for Malaysian Sign Language (BIM).

The development of BIM recognition systems has been shaped by key research contributions over the years. Figure 2.10 (Khan et al., 2021) illustrates a timeline of these advancements, highlighting significant milestones in gesture recognition techniques, dataset creation, and the adoption of machine learning frameworks.

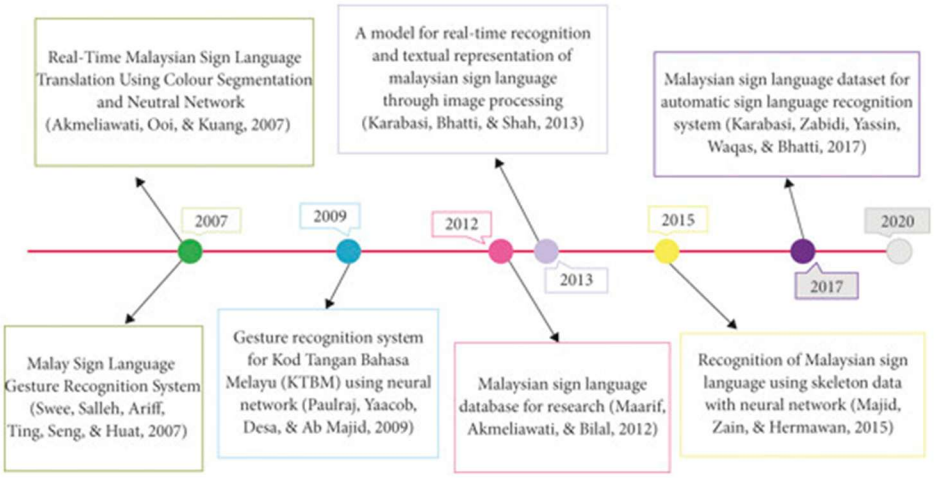


Figure 2.10: Timeline of key advancements in BIM recognition systems (Khan et al., 2021)

Despite notable progress in BIM gesture recognition, the reviewed systems fall short in real time dynamic gesture interpretation. There remains a significant research gap in developing a scalable, real time BIM recognition system that can handle the full complexity of two-handed and dynamic gestures

using lightweight, low-cost solutions like MediaPipe and LSTM. This project addresses this gap by proposing a prototype capable of accurate gesture recognition tailored specifically for Malaysian Sign Language.

2.4 Key Challenges

A critical challenge is the unavailability of BIM-related datasets. Most datasets are available for widely researched sign languages like ASL. They do not capture sufficient linguistic and cultural characteristics of BIM. Not having enough data does not enable ML models to generalize well especially with respect to regional differences and person signer differences

Another problem is the variation in gesture realization among users. BIM gestures can be distinguished based on their speed, orientation, and style, creating difficulties for recognition trained on limited and less diverse datasets. These differences make the systems less effective in the real world (Samaan et al., 2022). In addition, BIM has a lot of contextual and meaningful information in non-manual components like facial and postural cues. But, implementing these features into the recognition systems is not easy due to the necessity of processing the inputs related to the hand, face, and pose simultaneously (Samaan et al., 2022; Han Lin & Murli, 2022).

2.5 Summary

This chapter examines the development and challenges of sign language recognition systems. BIM is the official sign language of Malaysia as recognized by the Persons with Disabilities Act 2008. It is a language that consists of manual signs and non-manual components (facial expressions, body postures). This section underlines the unique language and cultural properties of BIM for recognition systems.

The literature review evaluates earlier sign language recognition systems, distinguishing between recognition of static gestures versus dynamic gestures. Processing static gestures with methods such as CNNs and dynamic gestures with more complex temporal modeling methods such as LSTMs and HMMs are easier. The use of Frameworks such as MediaPipe have helped in improved recognition accuracy and simpler pre-processing pipelines.

Through summarizing the review of several BIM recognition systems, the literature reviewed identifies MediaPipe and LSTM as good candidates for addressing the issues related to real-time recognition of both static BIM and dynamic BIM gestures. Nonetheless, dataset limitations and differences in gesture style are significant challenges. The goal of this project is to bridge the gaps in this area by implementing a real-time BIM recognition system based on machine learning, with the primary objective of increasing accessibility and inclusivity for the Malaysian deaf community.

Chapter 3: Requirement Analysis

3.1 Overview

This chapter outlines the detailed system analysis and requirements gathering process for the real-time BIM recognition system. The purpose of this chapter is to have a clear clarity of functional and non-functional requirements of the system along with the challenges and constraints involved during implementation. It starts off by looking at current existing systems and applications that deals with the communication problems faced by the deaf and hard-of-hearing. The paper analyzes the shortcomings of these existing applications to emphasize the need for a novel application which can be developed tailor-made for BIM. Then, the technical, economic, and operational feasibility of the project is examined. This part describes how the requirements gathering took place which will form the basis for the definition of the functional and non-functional requirements. Lastly, a system requirements specification is given which specifies the hardware, software and tools that will be necessary for the development process. This part makes the whole base for carrying out the implementation of different chapters for the system

3.2 Existing System Analysis

The analysis of existing systems reveals several limitations, justifying the development of a new, tailored solution for BIM recognition. Over time, sign language recognition systems have transitioned from hardware dependent methods, like data gloves and motion sensors, to vision-based approaches focusing on computer vision and machine learning. For example, ASL recognition systems primarily focus on static gestures and uses models such as CNN, which excel at extracting features from images but cannot handle the sequential data in dynamic gestures, to classify hand shapes and orientations [15]. These systems are often limited to ASL and lack consideration for dynamic and non-manual

elements critical to other sign languages. Many current systems emphasize static hand postures, such as letters and numbers, which are easier to process but fail to capture the temporal patterns of dynamic gestures.

The application HandTalk replicates the text and speech making use of animated avatars in sign language. This technology however, does not support multiple languages and does not convert its sign back into text and voice, restricting two-way communication. Furthermore, the systems for sign language recognition that are TensorFlow based operate on static datasets and simplified gestures without considering dynamic and expressive non-manual components important for more advanced sign language recognition systems [11]. The lack of BIM specific datasets further aggravates this problem as the training and validating of machine learning models for BIM recognition is hampered.

3.3 Feasibility Analysis

Feasibility analysis ensures that the proposed system is viable in terms of technical implementation, cost, and operational integration. This section examines the technical, economic, and operational aspects of the BIM recognition system.

3.3.1 Technical Feasibility

The system is built on reliable state-of-the-art frameworks like MediaPipe to extract keypoints, multiple neural networks to identify gestures dynamically and OpenCV for real-time video processing. These tools are commonly present on the internet, showcasing their reliability and scalability. The system needs a high-definition webcam as the hardware requirement for accurate gesture. To ensure data availability, the dataset will be prepared on custom-made data that will be recorded in a controlled environment using pre-defined BIM gestures. With the

help of MediaPipe, getting ready input data is made easy as it also returns keypoints which can help to reduce the complexity of working with the image as pre-processing would require a lot of resources.

3.3.2 Economic Feasibility

The cost of developing the system is small since it uses an ordinary camera and consumer-grade computer hardware. Thus, the cost of hardware is low. All software costs are eliminated by relying on open source library MediaPipe, TensorFlow and OpenCV. Expenses incurred for data collection are also low since BIM gestures will be recorded in-house instead of purchasing or acquiring data collection from third parties at a high cost. The operation costs are low because the system uses readily available hardware and does not need dedicated servers or cloud setup.

3.3.3 Operational Feasibility

The beneficiary of the system are the members of the deaf and hard-of-hearing community in Malaysia and non-sign language users who communicate with them. The system's ease of usage for translating users' gestures to text is user-friendly interface to enable real-time usage. Users just performs gestures in front of a webcam, and the outputs are displayed instantly. Thus, it is easy to use. The system is optimized for controlled environments that have regulated background and lighting conditions. Also, it has a modular system that allows for easy maintenance and updates like adding more features or gestures. The system is built upon open-source libraries that benefit from long-term guarantees and community support for bug fixes and updates.

3.4 Fact-Finding Techniques

Observation was used as the fact-finding technique for the BIM recognition system. We chose this technique to get a first-hand understanding of the system requirements while addressing the language barrier and technical issues with BIM. BIM gestures, their execution and variations were thoroughly observed that generated useful insights for system development.

3.4.1 Observation

Observation assists in understanding the movements and conversations used in BIM. Through the use of videos and presentations of BIM gestures from the BIM Sign Bank shown in Figure 3.1 below, we study these gestures. The BIM Sign Bank helped in answering questions about the structure of gestures, their variants and meanings and allowed a sufficiently well-defined set of gestures to be selected for the system. The system design includes both manual and non-manual components, based on the study of recorded videos.

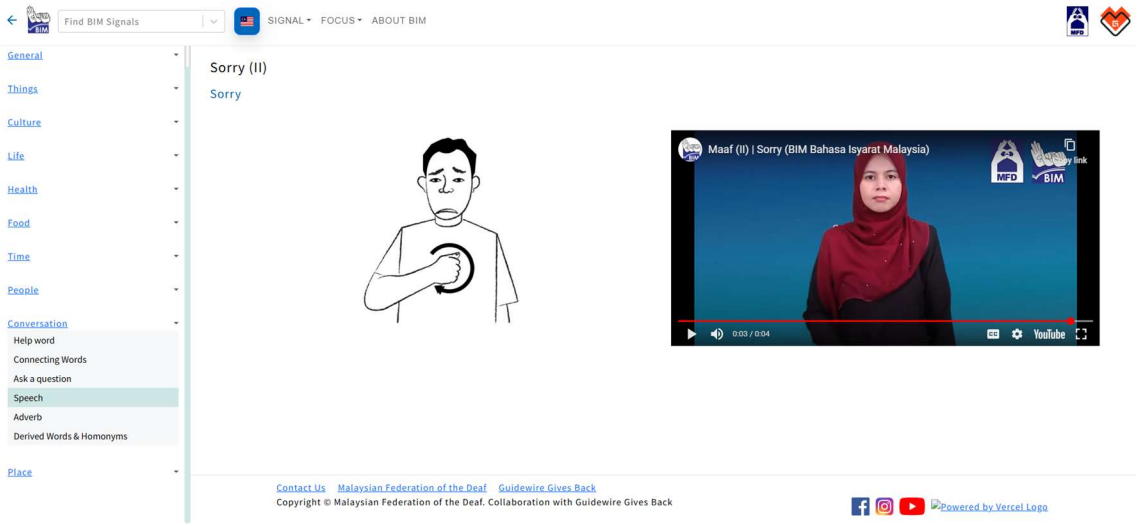


Figure 3.1: Illustration of 'Sorry' from BIM Sign Bank

3.5 Requirements

The BIM recognition system requirements are aimed at ensuring fairly accurate and efficient sign language recognition. These requirements split into functional and non-functional. Functional requirements are the essential features which need to be developed to implement a successful sign recognition system. Similarly, non-functional requirement ensures performance, usability, and scalability. The upcoming sub-sections elaborate these requirements.

3.5.1 Functional Requirements

The functional requirements of the BIM recognition system represent its essential functions for which it should perform. The system must recognize BIM gestures such as hand signs and translate them into text in real-time with low latency. The system will only recognize dynamic gestures. Also, the system must capture and understand other materials, which include facial expressions and head movements. Users should be able to do the gestures in front of the webcam and get the text output on the screen immediately. Table 3.1 below summarizes the functional requirements.

Table 3.1: Functional Requirements

Requirement	**Description**
Real-Time Gesture Recognition	Recognize and translate BIM gestures into text in real time with minimal latency.
Dynamic Gesture Processing	Process dynamic gestures (words, nouns, objects)
Non-Manual Component Interpretation	Capture and interpret non-manual elements such as facial expressions and head movements
User Interaction and Output	Allows users to perform gestures in front of a webcam and receive immediate text output

3.5.2 Non-Functional Requirements

The BIM recognition system's operational performance and usability depend directly on identifying non-functional needs that focus on quality attributes. The recognition accuracy of the system must be high enough to make reliable translations. The system must operate with minimal latency. The system should be easily scalable so that new gestures can be added without extensive re-development for future upgrades. The interface should be easy to use with not much training and setup requirement needed. The system must display regular performance, regardless of lighting condition, background or differences between users. Table 3.2 below depicts a summary of the non-functional requirements.

Table 3.2: Non-Functional Requirements

Requirement	**Description**
Recognition Accuracy	Ensure high accuracy for reliable gesture translations
Real-Time Performance	Operate with minimal computational delay
Scalability	Allow the addition of new gestures without major redevelopment
User-Friendliness	Provide an intuitive interface the requires minimal training
Robustness	Maintain consistent performance under different lighting, backgrounds, and user variations.

3.6 Summary

This chapter talks about what will be the requirements to develop a BIM recognition system. It points out the drawbacks of current systems for sign language recognition that focus on static gestures and do not support dynamic and non-manual components. The proposed system will target challenges using MediaPipe to obtain keypoints, LSTM to classify dynamic gestures and OpenCV to translate gesture into text. To ensure reliable and accurate functioning of the proposed system, confidence-based prediction, and modular architecture for easy updating are used.

The analysis of feasibility shows the viability of the system. It uses open-source frameworks along with commodity hardware to minimize cost and enhance robustness. High-quality training data for deep learning models can be recorded in controlled environments by using custom datasets. The requirements gathering involved observing the gestures of BIM, reviewing documents, and developing the dataset to provide the necessary inputs.

Finally, the functional requirements of the system will focus on recognizing gestures in real time, integrating gestures and non-manual components and user-friendly interaction while the non-functional requirrements will be of accuracy, performance, and scalability.

Chapter 4: System Design

4.1 Overview

In this chapter we present the system design. These are the specifications based on project requirements generated. We develop architectural, data, interface, and process designs. Through system design, appropriate solutions to the project requirements in the implementing phase are proposed. It connects a project's crucial elements to implement a system and enhances reliability, scalability, and performance by defining technical paths. The project gains a lot from this stage as it efficiently plans the architecture, data, workflows and UI.

4.2 Proposed System

The proposed system aims to overcome the shortcomings of currently available sign language recognition systems by leveraging an ensemble of advanced neural network models. The proposed system uses MediaPipe Holistic for real-time keypoint detection of hands, face, and body. These keypoints are extracted from video frames and serve as the input for the gesture recognition pipeline, capturing the temporal dynamics of each gesture sequence.

Data preprocessing involves structuring the extracted keypoints into consistent arrays, forming gesture video sequences that accurately represent the dynamic components of BIM gestures. The dataset is collected under varied lighting, backgrounds, and camera angles to ensure robustness and generalizability.

For gesture classification, the system employs multiple neural network architectures, including LSTM, BiLSTM, and GRU, which are well-suited for modeling sequential data and capturing long-range dependencies in gesture sequences. An ensemble learning approach is adopted, where the outputs of these models are combined and evaluated to identify the best-performing model

based on validation metrics. Only this best model is deployed in the real-time system to ensure optimal accuracy and efficiency during live gesture recognition.

The deep learning model is integrated with OpenCV to enable real-time video capture and processing. Users can perform BIM gestures in front of a webcam, and the system instantly translates recognized gestures into text displayed on the screen. The system is designed to avoid duplicate gesture outputs and only display recognized gestures that meet a confidence threshold, enhancing usability and user experience.

The performance is assessed by using key evaluation metrics such as the accuracy, precision, recall, F1-score, and confusion matrix which ensure the effectiveness in classification and real-time use in various scenarios. The software is tested in bad lighting, cluttering, and with other human beings also being present. The key evaluation metrics used are defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

where:

- $TP = True\ Positives$
- $TN = True\ Negatives$
- $FP = False\ Positives$
- $FN = False\ Negatives$

By integrating ensemble learning and selecting the best model for deployment, the proposed system not only addresses communication barriers for the deaf and hard-of-hearing community in Malaysia but also demonstrates the potential of combining MediaPipe with advanced neural network ensembles for state-of-the-art gesture recognition. The final prototype emphasizes inclusiveness and real-time translation of BIM gestures, bridging the communication gap in a practical and user-friendly manner.

4.3 System Architecture

The BIM recognition system has a modular architecture which consist of three modules, which are Learning Module, Training Module, and Detection Module. These modules are shown in Figure 4.1 below.

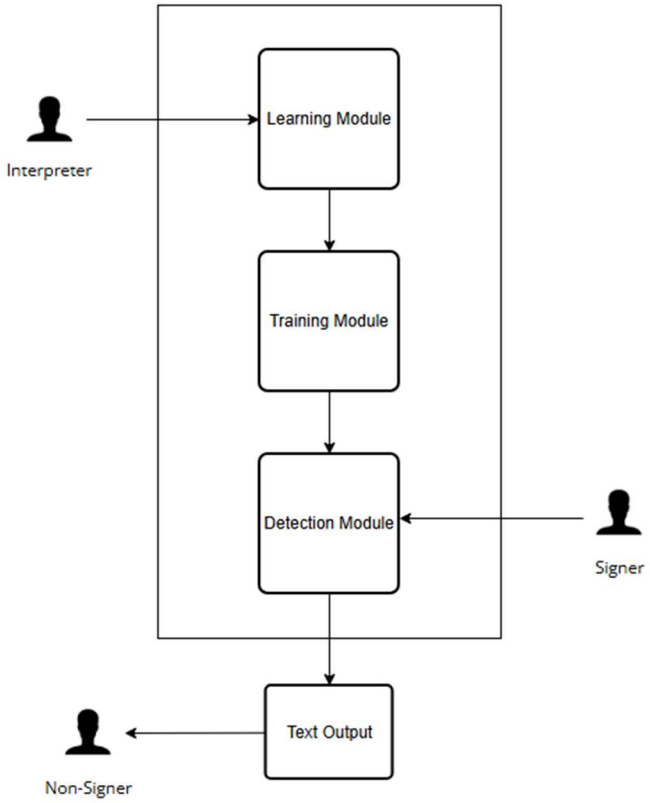


Figure 4.1: High-level System Architecture Diagram

4.3.1 Architectural Design

As seen from the 3 layer architecture diagram in Figure 4.2 below, the user layer provides a simple interface through which the user performs gesture in real time with the help of a camera. This layer captures live video inputs that the system processes and displays the recognized gesture as text for ease of use to the signer since its use is straightforward.

The application layer forms the core of the system and is responsible for all computational processes. It is divided into three key modules: the Learning Module, the Training Module, and the Detection Module. The Learning Module allows new BIM vocabulary to be added to the system by collecting keypoints using MediaPipe. These keypoints are stored in the action dataset for future use.

The Training Module processes the collected data, training gesture recognition models using multiple neural networks with TensorFlow/Keras. Once trained, the models are serialized and saved into the classifier database for future predictions. The Detection Module restores the trained model, processes real-time video input by extracting temporary keypoints through MediaPipe, and predicts gestures using the trained classifier. The predicted gestures are then displayed as text output for the user.

The storage layer supports the application layer by providing persistent data management. The system maintains an BIM action dataset, which stores the collected keypoints for added gestures, and an BIM action model classifier database, which stores serialized gesture recognition models. This allows the system to update its vocabulary and improve model performance over time.

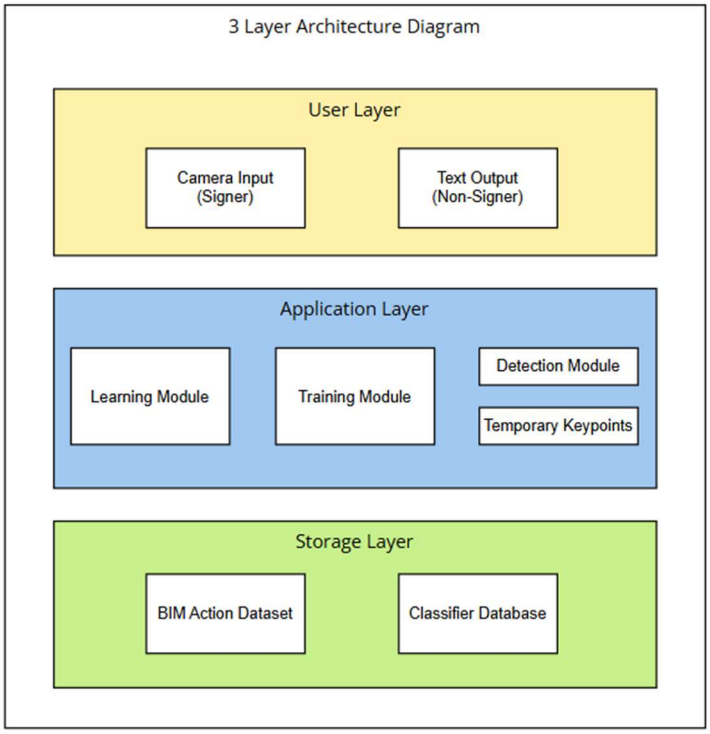


Figure 4.2: 3 Layer Architecture Diagram

4.4 Use Case Diagram

As shown in the use case diagram in Figure 4.3, the functional interaction between the BIM recognition system and external actors. The main actor is the BIM signers who make hand gestures (input) that the system can capture. The gestures are processed in the Detection Module. This processes the gestures for keypoint extraction. The model with transfer learning classifies the gestures. Finally, the recognized gestures are converted to text output. After that, the system gives the Translation Output to the Non-Signer; the secondary actor reads the text to understand the gestures. The figure illustrates how gestures are processed from input to recognition to translating for efficient communication between BIM Signer and Non-Signer in real-time.

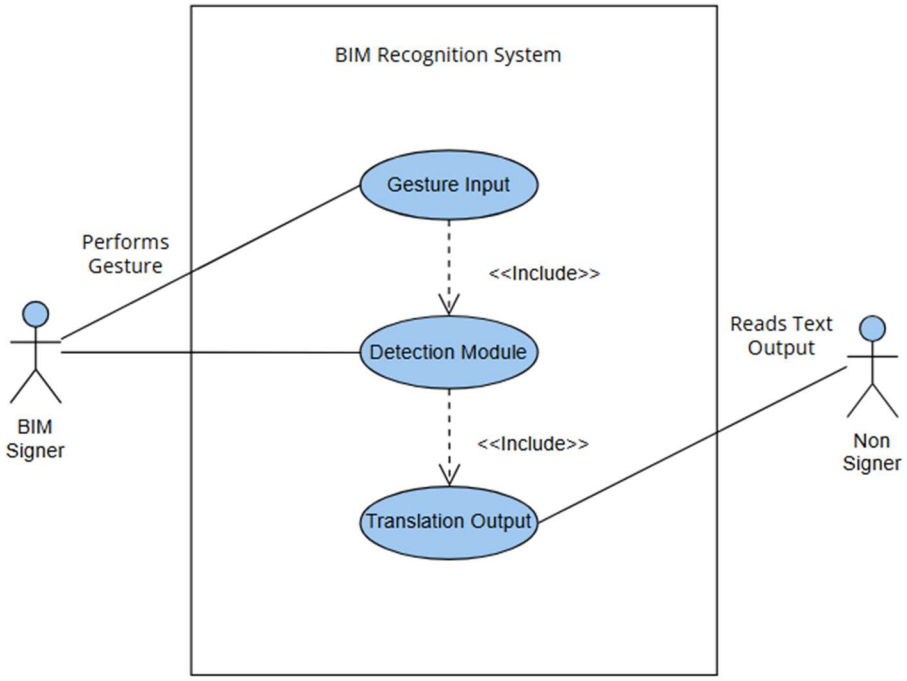


Figure 4.3: BIM Recognition System Use Case Diagram

4.5 User Interface (UI) Design

The user interface of the BIM recognition system consists of two primary screens: the Start Screen and the Recognition Screen. The Start Screen shown in Figure 4.4, is a simple, clean interface that features a “Start” and “Quit” button. These buttons allow the user to initiate the gesture recognition process or exit the window. Once “Start” is pressed, the system transitions seamlessly to the Recognition Screen shown in Figure 4.5, where the live video feed and text output are integrated into a single view. The live video feed displays the user’s gestures in real time, while the recognized gestures are translated and shown as text above. This design ensures that both gesture input and output are accessible within a unified screen.

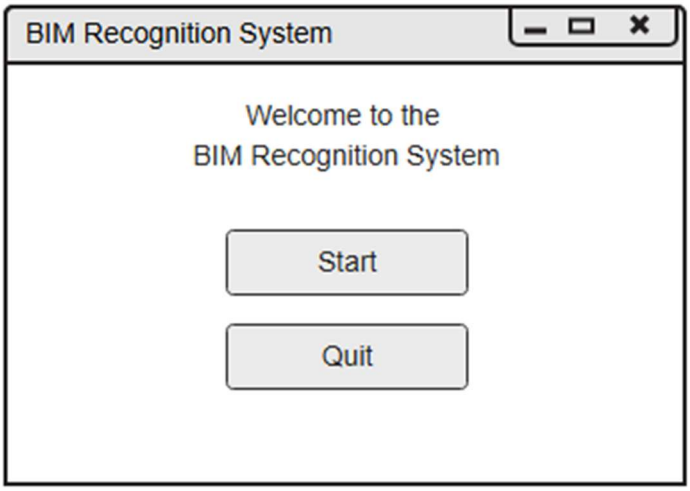


Figure 4.4: BIM Recognition System Start Screen

Figure 4.5: Live Feed Recognition Screen

The user flow begins at the Start Screen, where the user presses the "Start" button to initiate the recognition process. The interface then transitions to the Recognition Screen, enabling the user to perform gestures while observing the live video feed and corresponding text output in real time. The system processes the gestures dynamically, providing instant feedback through the text output. This straightforward flow ensures an intuitive experience, with minimal steps required for effective interaction.

4.6 Process Design

The design process is a sequence that helps in the recognition of gestures in a BIM recognition system. It starts with the initialization and finally displays the gestures as text. When the user hits “Start” button the system starts and goes for recognition. The camera that is connected to the system will generate real time gestures. The video frames are processed using MediaPipe to extricate the keypoints. The neural network model that has already been trained is fed the points of the gesture to identify the gesture. The model can tell the meaning of the gesture, and a text appears on the recognition screen in real-time. The loop

continues processing gestures until the user says it to stop. The process is represented graphically through the activity diagram in Figure 4.6.

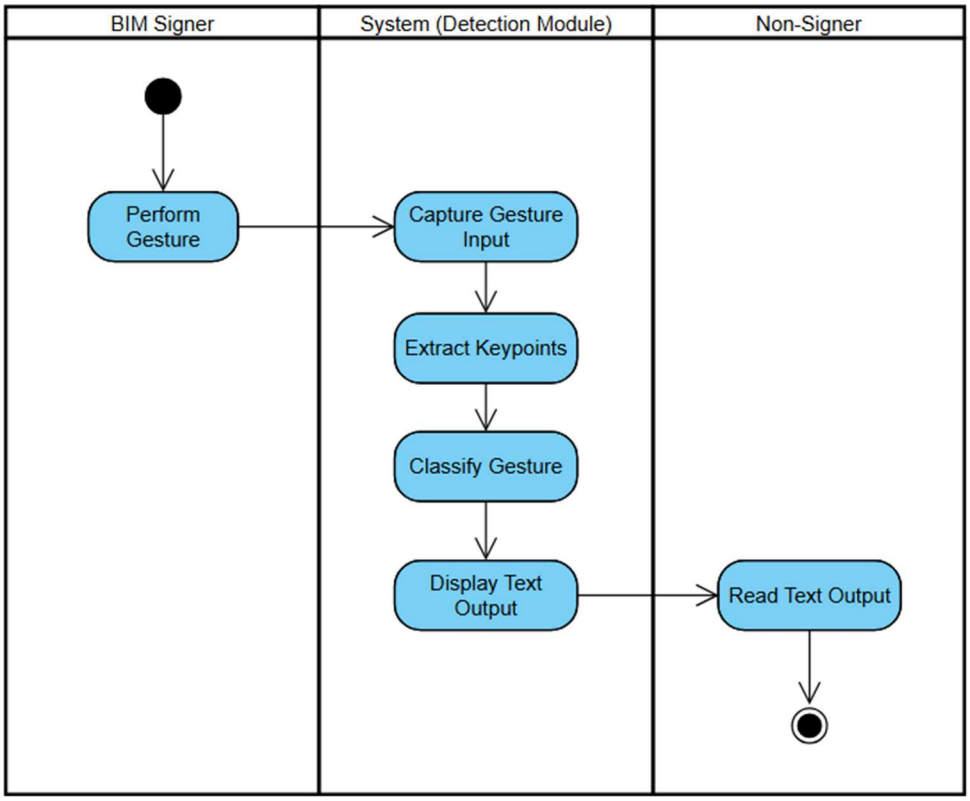


Figure 4.6: Activity Diagram

4.6.1 System Workflow

The system workflow is a data flow of the BIM recognition system. When a user makes a gesture, the gesture is captured and sent to the Detection Module. Here, MediaPipe takes the video frames and extracts keypoints. The keypoints will then be classified according to the model from the Classifier Database. The recognized gesture is then translated into text by the system which is displayed to the user in real time. The data flow diagram in Figure 4.7 below illustrates how the gesture input passes through different sections of the system before displaying it as text.

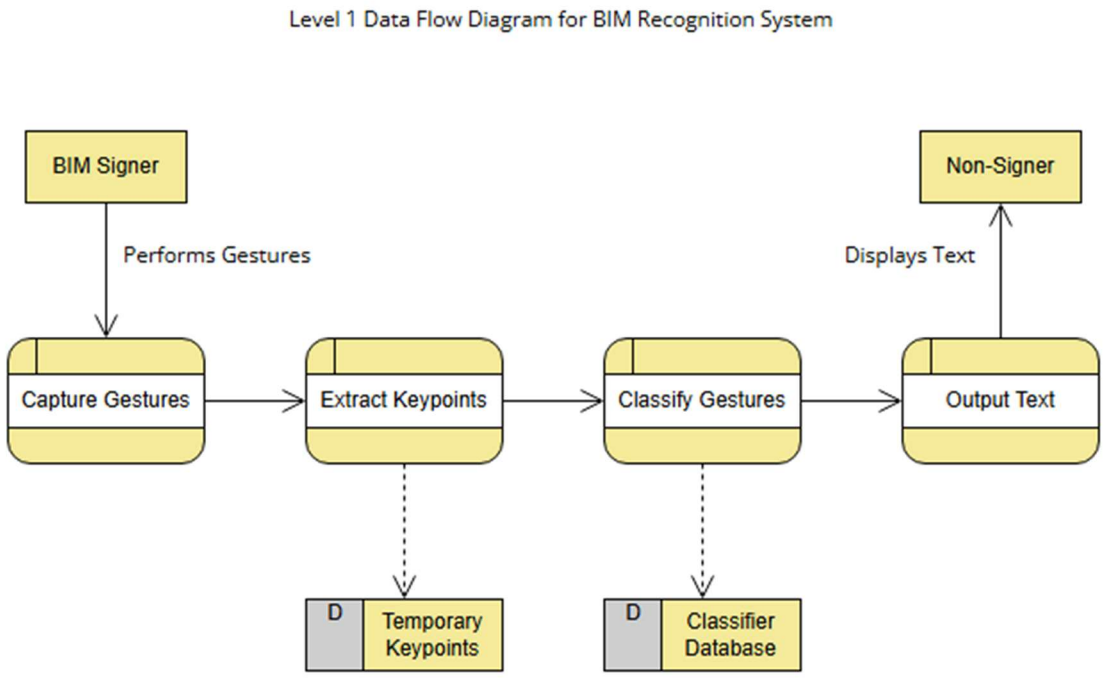


Figure 4.7: Level 1 Data Flow Diagram for BIM Recognition System

4.7 Class Diagram

The class diagram given in Figure 4.8 below shows the significant components of the BIM recognition system and their workings. The BIMRecognitionSystem works as the master controller for the LearningModule, TrainingModule and DetectionModule. The gesture data is collected and stored in KeypointsDataset, in which the key points are used for training and recognition. The system's TrainingModule trains multiple neural network models with the help of the dataset and then saves it in the system. On the other hand, the DetectionModule uses the trained model for gesture recognition in real-time.

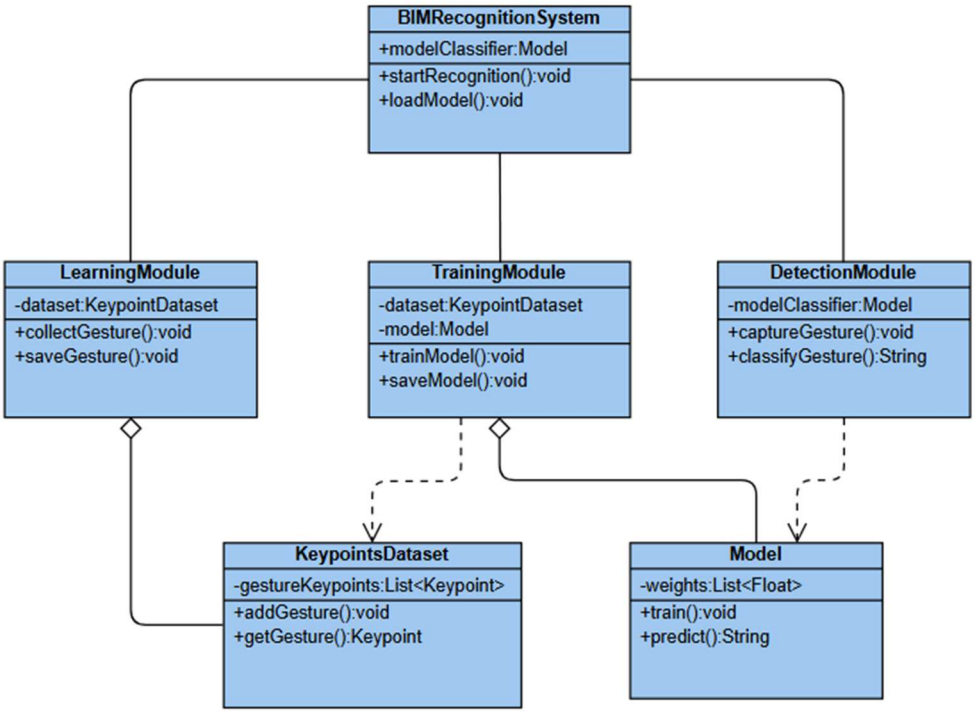


Figure 4.8: Class Diagram

4.8 Sequence Diagram

The sequence diagram in Figure 4.9 below, shows the flow of interactions in the BIM Recognition System. The BIM Signer performs a gesture, which is captured by the system and processed by the Detection Module. The module sends the gesture data to the Trained Model for classification and receives the predicted gesture. The recognized gesture is then sent back to the system and displayed as text output for the Non-Signer to read, enabling real-time communication.

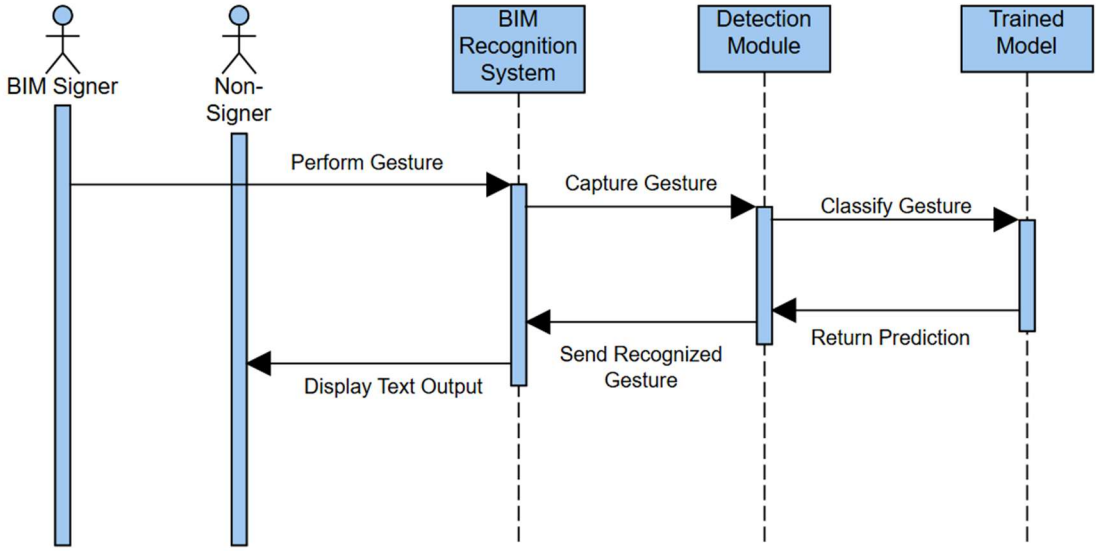


Figure 4.9: Sequence Diagram

4.9 Component Design

The component design of the BIM recognition system defines the key functional modules and their interactions within the system. The system is structured into three primary components as shown in Figure 4.10 below: the Learning Module, the Training Module, and the Detection Module, each responsible for different stages of gesture recognition.

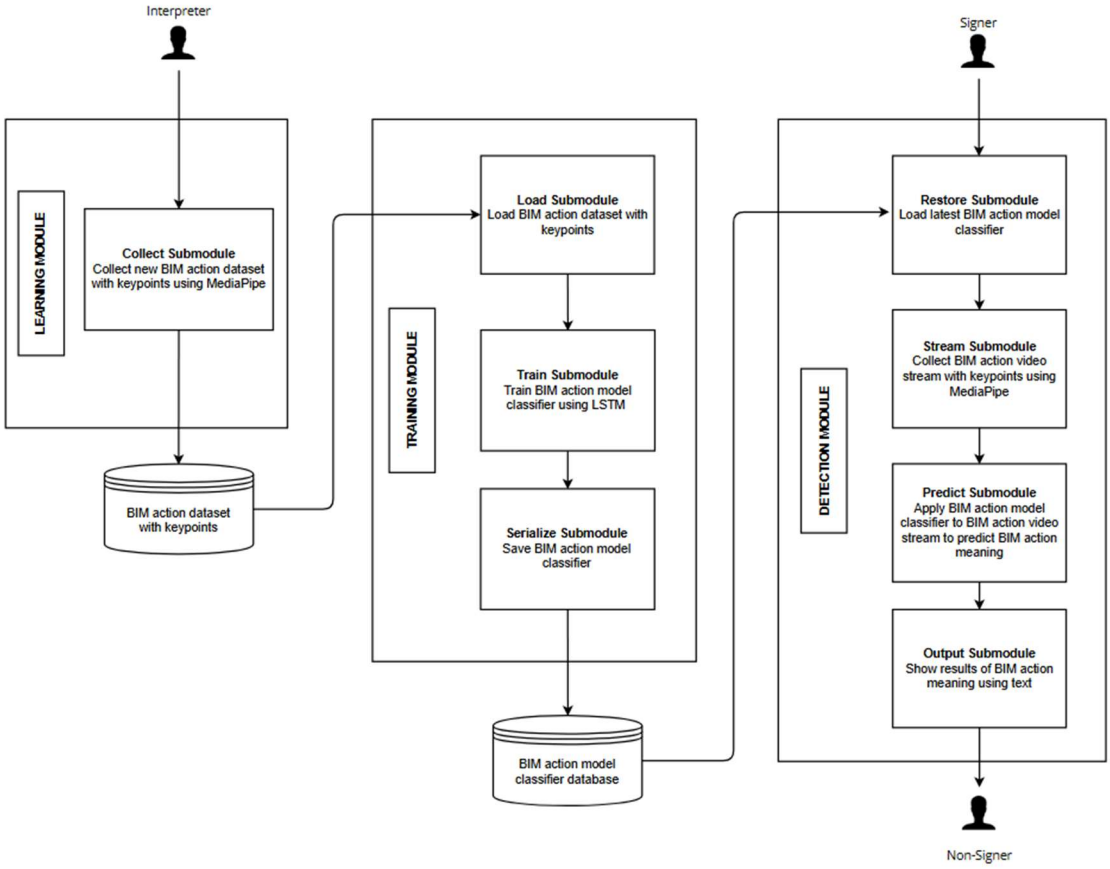


Figure 4.10: Complete Logical Architecture Diagram

4.9.1 Learning Module

The Learning Module shown in Figure 4.11 below allows the capture and storage of gesture keypoints to add new BIM vocabularies. When MediaPipe generates the keypoints for a new action, the keypoints are labelled and added to the BIM Action Dataset for future training.

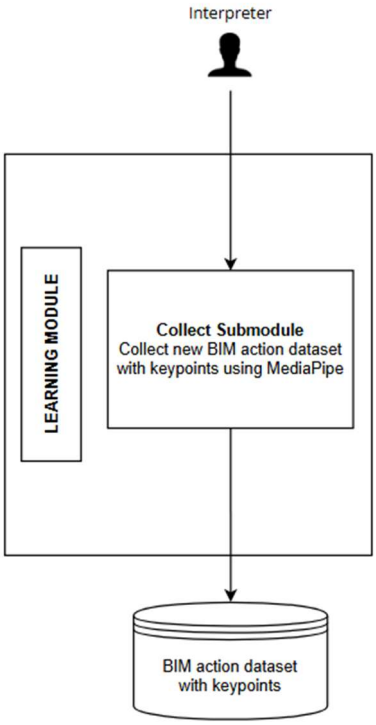


Figure 4.11: Learning Module Diagram

Collecting new gestures of interpreter will be done by Collect Submodule. The system uses MediaPipe to extract keypoints from the gestures of the interpreter. The BIM Action Dataset stores keypoints for later training collection.

4.9.2 Training Module

The Training Module in Figure 4.12 below trains gesture recognition models using the keypoints stored in the BIM Action Dataset. This module makes sure that the system is able to recognize gestures accurately by generating and saving updated classifiers. The whole module consists of three submodules Load Submodule, Train Submodule and Serialize Submodule.

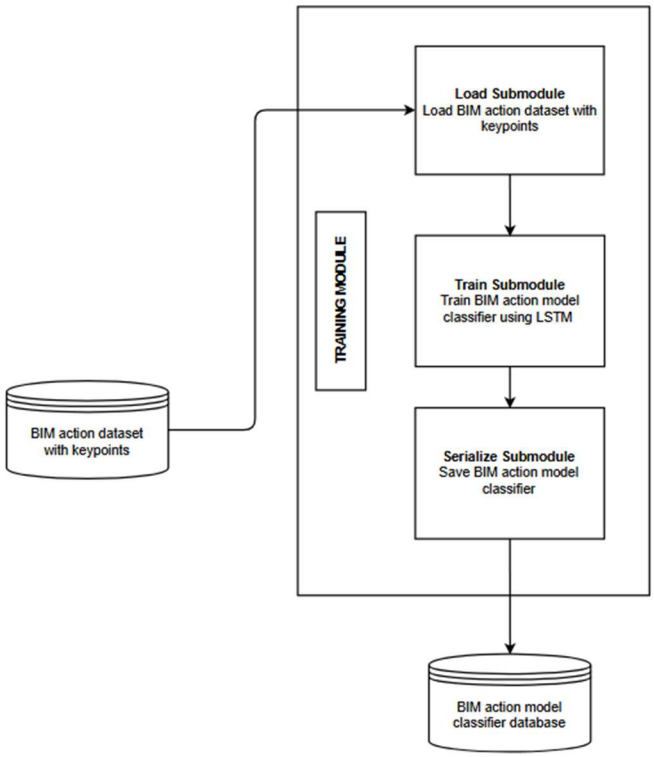


Figure 4.12: Training Module Diagram

The Load Submodule fetches the necessary BIM action dataset from the memory along with keypoints to start the training process. Once the necessary data is loaded, the Train Submodule applies multiple neural network models to the keypoints for the gesture model training. The model is fed the keypoints and their labels by the submodule which learns the movements and styles of the gestures.

Finally, serialized submodule saves the trained model in a serialized format for efficiency. The BIM Action Model Classifier Database stores the serialized model, which the Detection Module can readily access to recognize a gesture in real time. In conclusion, the training module will take care of the comprehensive lifecycle of Action BIM models and their scaling in the system.

4.9.3 Detection Module

The Detection Module in Figure 4.13 is responsible for recognizing gestures in real-time. It is the core unit that converts signer inputs into text outputs for communication. It interacts with Signer and Non-Signer. The main component of this module is composed of the four key submodules which are the Restore Submodule, Stream Submodule, Predict Submodule, and Output Submodule.

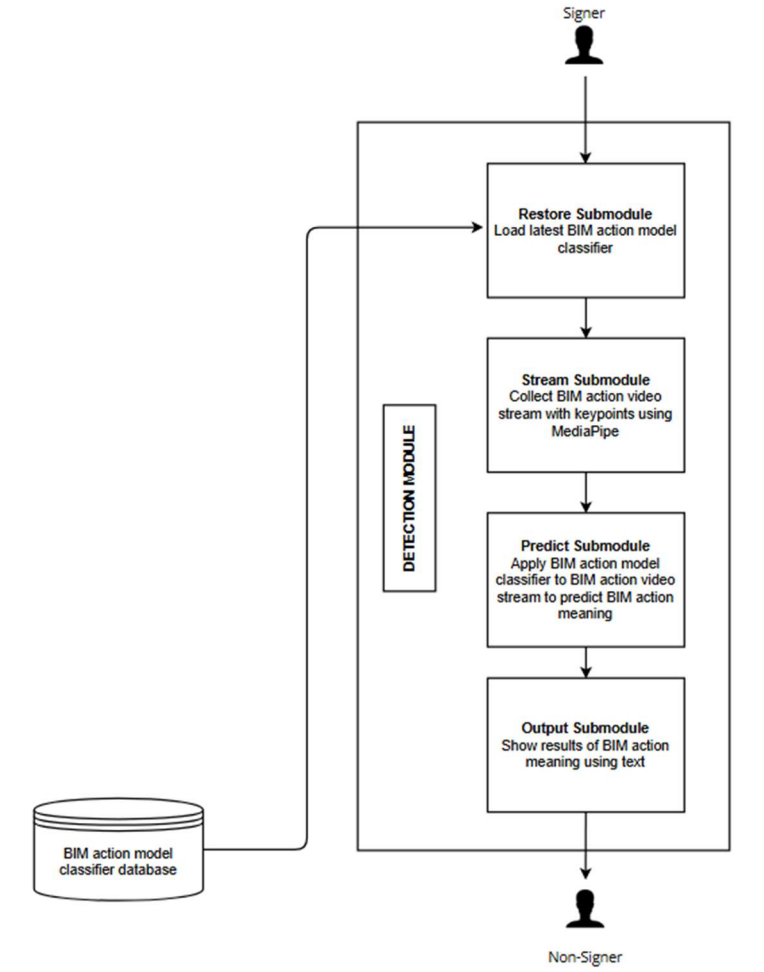


Figure 4.13: Detection Module Diagram

The process starts with the Restore Submodule which will loads the latest trained BIM action model classifier from Classifier Database. Once the model is restored the Stream Submodule utilizes MediaPipe to stream real-time video of the signer. This submodule extracts keypoints from the video.

The Predict Submodule receives these keypoints and applies the restored model to classify the gesture and predict its meaning. The model that has been trained is being used to analyze the keypoints and map them to gestures corresponding to it. Finally, the Output Submodule shows the text, allowing the non-signer to understand what the signer is doing. The recognition screen provides the output in a clear and understandable manner.

4.10 Summary.

The chapter presents the system design for the BIM recognition system, which includes system architecture, component design, and workflow design. The design of system including the role of Learning Module, Training Module and Detection Module has been discussed. Diagrams of the key processes were done for the better understanding which include class diagram, sequence diagram, and data flow diagram. This design provides an implementation-oriented approach that would allow for scalability and modularity in real-time gesture recognition.

Chapter 5: Implementation

5.1 Overview

In this chapter, we will discuss the implementation of the BIM recognition system. The implementation process involves system setup, data collection, keypoint extraction, model training, and real-time testing. These are crucial in ensuring that the system functions effectively in recognizing BIM gestures through a webcam feed. With the appropriate development environment and tools, the system can successfully be implemented to perform gesture recognition in real time.

5.2 Development Environment Setup

5.2.1 Tools and Platform Used

The development of the BIM recognition system was conducted on a personal computer running Windows 11 (Version 10.0.26100 Build 26100). The primary environment for coding and model training was Visual Studio Code (VSCode). VSCode is a lightweight yet powerful source code editor that supports Python development through various available extensions. Additional tools such as Jupyter Notebook were used for different stages of development such as data collection, modal training and evaluation. Figure 5.1 below shows the interface of VS Code used for development.

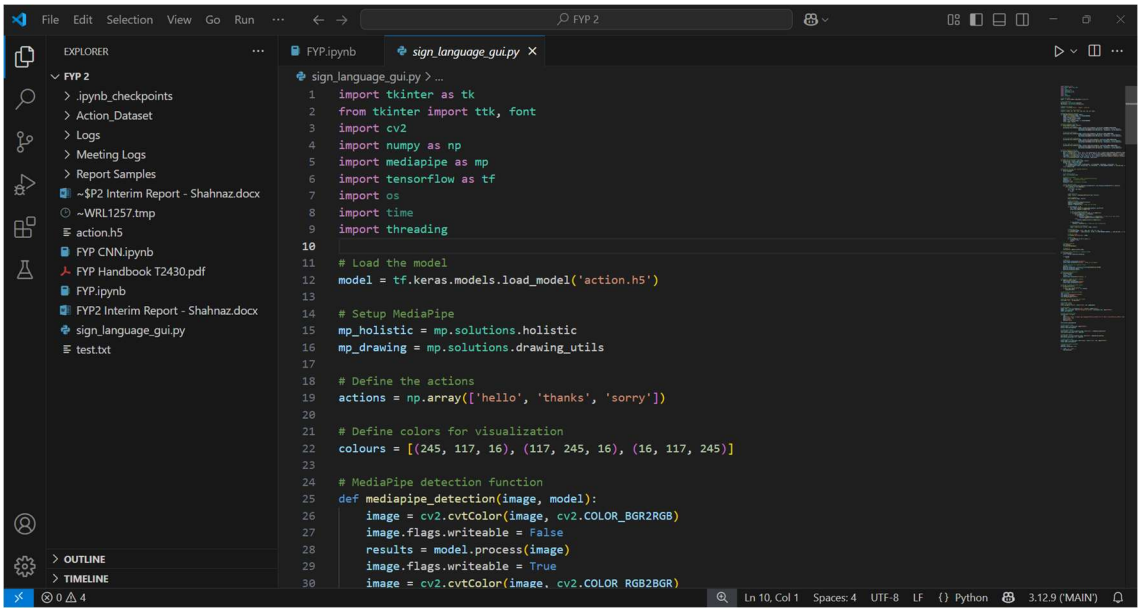


Figure 5.1: VS Code Interface

The Python environment was managed using Anaconda, which enabled the installation and maintenance of required libraries and dependencies. This combination of tools and platforms enabled efficient workflow, reproducibility, and ease of debugging throughout the project lifecycle.

Lastly, to support the training and evaluation process, TensorBoard was used as a visualization tool. It provided real time monitoring of the model's training progress. It's visualization of training and validation metrics, such as loss and accuracy, allowed for early detection of performance issues. The visualization output from TensorBoard is shown in Figure 5.2 below.

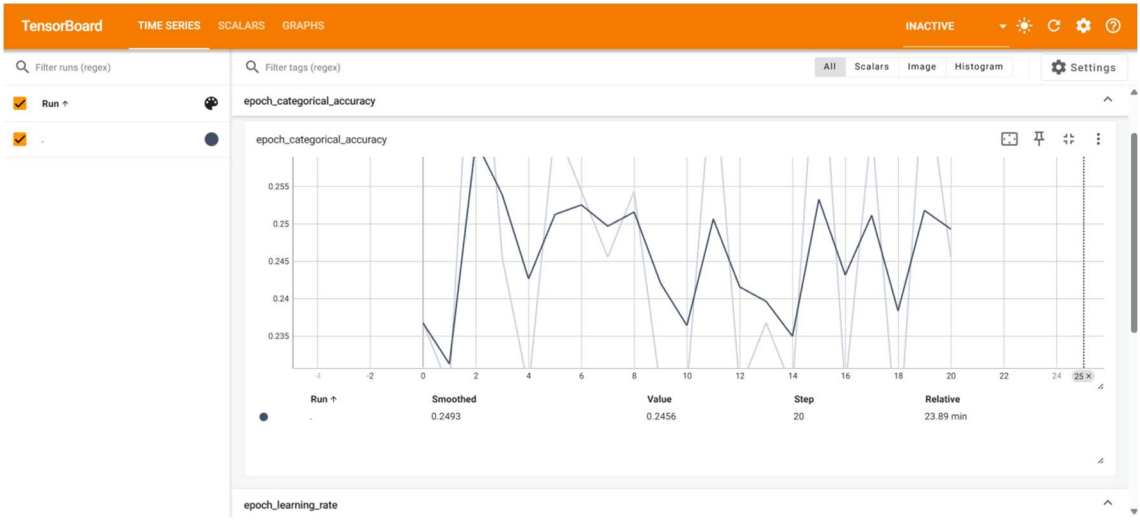


Figure 5.2: TensorBoard Interface

5.2.2 Programming Languages and Libraries

Python was chosen as the main programming language due to its simplicity and its extensive libraries available for machine learning and data processing. The Python environment was configured with the required libraries to support the implementation. These libraries are installed using the Python package installer (PIP). The installation and import commands are shown in the table below:

```
# installing necessary libraries
pip install tensorflow
pip install opencv-python
pip install mediapipe
pip install numpy
pip install matplotlib
pip install scikit-learn

# importing core libraries
import cv2
```

```
import numpy as np
from matplotlib import pyplot as plt
import time
import mediapipe as mp
import os
import tkinter as tk
from tkinter import ttk, font
import threading

# importing machine learning libraries
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import TensorBoard, EarlyStopping
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
```

These libraries serve various functions to the system. OpenCV and MediaPipe are used for video capture and keypoint detection, while Numpy and Matplotlib support numerical operations and data visualization. Tkinter provides the graphical user interface (GUI). For machine learning, Tensorflow and Keras are used to build and train the model, while Scikit-learn is used for data splitting and evaluation.

5.2.3 Software Stack and Version Details

To ensure compatibility, the following table shows the software stack and version details that were used throughout the project.

Software/Library	Version
Python	3.12.9
TensorFlow	2.19.0
Keras	3.9.0
Numpy	2.3.0
OpenCV	4.11.0
Matplotlib	3.10.1
Pandas	2.2.3
Mediapipe	0.10.21
Scikit-learn	1.6.1
Anaconda	24.11.3

5.3 Data Collection and Preprocessing

5.3.1 Data Source

The primary data source for this project was a custom-built dataset using a Python script to automate the collection and organization of sign language samples. This script created the necessary folder structure and captured frames for each sign instance, ensuring consistency across the dataset. Each action (such as "sorry", "drink," "eat, food," etc.) was represented as a separate folder within the main dataset directory. Within each action folder, there were multiple subfolders, each corresponding to a unique sample or sequence, and each containing a series of frames or data points representing that instance of the sign. The dataset includes 15 signs with 150 samples per action, with each sample consisting of 30 frames.

5.3.2 Data Cleaning Steps

To maintain data quality, several data cleaning steps were implemented. First, the dataset was reviewed to ensure that all action folders contained the correct number of samples and that each sample folder included the expected number of frames. Additionally, the naming conventions and directory structures were

standardized to prevent errors during data loading. These cleaning steps helped minimize inconsistencies, thereby improving the reliability of the training process.

5.3.3 Feature Engineering and Transformation

Feature engineering and transformation are crucial steps in preparing the raw data for input into the deep learning models. For each frame in a sample, key features are extracted using MediaPipe Holistic, which provides detailed keypoint coordinates for the hands, face, and body. These keypoints are flattened into a single feature vector per frame, resulting in a consistent feature dimension (e.g., 1662 features per frame). Each sample is represented as a sequence of 30 frames, each with its corresponding feature vector, forming a three-dimensional input array. The final preprocessed data is structured as arrays of shape (number of samples, 30, 1662), ready for input into the ensemble of neural network models (such as LSTM, BiLSTM, and GRU) used in this system

5.4 Model Implementation

5.4.1 Algorithms Used

The BIM recognition system consists of several main modules/algorithms that have been implemented to provide various functionalities to support the system. These modules work together to provide accurate BIM detection.

5.4.1.1 Algorithm 1: Data Collection

Algorithm 1 handles the process of collecting keypoint data from the webcam using Mediapipe and integrating it into a structured dataset. It ensures that the keypoints for each gesture (or sign) are captured frame by frame and saved as a .npy file for training. The following pseudocode shows the implementation of this algorithm.

Table 5.1: Algorithm 1 Pseudocode

```
Initialize camera capture
Initialize MediaPipe Holistic model with detection confidence = 0.5, tracking confidence = 0.5
FOR EACH action IN actions:
    FOR EACH sequence number:
        FOR EACH frame number in sequence:
            Capture frame from camera
            Process frame with MediaPipe to detect landmarks
            Draw landmarks on frame
            IF frame number == 0 THEN:
                Display "STARTING COLLECTION" on frame
                Display "Collecting frames for {action} video number {sequence}" on frame
                Pause briefly (2 seconds)
            ELSE:
                Display "Collecting frames for {action} video number {sequence}" on frame
            Extract keypoints from landmarks (pose, face, left hand, right hand)
            Save keypoints as numpy array to:
                DATA_PATH/action/sequence/frame number.npy
            IF user pressed 'q' key THEN:
                Exit loops
Release camera and close all OpenCV windows
END FUNCTION
```

The process begins by initializing the webcam and Mediapipe Holistic model, which is used to detect face, hand, and pose landmarks. The algorithm loops through a list of predefined actions and captures a fixed number of sequences for each action and each sequence consists of multiple frames. For every frame captured the algorithm detects the landmarks, extracts the keypoints

and saves them as numpy arrays. OpenCV displays visual cues on the window screen during data collection and a brief pause (2 seconds) occurs at the beginning of each new sequence to allow users to get ready. The process can be interrupted by pressing the ‘q’ key. The figures below are snapshots of algorithm 1 during execution.

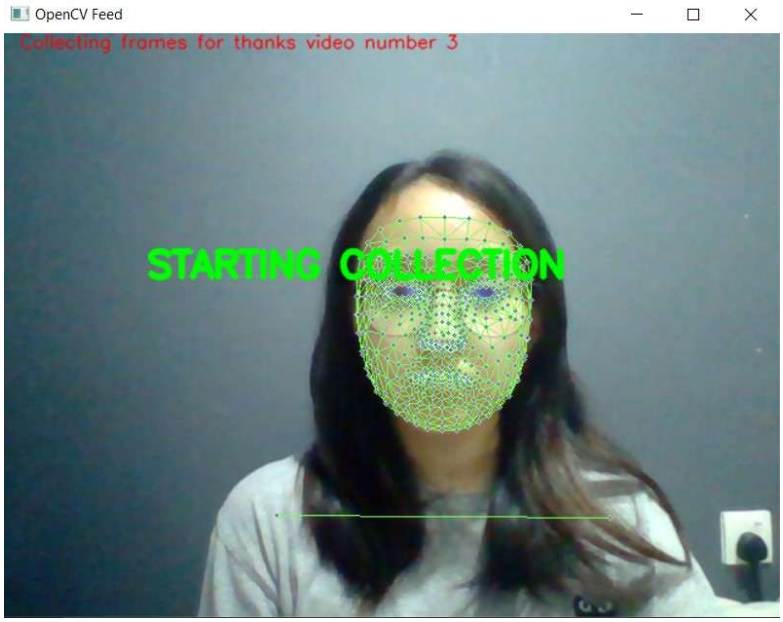


Figure 5.3: Starting Collection Frame for Algorithm 1

This first image in Figure 5.3 shows the “STARTING COLLECTION” frame, which appears at the beginning of each recording sequence. This visual cue informs the user that data collection is about to begin and gives them time to prepare for the gesture.

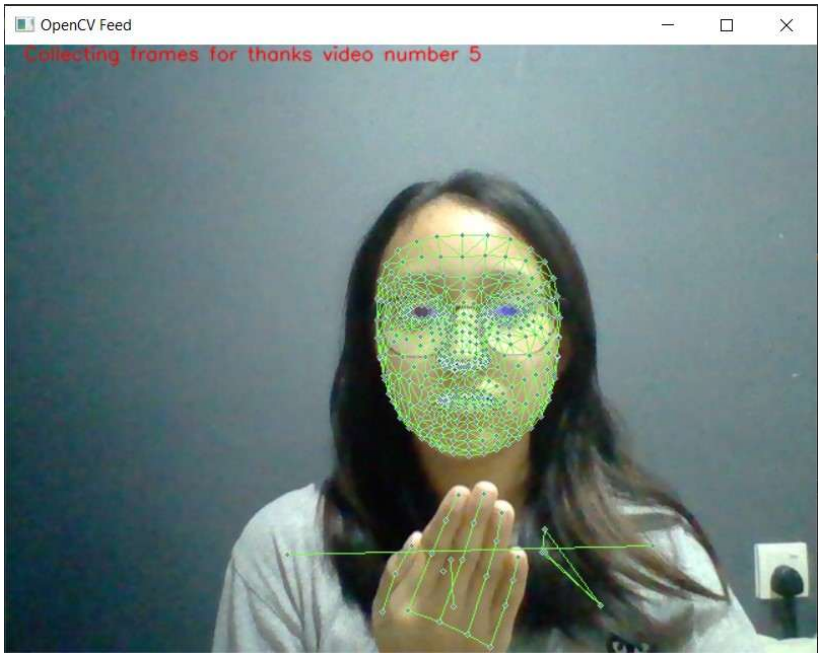


Figure 5.4: Example Signing Session for Data Collection

This second image in Figure 5.4 is an example of a frame from a signing session, where the MediaPipe Holistic model overlays landmarks on the user's face, hands, and body. These landmarks represent the extracted keypoints, which are then saved for model training.

5.4.1.2 Algorithm 2: Data Preparation

Algorithm 2 is responsible for preparing the dataset for training. It organizes the collected keypoint data stored in .npy files into sequences of frames, where each sequence represents one video sample of a sign gesture. These sequences are paired with their corresponding labels and stored in two separate lists, one for feature data and one for target labels. This structure allows the model to learn temporal patterns across frames for each action class. The pseudocode below illustrates the algorithm.

Table 5.2: Algorithm 2 Pseudocode

Initialize an empty list called sequences

```
Initialize an empty list called labels

Create a mapping called label_map:
    For each action in the list of actions,
        Assign a unique number to each action

For each action in the list of actions:
    For each sequence number from 0 to no_sequence - 1:
        Initialize an empty list called window

        For each frame number from 0 to sequence_length - 1:
            Load the .npy file for the current frame of the current action and sequence
            Append the loaded frame data to the window

        Append the complete window (sequence of frames) to sequences
        Append the corresponding label number (from label_map) to labels
```

The algorithm iterates through each action, sequence, and frame to load and group the data, then maps each action to a numerical label using a dictionary. This step is crucial for transforming raw keypoint data into a format suitable for training the deep learning model.

5.4.1.3 Algorithm 3: Detection Algorithm

Algorithm 3 handles the real time detection of hand and body gestures using the model trained on keypoints collected. It continuously captures frames from the webcam, performs landmark detection, processes the data into temporal sequences and uses the trained model to predict the gestures in real time. The detected gestures are then translated on screen providing immediate feedback

to the user. The following pseudocode below shows the implementation of this algorithm.

Table 5.3: Algorithm 3 Pseudocode

```
Initialize variables:
    sequence ← empty list        // To store the latest 30 frames of keypoints
    sentence ← empty list        // To store predicted action history
    predictions ← empty list     // To store recent predictions
    threshold ← 0.95             // Confidence threshold for accepting prediction

Start capturing video from webcam

Initialize MediaPipe Holistic model with detection and tracking confidence = 0.5


WHILE webcam is open:
    Read a frame from webcam


    Perform keypoint detection using MediaPipe
    Draw landmarks on the image


    Extract keypoints from the detection results
    Append keypoints to sequence
    Keep only the last 30 keypoints in sequence


    IF sequence has 30 frames:
        Predict the action using the trained model
        Get the highest predicted class index and its confidence
        Append predicted index to predictions list


        IF the most recent 10 predictions are consistent AND confidence > threshold:
            IF sentence is not empty:
                IF predicted action ≠ last sentence entry:
```

```
                Append predicted action to sentence
            ELSE:
                Append predicted action to sentence

        IF sentence has more than 5 entries:
            Keep only the last 5

        Visualize prediction probabilities on the image

    Draw prediction sentence text on the image

    Display the image in a window

    IF 'q' key is pressed:
        Exit loop

Release webcam and close display window
```

The algorithm starts by initializing several key variables: sequence for storing a rolling window of 30 frames, sentence for keeping track of predicted gestures, and predictions for storing recent outputs to ensure consistency. A confidence threshold of 0.95 is set to filter out low-confidence predictions.

The system activates the webcam and applies the MediaPipe Holistic model to each captured frame to extract pose, face, and hand landmarks. These landmarks are then converted into a flattened keypoint array and added to the sequence. Once the sequence contains 30 frames, it is passed to the trained model for classification.

The model returns a probability distribution over all possible gestures. If the same gesture is predicted consistently across the last 10 frames and its

confidence exceeds the threshold, it is added to the sentence list, avoiding repeated entries. The top five recent predictions are retained for clarity.

Visual feedback is provided through OpenCV, which overlays the detected gesture and its probability onto the video feed. The loop continues until the user presses the ‘q’ key, at which the system releases the webcam and closes all OpenCV windows. Figure 5.5 below shows the output generated by the algorithm during the detection of a sign.

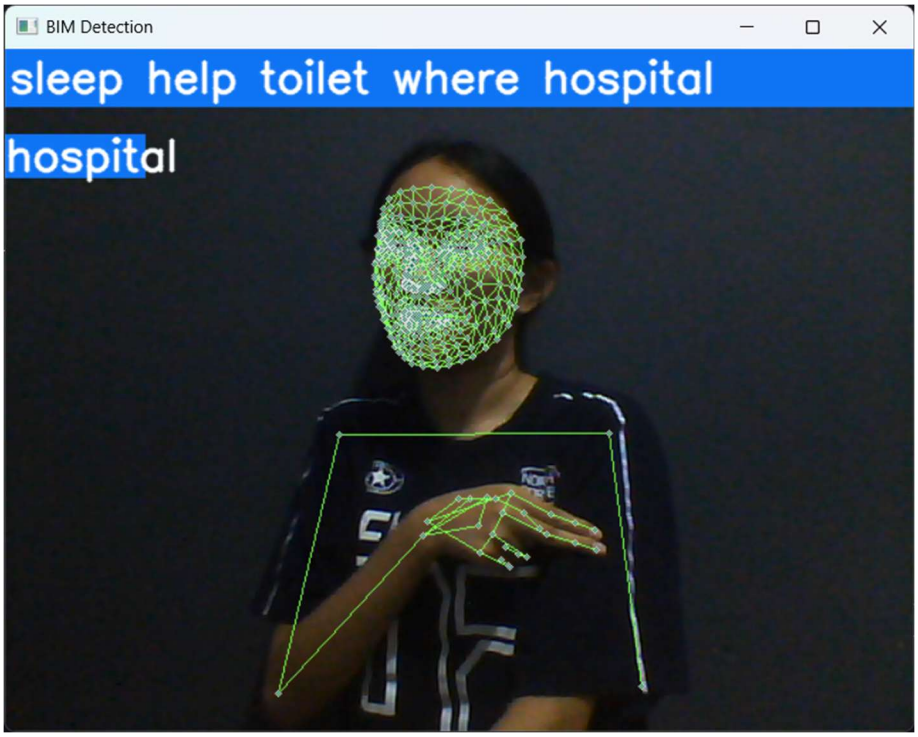


Figure 5.5: Output Generated during Real Time Prediction

In this implementation, the output sentence is updated from right to left, meaning the most recently recognized sign appears at the end (right side) of the sentence. This allows for a readable flow that resembles natural language typing, where new words appear at the end of the string.

5.4.2 Neural Network Models

The models in this system are constructed according to the proposed framework, focusing on RNN architectures that are well-suited for sequential gesture data. The three main models explored are LSTM, Bidirectional LSTM BiLSTM, and Gated Recurrent Unit (GRU). These models share a similar architecture and implementation, with the primary differences being in the type and arrangement of RNN layers.

5.4.2.1 LSTM

LSTM networks are a type of RNN specifically designed to capture long-term dependencies in sequential data. They use memory cells and gating mechanisms to control the flow of information, making them highly effective for modeling temporal patterns in gesture sequences. The LSTM model consists of an LSTM layer followed by dropout and batch normalization for regularization, and then dense layers for classification. The final dense layer uses the softmax activation function to output class probabilities. Figure 5.6 shows the model summary after running.

Table 5.4: LSTM Model Code

```
model1 = Sequential()
model1.add(LSTM(128, return_sequences=False, input_shape=(30, 1662)))
model1.add(Dropout(0.5))
model1.add(BatchNormalization())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(actions.shape[0], activation='softmax'))

model1.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history1 = model1.fit(X_train, y_train, epochs=400, callbacks=[tb_callback,
early_stopping], validation_data=(X_val, y_val))
```

```
model1.save('model1.h5')
```

Layer (type)	Output Shape	Param #
lstm_8 (LSTM)	(None, 128)	916,992
dropout_4 (Dropout)	(None, 128)	0
batch_normalization_3 (BatchNormalization)	(None, 128)	512
dense_16 (Dense)	(None, 64)	8,256
dense_17 (Dense)	(None, 15)	975

Total params: 2,779,695 (10.60 MB)

Trainable params: 926,479 (3.53 MB)

Non-trainable params: 256 (1.00 KB)

Optimizer params: 1,852,960 (7.07 MB)

Figure 5.6: LSTM Model Summary

5.4.2.2 BiLSTM

Bidirectional LSTM (BiLSTM) extends the standard LSTM by processing the input sequence in both forward and backward directions. This allows the model to access both past and future context, which can improve performance for gestures where the entire sequence provides important information. The BiLSTM model consists of two Bidirectional LSTM layers, followed by dropout for regularization, and then dense layers for classification. The final dense layer uses the softmax activation function to output class probabilities. Figure 5.7 below shows the model summary after running.

Table 5.5: BiLSTM Model Code

```
model2 = Sequential()
model2.add(Bidirectional(LSTM(64, return_sequences=True),
input_shape=(30, 1662)))
model2.add(Dropout(0.4))
model2.add(Bidirectional(LSTM(32)))
model2.add(Dense(64, activation='relu'))
model2.add(Dense(actions.shape[0], activation='softmax'))

model2.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history2 = model2.fit(X_train, y_train, epochs=400, callbacks=[tb_callback,
early_stopping], validation_data=(X_val, y_val))
model2.save('model2.h5')
```

Layer (type)	Output Shape	Param #
bidirectional_5 (Bidirectional)	(None, 30, 128)	884,224
dropout_6 (Dropout)	(None, 30, 128)	0
bidirectional_6 (Bidirectional)	(None, 64)	41,216
dense_22 (Dense)	(None, 64)	4,160
dense_23 (Dense)	(None, 15)	975

Total params: 2,791,727 (10.65 MB)

Trainable params: 930,575 (3.55 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 1,861,152 (7.10 MB)

Figure 5.7: BiLSTM Model Summary

5.4.2.3 GRU

Gated Recurrent Unit (GRU) is a simplified variant of LSTM that combines the forget and input gates into a single update gate. GRUs are computationally more efficient and can perform as well as LSTMs on many sequence modeling tasks. The GRU model consists of a GRU layer, followed by dropout and batch normalization for regularization, and then dense layers for classification. The final dense layer uses the softmax activation function to output class probabilities. Figure 5.8 below shows the model summary after running.

Table 5.6: GRU Model Code

```
model3 = Sequential()
model3.add(Bidirectional(GRU(128, return_sequences=False),
input_shape=(30, 1662)))
model3.add(Dropout(0.5))
model3.add(BatchNormalization())
model3.add(Dense(64, activation='relu'))
model3.add(Dense(actions.shape[0], activation='softmax'))

model3.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history3 = model3.fit(X_train, y_train, epochs=400, callbacks=[tb_callback,
early_stopping], validation_data=(X_val, y_val))
model3.save('model3.h5')
```

Layer (type)	Output Shape	Param #
bidirectional_7 (Bidirectional)	(None, 256)	1,376,256
dropout_7 (Dropout)	(None, 256)	0
batch_normalization_4 (BatchNormalization)	(None, 256)	1,024
dense_24 (Dense)	(None, 64)	16,448
dense_25 (Dense)	(None, 15)	975

Total params: 4,183,087 (15.96 MB)

Trainable params: 1,394,191 (5.32 MB)

Non-trainable params: 512 (2.00 KB)

Optimizer params: 2,788,384 (10.64 MB)

Figure 5.8: GRU Model Summary

5.4.3 Training and Testing

The training and testing process began with splitting the dataset into training, validation, and testing subsets, using an 80:10:10 or 80:20 ratio to ensure robust evaluation on unseen data. Each sample, consisting of a sequence of 30 frames with 1662 keypoint features per frame, was used as input to the models. Multiple neural network architectures were trained using supervised learning, where the correct sign label for each sequence was provided. During training, the Adam optimizer was employed to update model weights, and categorical crossentropy was used as the loss function to measure the difference between predicted and actual labels. The training process was conducted over multiple epochs, with each model's performance monitored on a validation set to detect overfitting and guide early stopping if necessary. After training, all models were evaluated on the test set to assess their generalization capability and real-world performance. An

ensemble evaluation was performed to compare the models, and the best-performing model was selected for deployment in the real-time BIM recognition system.

5.4.4 Hyperparameter Tuning

In this project, several key hyperparameters were systematically explored to enhance the accuracy and generalization capability of the LSTM, BiLSTM, and GRU models used for BIM gesture recognition. The main hyperparameters considered included:

- Number of units in RNN layers: The number of hidden units in LSTM, BiLSTM, and GRU layers was varied (e.g., 32, 64, 128) to balance model complexity and learning capacity.
- Number of layers: Experiments were conducted with both single and stacked RNN layers to determine the optimal network depth for capturing temporal dependencies in gesture sequences.
- Dropout rate: Dropout rates ranging from 0.3 to 0.5 were tested to prevent overfitting and improve model robustness.
- Batch size: Different batch sizes (e.g., 16, 32, 64) were evaluated to find the best trade-off between training speed and model convergence.
- Number of epochs: The maximum number of training epochs was set high (e.g., 400), with early stopping employed to halt training when validation performance stopped improving.

The tuning process involved training each model with different combinations of these hyperparameters and monitoring their performance on the validation set. The combination that yielded the highest validation accuracy and lowest loss, while avoiding overfitting, was selected for the final model configuration.

Early stopping and TensorBoard visualization were used to further guide the tuning process, allowing for real-time monitoring of training and validation metrics. This systematic approach to hyperparameter tuning ensured that each model was optimized for both accuracy and generalization before being evaluated as part of the ensemble and for final deployment in the real-time BIM recognition system.

5.4.5 Performance Metrics Used

The primary performance metric used during training and evaluation was categorical accuracy, which measures the proportion of correctly predicted samples out of the total. After training, a confusion matrix was generated to analyze the model's performance across different sign classes and to identify any patterns of misclassification. Additional metrics such as precision, recall, and F1-score were calculated to provide a more comprehensive evaluation. These metrics collectively provided insight into the model's strengths and areas for improvement.

5.5 System Integration

The final system integrates the best-performing deep learning model with real-time video processing and user interface. The trained model is loaded into a Python application that utilizes OpenCV for live video capture and MediaPipe Holistic for extracting keypoints from each video frame. These keypoints are processed and fed into the model to generate gesture predictions in real time. The recognized gestures are then displayed as text on a graphical user interface built with Tkinter, providing immediate feedback to the user. This seamless integration of machine learning, computer vision, and user interface technologies ensures that the system operates efficiently and accurately in real-world scenarios. Figure 5.9 below represents the system's initial interface where the

user is presented with options to either start the detection module or quit the system. It acts as a simple user menu before initiating the webcam and real-time processing.

Figure 5.9: Initial Start Screen

5.7 Summary

In summary, chapter 5 details the comprehensive implementation of the real-time BIM recognition system. The development process began with setting up a robust environment using Python, Anaconda, and essential libraries such as TensorFlow, Keras, OpenCV, and MediaPipe. Data collection was performed using a custom script to capture and organize gesture samples, ensuring consistency and quality through careful data cleaning and preprocessing. Keypoint features were extracted from video frames using MediaPipe Holistic, and each gesture sample was represented as a sequence of 30 frames with 1662 features per frame.

The system leverages advanced deep learning models, LSTM, BiLSTM, and GRU, each tailored to capture the temporal dynamics of sign language

gestures. These models were trained and validated using supervised learning, with hyperparameters such as the number of units, layers, dropout rates, batch size, and epochs systematically tuned for optimal performance. Early stopping and TensorBoard were employed to monitor and guide the training process.

Model performance was rigorously evaluated using metrics including accuracy, precision, recall, F1-score, and confusion matrix. An ensemble evaluation was conducted to compare the models, and the best-performing model was selected for deployment.

Finally, the system was integrated into a user-friendly application that combines real-time video capture, keypoint extraction, gesture prediction, and a graphical user interface. This seamless integration ensures efficient, accurate, and accessible BIM gesture recognition, providing immediate feedback to users and demonstrating the system’s effectiveness in real-world scenarios.

Chapter 6: Testing & Evaluation

6.1 Data Validation

6.1.1 Data Quality Checks

The initial phase of data quality checks involved addressing potential issues with incomplete or missing landmark detections. A handling mechanism was implemented where if any keypoints were not detected in a given frame, the corresponding keypoint arrays were initialized with zeros. This strategy not only prevented processing errors but also ensured that the feature vectors maintained consistent across all samples.

Additionally, the dataset structure was organized using a try-except block to create directories for each action and sequence. This helped avoid issues with file saving or folder creation errors. While not explicitly programmed, an important aspect of quality assurance was the real-time visual inspection of video frames. This manual verification allowed the immediate detection of anomalies such as incorrect landmark placements, poor lighting conditions, or mislabelled actions.

6.1.2 Data Preprocessing Validation

During preprocessing, the raw keypoint data was converted into a format suitable for training. Each video frame was processed to produce a 1662 length feature vector by combining pose, face, and both hand keypoints using NumPy functions like flatten() and concatenate(). This step ensured that all frames had consistent structure and matched the input requirements of the model.

To prepare the labels, a dictionary was created to assign each action a unique number. These numerical labels were then converted into one-hot encoded vectors. This validated that each action was properly encoded and usable by the model.

Lastly, the sequence data was compiled into a NumPy array, and its shape was verified to match the model's expected input dimensions of (30, 1662). These checks helped confirm that all data was correctly preprocessed and ready for model training.

6.1.2 Data Splitting

To evaluate the model, the dataset was divided into separate training, validation, and testing sets. This structure allowed the model to be trained on one part of the data, tuned and monitored using a second part (validation), and finally evaluated on a third, unseen part (test set) to assess its ability to generalize.

The splitting was done using the train_test_split function from the sklearn.model_selection module. Initially, the dataset was split into 80% for training and 20% as a temporary set. The stratify=y_labels parameter was used to ensure that all classes were evenly represented in both subsets, which is important for preventing any class imbalance. Next, the temporary 20% was split again into two equal parts, 10% for validation and 10% for testing. Stratification was applied once more to maintain class balance.

6.2 Model Evaluation

6.2.1 LSTM Model Results

The model achieved an accuracy of 94.67% on the test dataset, indicating strong performance in recognizing BIM gestures. Figure 6.1 and Figure 6.2 below shows the training curve and classification report for the model.

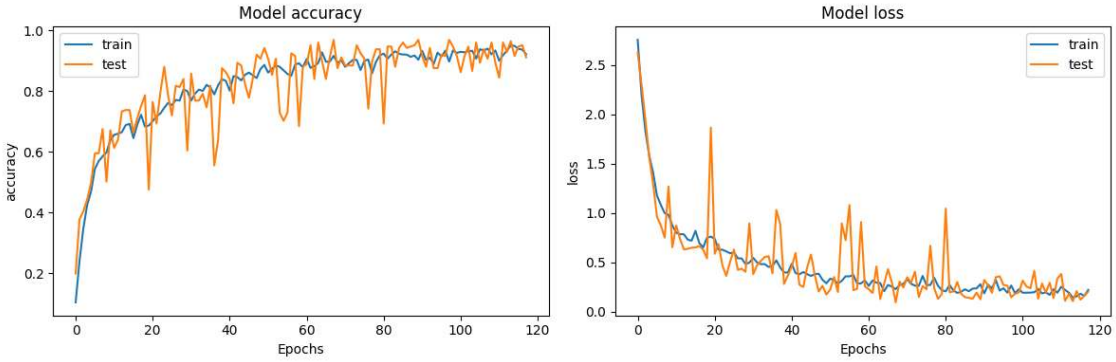


Figure 6.1: LSTM Training Curve

LSTM Classification Report:

	precision	recall	f1-score	support
drink	0.88	1.00	0.94	15
eat, food	0.93	0.87	0.90	15
family	1.00	1.00	1.00	15
go	1.00	0.93	0.97	15
happy	1.00	1.00	1.00	15
help	1.00	1.00	1.00	15
hospital	1.00	1.00	1.00	15
medicine	1.00	1.00	1.00	15
no	0.91	0.67	0.77	15
now	0.94	1.00	0.97	15
sleep	1.00	0.93	0.97	15
thankyou	0.94	1.00	0.97	15
toilet	1.00	0.87	0.93	15
where	0.94	1.00	0.97	15
yes	0.74	0.93	0.82	15
accuracy			0.95	225
macro avg	0.95	0.95	0.95	225
weighted avg	0.95	0.95	0.95	225

Figure 6.2: LSTM Classification Report

The LSTM model demonstrated strong performance in the classification task, achieving an overall test accuracy of 95%. The precision, recall, and F1-scores were consistently high for most classes, with many values reaching 1.00, indicating that the model was able to correctly identify and distinguish between the majority of categories. However, a few classes, such as “no” and “yes”, exhibited slightly lower precision and recall, suggesting that the model found these particular classes more challenging, possibly due to similarities with other

classes or limited representative samples. The macro and weighted averages for precision, recall, and F1-score were all 0.95, reflecting balanced and robust performance across the dataset.

The learning curves further support the effectiveness of the LSTM model. Both the training and testing accuracy increased steadily and closely followed each other throughout the epochs, while the loss curves consistently decreased. This close alignment between training and testing metrics indicates that the model generalizes well to unseen data and does not suffer from significant overfitting or underfitting. Although there were some fluctuations in the test accuracy and loss, these are expected and may be attributed to the relatively small sample size or class imbalance. Overall, the LSTM model proved to be a reliable and accurate approach for this classification problem, providing strong generalization and robust predictive performance.

6.2.2 BiLSTM Model Results

The model achieved an accuracy of 98.22% on the test dataset. Figure 6.3 and Figure 6.4 below shows the training curve and classification report for the model.

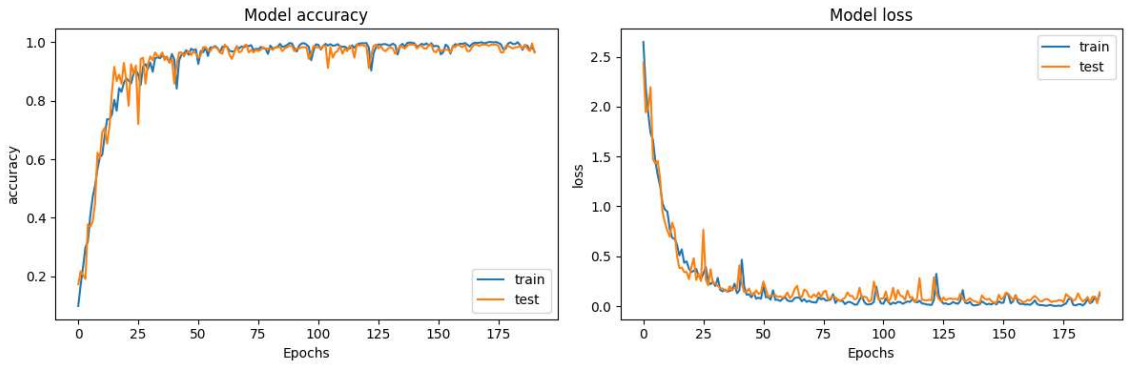


Figure 6.3: BiLSTM Training Curve

BiLSTM Classification Report:

	precision	recall	f1-score	support
drink	1.00	1.00	1.00	15
eat, food	1.00	1.00	1.00	15
family	1.00	1.00	1.00	15
go	1.00	1.00	1.00	15
happy	1.00	1.00	1.00	15
help	0.94	1.00	0.97	15
hospital	1.00	1.00	1.00	15
medicine	1.00	0.93	0.97	15
no	0.94	1.00	0.97	15
now	1.00	1.00	1.00	15
sleep	0.94	1.00	0.97	15
thankyou	1.00	1.00	1.00	15
toilet	1.00	0.87	0.93	15
where	0.94	1.00	0.97	15
yes	1.00	0.93	0.97	15
accuracy			0.98	225
macro avg	0.98	0.98	0.98	225
weighted avg	0.98	0.98	0.98	225

Figure 6.4: BiLSTM Classification Report

The BiLSTM model exhibited outstanding performance in the classification task, achieving an overall test accuracy of 98%. The precision, recall, and F1-scores for nearly all classes were perfect (1.00), indicating that the model was able to accurately and consistently identify each category. Only a few classes, such as "help," "no," "sleep," "toilet," and "where," showed slightly lower precision or recall values (0.94 or 0.87), but their F1-scores remained high, reflecting only minor misclassifications. The macro and weighted averages for precision, recall, and F1-score were all 0.98, demonstrating the model's balanced and robust performance across all classes.

The learning curves further reinforce the effectiveness of the BiLSTM model. Both the training and testing accuracy curves rapidly increased and plateaued at high values, while the loss curves for both sets decreased and stabilized at low values. The close alignment between the training and testing curves suggests that the model generalizes very well to unseen data, with no significant signs of overfitting or underfitting. Overall, the BiLSTM model proved to be highly reliable and accurate for this classification problem, offering excellent generalization and predictive capabilities across all target classes.

6.2.3 GRU Model Results

The model achieved an accuracy of 97.33% on the test dataset. Figure 6.5 and Figure 6.6 below shows the training curve and classification report for the model.

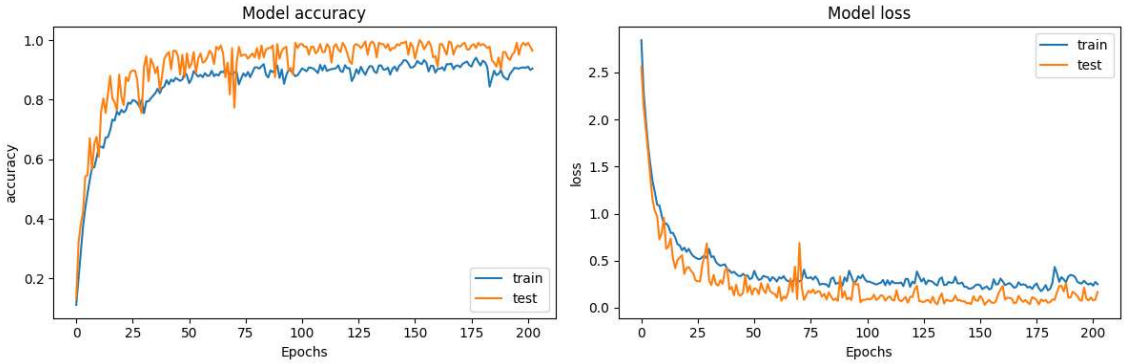


Figure 6.5: BiLSTM Training Curve

GRU Classification Report:

	precision	recall	f1-score	support
drink	1.00	1.00	1.00	15
eat, food	1.00	1.00	1.00	15
family	0.94	1.00	0.97	15
go	1.00	0.87	0.93	15
happy	1.00	1.00	1.00	15
help	0.93	0.87	0.90	15
hospital	1.00	1.00	1.00	15
medicine	0.93	0.93	0.93	15
no	1.00	1.00	1.00	15
now	0.88	1.00	0.94	15
sleep	1.00	1.00	1.00	15
thankyou	0.94	1.00	0.97	15
toilet	1.00	0.93	0.97	15
where	1.00	1.00	1.00	15
yes	1.00	1.00	1.00	15
accuracy			0.97	225
macro avg	0.97	0.97	0.97	225
weighted avg	0.97	0.97	0.97	225

Figure 6.6: BiLSTM Classification Report

The Gated Recurrent Unit (GRU) model also demonstrated excellent performance in the classification task, achieving an overall test accuracy of 97%. The precision, recall, and F1-scores were high across almost all classes, with many classes reaching perfect scores of 1.00. A few classes, such as "family," "help," and "now," showed slightly lower precision or recall values, with "help" having a recall of 0.87 and "now" a precision of 0.88, indicating some minor misclassifications. Nevertheless, the F1-scores for these classes remained strong, and the macro and weighted averages for precision, recall, and F1-score were all 0.97, reflecting the model's robust and balanced performance.

The learning curves for the GRU model further support its effectiveness. Both the training and testing accuracy increased rapidly and stabilized at high values, while the loss curves for both sets decreased and remained low throughout the epochs. The test accuracy and loss curves closely followed the training curves, suggesting that the model generalizes well to unseen data and does not suffer from significant overfitting or underfitting. Overall, the GRU model proved to be a highly reliable and accurate approach for this classification problem, offering strong generalization and predictive capabilities across all target classes.

6.2.4 Ensemble Model Results

The ensemble approach, which combines the predictions of the LSTM, BiLSTM, and GRU models, achieved the highest accuracy of 98.67%. This demonstrates the effectiveness of ensemble learning in improving classification robustness and overall system performance. Figure 6.7 below shows the ensemble learning classification report.

Ensemble Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	15
1	1.00	1.00	1.00	15
2	1.00	1.00	1.00	15
3	1.00	1.00	1.00	15
4	1.00	1.00	1.00	15
5	0.94	1.00	0.97	15
6	1.00	1.00	1.00	15
7	1.00	0.93	0.97	15
8	1.00	1.00	1.00	15
9	1.00	1.00	1.00	15
10	1.00	1.00	1.00	15
11	0.94	1.00	0.97	15
12	1.00	0.87	0.93	15
13	0.94	1.00	0.97	15
14	1.00	1.00	1.00	15
accuracy			0.99	225
macro avg	0.99	0.99	0.99	225
weighted avg	0.99	0.99	0.99	225

Figure 6.7: Ensemble Learning Classification Report

The ensemble model achieved the highest performance among all tested approaches, with an impressive overall test accuracy of 99%. The precision, recall, and F1-scores were nearly perfect for all classes, with most values at 1.00 and only a few slightly lower, indicating that the ensemble method was highly effective at correctly classifying each category. The macro and weighted averages for precision, recall, and F1-score were all 0.99, reflecting the model's exceptional and balanced performance across the entire dataset. This demonstrates that combining the predictions of multiple models can further enhance classification accuracy and robustness.

Despite the superior accuracy of the ensemble and BiLSTM models, the GRU model was selected for deployment in the real-time system. This decision was made because GRU networks are computationally more efficient and faster than BiLSTM models, making them better suited for real-time prediction scenarios where low latency is critical. While the BiLSTM achieved slightly higher accuracy, the GRU model still provided excellent predictive performance with

97% accuracy, ensuring reliable results while meeting the real-time requirements of the application.

6.2.5 Model Confusion Matrices

The confusion matrices for the LSTM, BiLSTM, GRU, and Ensemble models provide a detailed visualization of each model's classification performance across all classes. For all models, the majority of predictions are concentrated along the diagonal, indicating that most samples were correctly classified. The LSTM confusion matrix in Figure 6.8 (a) below shows a few off-diagonal entries, particularly for the classes "no", "eat, food", "toilet" where some misclassifications occurred. This aligns with the slightly lower precision and recall observed for these classes in the classification report.

The BiLSTM and Ensemble models shown in Figure 6.8 (b) and Figure 6.8 (c) below exhibit nearly perfect confusion matrices, with almost all predictions falling on the diagonal and only one or two minor misclassifications for classes such as "medicine," and "toilet,". This demonstrates the superior discriminative power of these models, as also reflected in their high accuracy and F1-scores.

The GRU model's confusion matrix in Figure 6.8 (d) below is also highly diagonal, with only a few misclassifications, such as for the "help," and "go" classes. These errors are minimal and do not significantly impact the overall performance, as indicated by the high accuracy and F1-scores.

Overall, the confusion matrices confirm that all models are highly effective at distinguishing between the different classes, with the BiLSTM and Ensemble models achieving near-perfect classification. The few misclassifications that do occur are limited to specific classes and are relatively infrequent, further supporting the robustness and reliability of the models for this multi-class classification task.

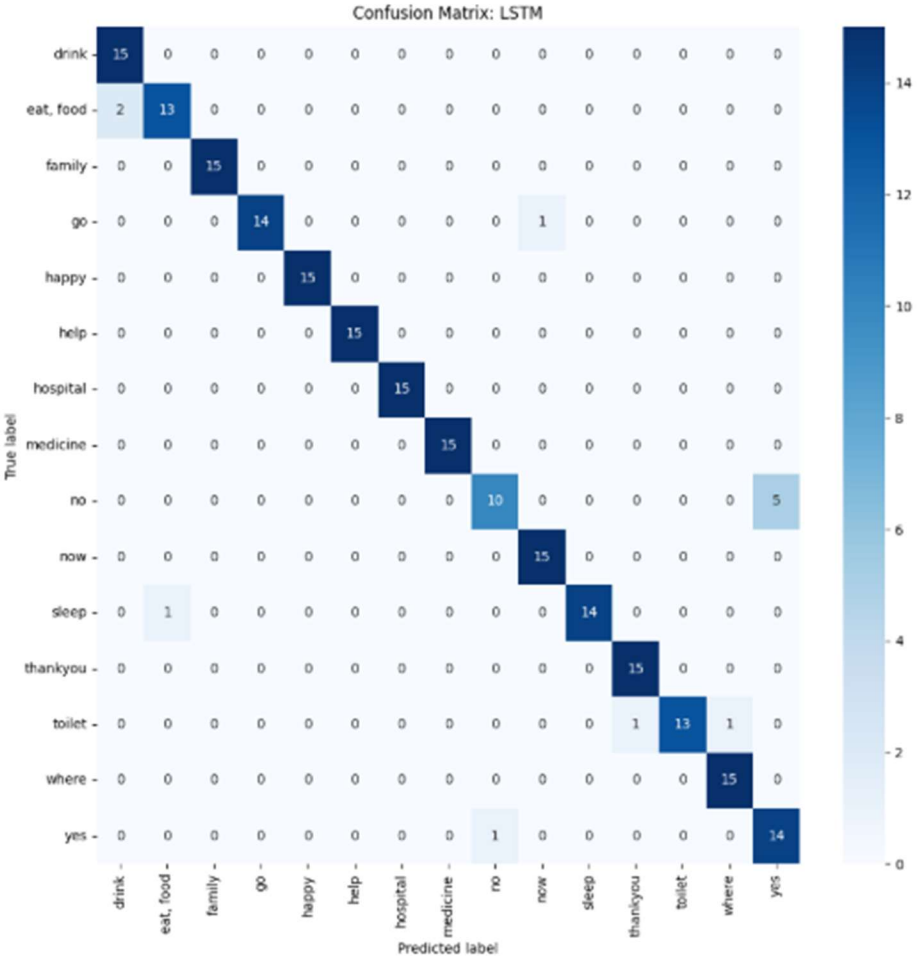


Figure 6.8 (a): LSTM Confusion Matrix

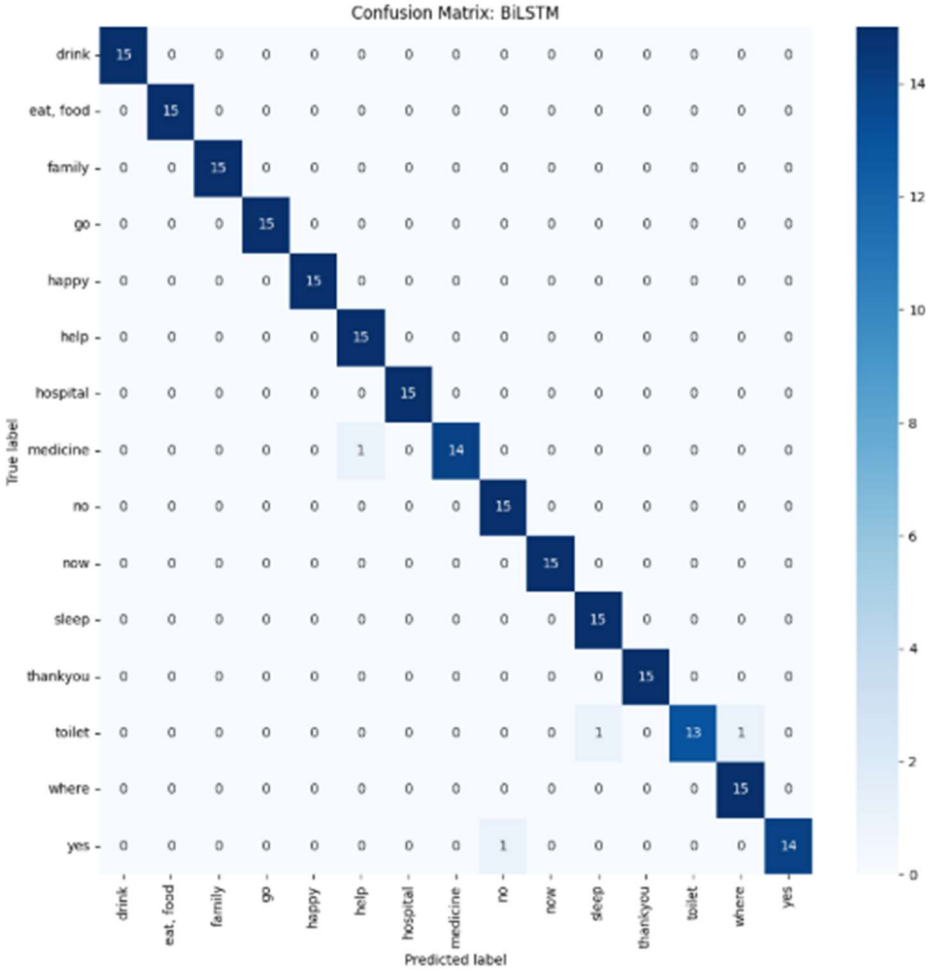


Figure 6.8 (b): BiLSTM Confusion Matrix

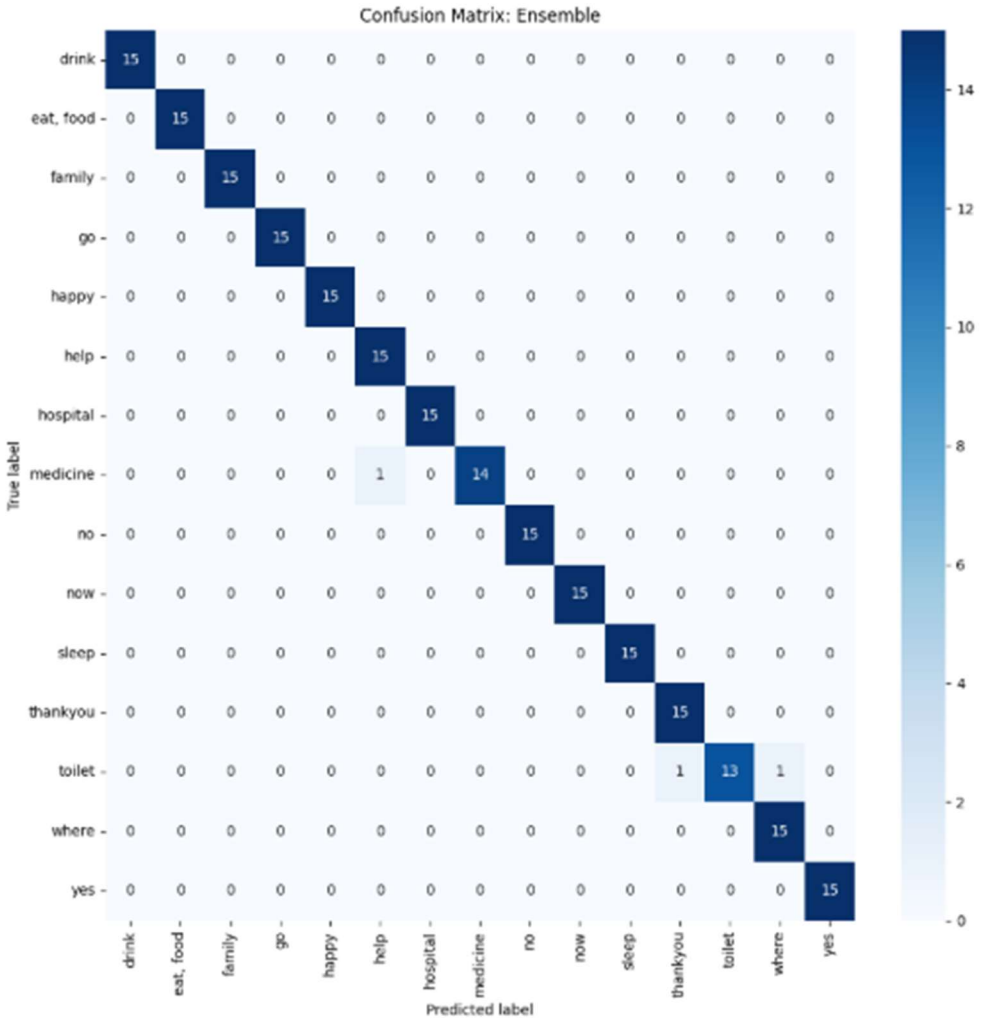


Figure 6.8 (c): Ensemble Confusion Matrix

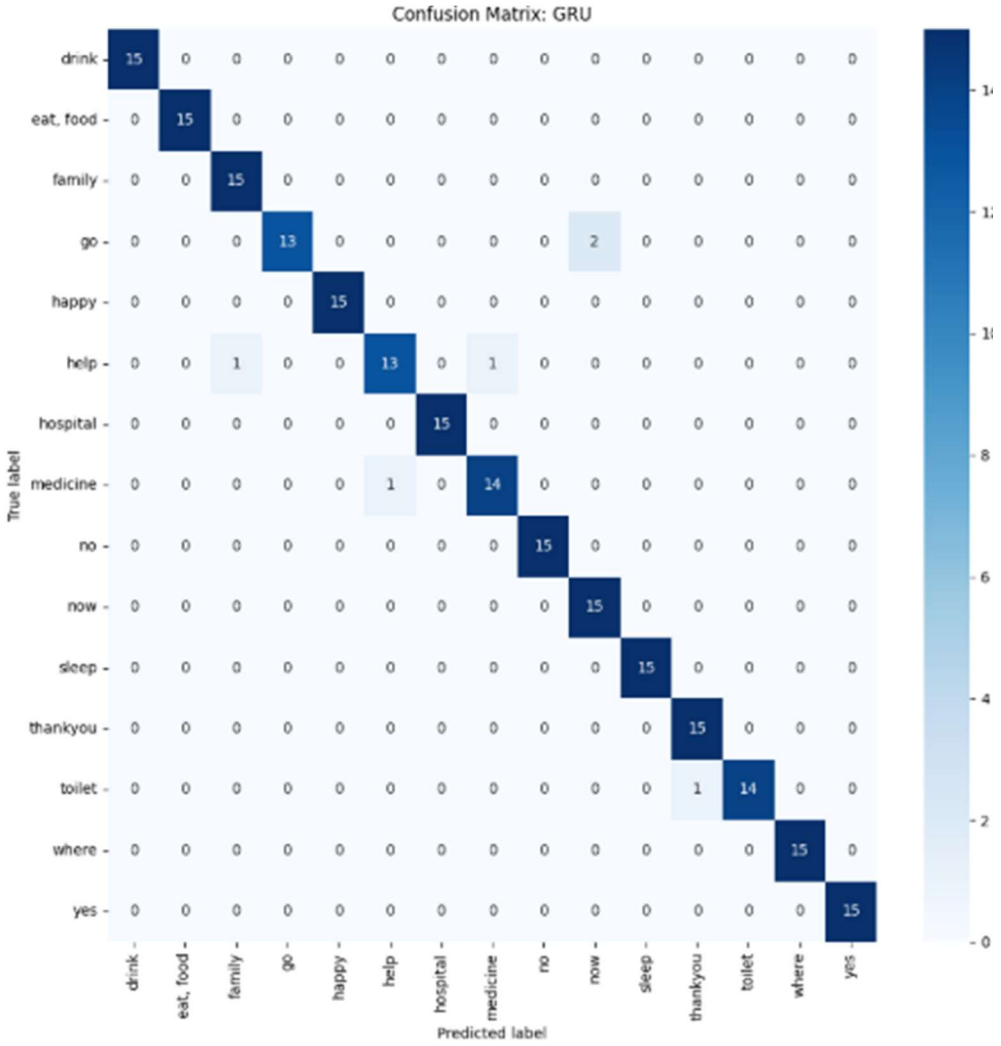


Figure 6.8 (d): GRU Confusion Matrix

6.2.6 Baseline Model Comparison

Table 6.1: Summary of All Models

Model	Accuracy	Precision	Recall	F1-Score
LSTM	0.9467	0.95	0.95	0.95
BiLSTM	0.9822	0.98	0.98	0.98
GRU	0.9733	0.97	0.97	0.97
Ensemble	0.9867	0.99	0.99	0.99

For baseline comparison, several previous works in sign language or gesture recognition have been considered. (Md, 2022) utilized a webcam-based system with both static and dynamic gestures, employing a MediaPipe and LSTM approach. However, their results did not specify the exact accuracy achieved, making direct comparison challenging. (Han Lin & Murli, 2022) implemented a system using an HD webcam for single-user, static gesture recognition with TensorFlow, and reported the use of data augmentation, but again, did not provide specific accuracy metrics. In contrast, (Khan et al., 2021) used a webcam for single-user, static gesture recognition with a CBAM-ResNet architecture, achieving an accuracy of over 90%. Compared to these baselines, the models developed in this work, particularly the BiLSTM, GRU, and ensemble approaches, demonstrate superior performance, with accuracies exceeding 97%. This highlights the effectiveness of the proposed methods, especially in dynamic gesture recognition scenarios, and underscores the advantage of using advanced recurrent neural network architectures for real-time sign language recognition.

6.3 Usability Testing

6.3.1 Test Setup

The usability of the system was evaluated using a webcam interface, with the GRU model from the ensemble evaluation deployed for real-time testing. During the test sessions, users performed BIM sign gestures in front of the webcam, and the system processed the video stream to extract keypoints and generate predictions in real time. The recognized gestures were immediately displayed as text overlays on the interface, simulating practical communication scenarios. This setup allowed for direct assessment of the system’s responsiveness, accuracy, and user experience in a realistic, interactive environment.

6.3.2 Feedback Collection

To collect real world feedback, a user testing survey was conducted involving 13 participants. Each participant was asked to perform 3-5 signs using the system. Afterwards, participants completed a feedback form evaluating usability, accuracy, speed, and overall satisfaction. Questions asked during feedback and their results are listed below:

Demographics:

1. Have you used a sign language recognition system before?

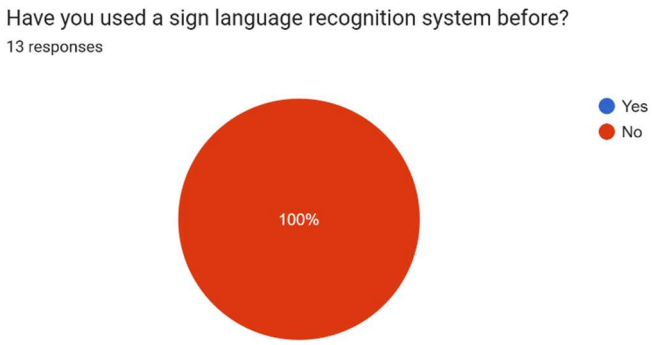


Figure 6.9: Question 1 Pie Chart

2. Are you familiar with Malaysian Sign Language (BIM)?

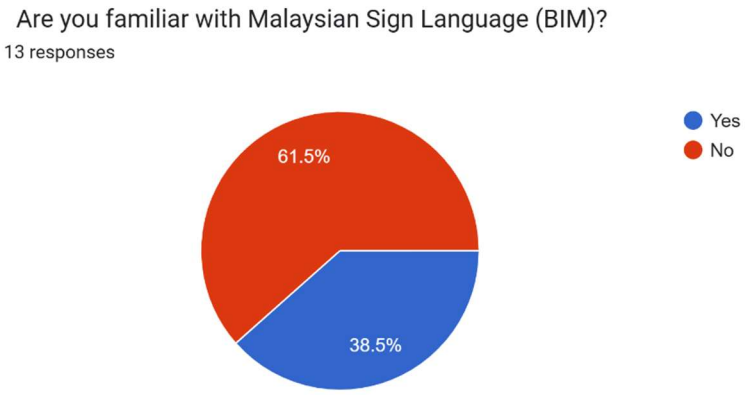


Figure 6.10: Question 2 Pie Chart

Likert Scale (1- Strongly Disagree 5- Strongly Agree):

3. The system was easy to use

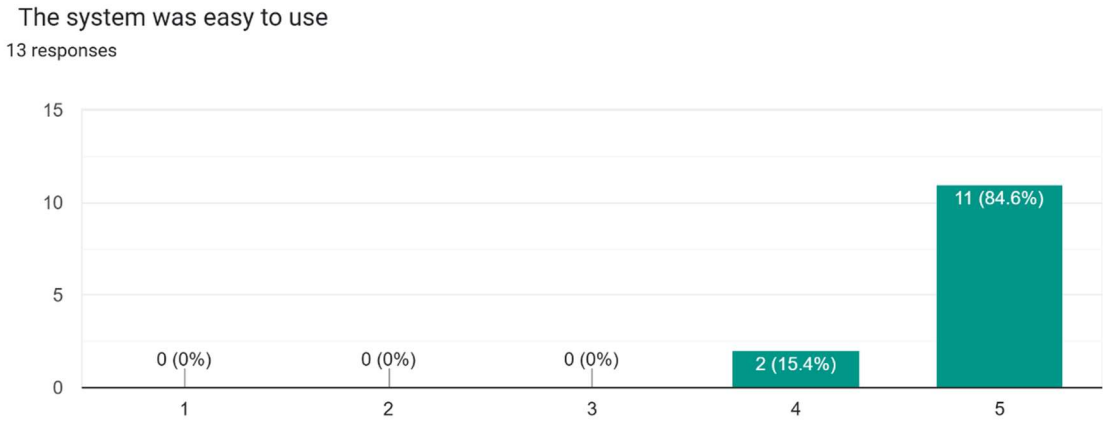


Figure 6.11: Question 3 Likert Scale Chart

4. The prediction was accurate

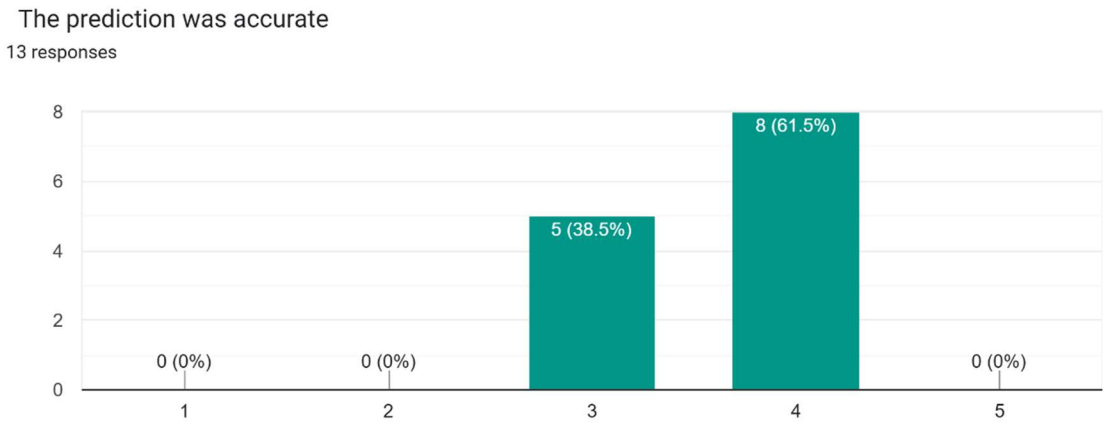


Figure 6.12: Question 4 Likert Scale Chart

5. I felt the system was slow (high delay)

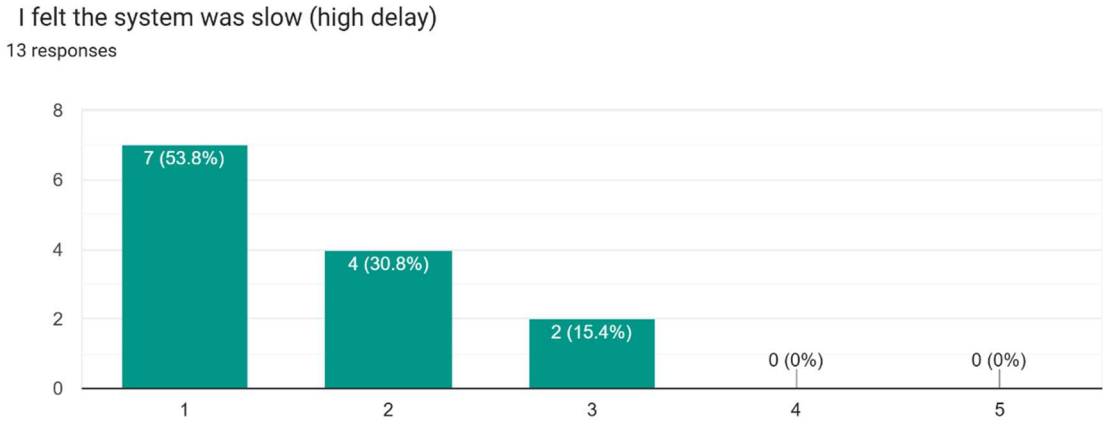


Figure 6.13: Question 5 Likert Scale Chart

6. The interface was clear and understandable

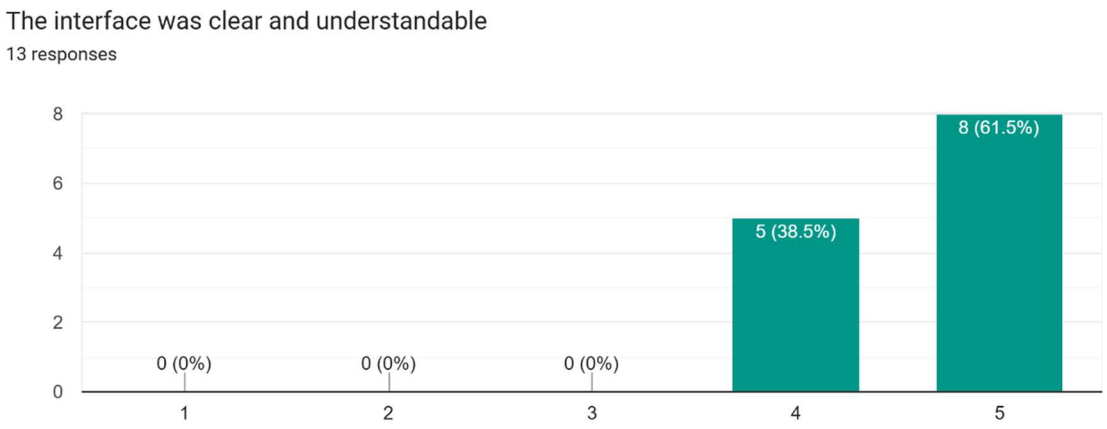


Figure 6.14: Question 6 Likert Scale Chart

7. This system would be helpful for learning or communicating using Malaysian Sign Language (BIM)

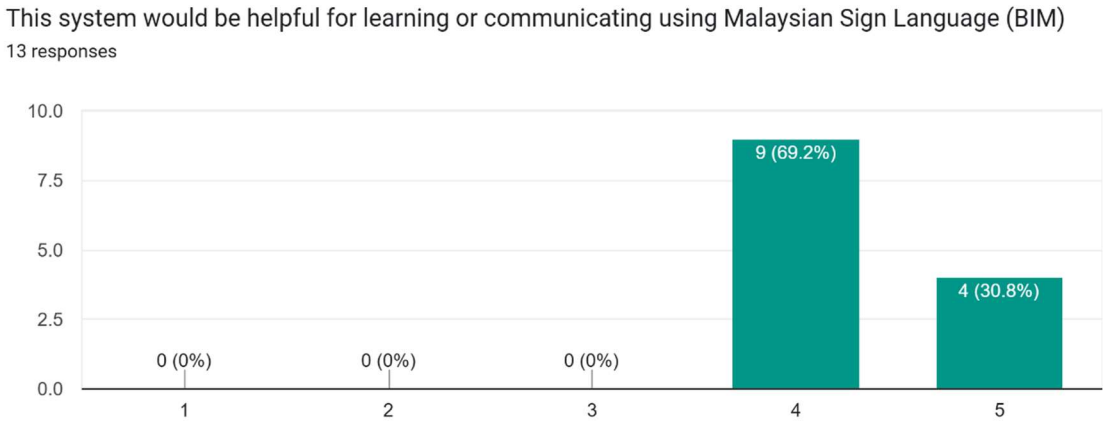


Figure 6.15: Question 7 Likert Scale Chart

6.3.3 Key Findings

The survey results indicate a positive user experience with the BIM recognition system. All respondents agreed that the system would be helpful for learning or communicating using Malaysian Sign Language (BIM), with 100% rating its usefulness as either 4 or 5 out of 5. The interface was also well-received, as every participant found it clear and understandable, and the majority (61.5%) gave it the highest possible rating. Most users did not perceive the system as slow, with over 84% indicating low delay, suggesting that the system operates responsively in real time. In terms of prediction accuracy, all users rated the system as either 3 or 4 out of 5, reflecting generally accurate gesture recognition. Ease of use was another strong point, with 84.6% of respondents finding the system very easy to use. Notably, the survey included both users familiar and unfamiliar with BIM, and none had prior experience with sign language recognition systems, highlighting the system's accessibility and intuitiveness for first-time users. Overall, the feedback demonstrates that the system is user-friendly, effective, and well-suited for both new and experienced users in the context of BIM communication. Table 6.2 below shows the overall summary of user testing.

Table 6.2: User Testing Summary

Aspect	**Key Insight**
Usefulness	100% found it helpful for BIM learning/communication
Interface Clarity	100% found it clear and understandable
System Speed	85% did not experience high delay
Prediction Accuracy	100% rated accuracy as 3 or 4 out of 5
Ease of Use	100% rated it 4 or 5 out of 5
BIM Familiarity	39% familiar, 61% not familiar
Prior SLR System Experience	0% had use such system before

6.4 Summary

In this chapter, the performance of the LSTM, BiLSTM, and GRU models, as well as the ensemble approach, was thoroughly evaluated using key metrics such as accuracy, precision, recall, F1-score, and confusion matrices. The results demonstrated that all models achieved high accuracy in recognizing BIM gestures, with the ensemble method providing the best overall performance with 98.67% accuracy. Visualizations such as confusion matrix heatmaps and training curves further illustrated the strengths and areas for improvement for each model. The evaluation also highlighted the system's ability to generalize well to unseen data and perform reliably in real-time scenarios. Based on these findings, GRU with 97.33% accuracy was selected for deployment in the real-time BIM recognition system, ensuring both accuracy and user satisfaction. Overall, the evaluation confirms that the developed system is effective, user-friendly, and capable of supporting communication using BIM.

Chapter 7: Conclusion

This project successfully developed and evaluated a real-time recognition system for Malaysian Sign Language (BIM) gestures, integrating state-of-the-art computer vision and deep learning techniques. The system utilized MediaPipe Holistic for robust keypoint extraction and explored multiple recurrent neural network architectures such as LSTM, BiLSTM, and GRU, to effectively model the temporal dynamics of sign language gestures. Through systematic data collection, preprocessing, and feature engineering, a high-quality dataset was constructed, enabling the models to achieve strong generalization and high accuracy.

Comprehensive model evaluation demonstrated that all three neural network models performed well, with the ensemble approach yielding the highest accuracy of 98.67%. GRU was selected for deployment in the real-time system, ensuring optimal performance for end users. The integration of OpenCV for video capture and Tkinter for the graphical user interface resulted in a seamless and user-friendly application capable of providing immediate feedback to users.

User survey results further validated the system's effectiveness, highlighting its usefulness, ease of use, and accessibility, even for individuals unfamiliar with BIM or sign language recognition technology. Most users found the system accurate, responsive, and intuitive, underscoring its potential as a valuable tool for learning and communication within the deaf and hard-of-hearing community.

In summary, this project not only bridges the communication gap for BIM users but also demonstrates the potential of combining advanced machine learning models with real-time computer vision for practical, inclusive assistive technologies. Future work may focus on expanding the gesture vocabulary,

improving system speed, and exploring deployment on mobile or embedded platforms to further enhance accessibility and impact.

References

1. Adaloglou, N. M., Chatzis, T., Papastratis, I., Stergioulas, A., Papadopoulos, G. T., Zacharopoulou, V., Xydopoulos, G., Antzakas, K., Papazachariou, D., & Daras, P. none. (2021). A Comprehensive Study on Deep Learning-based Methods for Sign Language Recognition. *IEEE Transactions on Multimedia*, *24*, 1–1. https://doi.org/10.1109/tmm.2021.3070438

2. Al-Qurishi, M., Khalid, T., & Souissi, R. (2021). Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues. *IEEE Access*, *9*, 126917–126951. https://doi.org/10.1109/access.2021.3110912

3. Bhatia, P., & Ankita Wadhawan. (2021). Deep learning-based sign language recognition system for static signs. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-019-04691-y

4. Md, I. (2022). A Framework for Malaysian Sign Language Recognition using Deep Learning Initiatives. *Mathematical Sciences and Informatics Journal*, *3*(2), 65–79. https://doi.org/10.24191/mij.v3i2.19395

5. Galván-Ruiz, J., Travieso-González, C. M., Tejera-Fettmilch, A., Pinan-Roescher, A., Esteban-Hernández, L., & Domínguez-Quintana, L. (2020). Perspective and Evolution of Gesture Recognition for Sign Language: A Review. *Sensors (Basel, Switzerland)*, *20*(12), 3571. https://doi.org/10.3390/s20123571

6. GeeksforGeeks. (2017). *Introduction to Convolution Neural Network*. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/

7. Han Lin, H. Y., & Murli, N. (2022). BIM Sign Language Translator Using Machine Learning (TensorFlow). *Journal of Soft Computing and Data Mining*, *3*(1). https://doi.org/10.30880/jscdm.2022.03.01.007

8. Jain, R., Jain, M., Jain, R., & Madan, S. (2021). Human Computer Interaction – Hand Gesture Recognition. *Advanced Journal of Graduate Research*, *11*(1), 1–9. https://doi.org/10.21467/ajgr.11.1.1-9

9. Khan, R. U., Khattak, H., Wong, W. S., AlSalman, H., Mosleh, M. A. A., & Mizanur Rahman, S. M. (2021). Intelligent Malaysian Sign Language Translation System Using Convolutional-Based Attention Module with Residual Network. *Computational Intelligence and Neuroscience*, *2021*, e9023010. https://doi.org/10.1155/2021/9023010

10. *LAWS OF MALAYSIA ONLINE VERSION OF UPDATED TEXT OF REPRINT Act 685 PERSONS WITH DISABILITIES ACT 2008*. (2014). https://www.un.org/development/desa/disabilities/wp-content/uploads/sites/15/2022/11/Malaysia-Pwd-Act-2008.pdf

11. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A Framework for Building Perception Pipelines. *ArXiv:1906.08172 [Cs]*. https://arxiv.org/abs/1906.08172

12. Madhiarasan, D. M., & Roy, P. P. P. (2022). A Comprehensive Review of Sign Language Recognition: Different Types, Modalities, and Datasets. *ArXiv:2204.03328 [Cs]*. https://arxiv.org/abs/2204.03328

13. MediaPipe. (n.d.). Home. https://chuoling.github.io/mediapipe/

14. Papastratis, I., Chatzikonstantinou, C., Konstantinidis, D., Dimitropoulos, K., & Daras, P. (2021). Artificial Intelligence Technologies for Sign Language. *Sensors (Basel, Switzerland)*, *21*(17), 5843. https://doi.org/10.3390/s21175843

15. Rastgoo, R., Kiani, K., & Escalera, S. (2021). Sign Language Recognition: A Deep Survey. *Expert Systems with Applications*, *164*, 113794. https://doi.org/10.1016/j.eswa.2020.113794

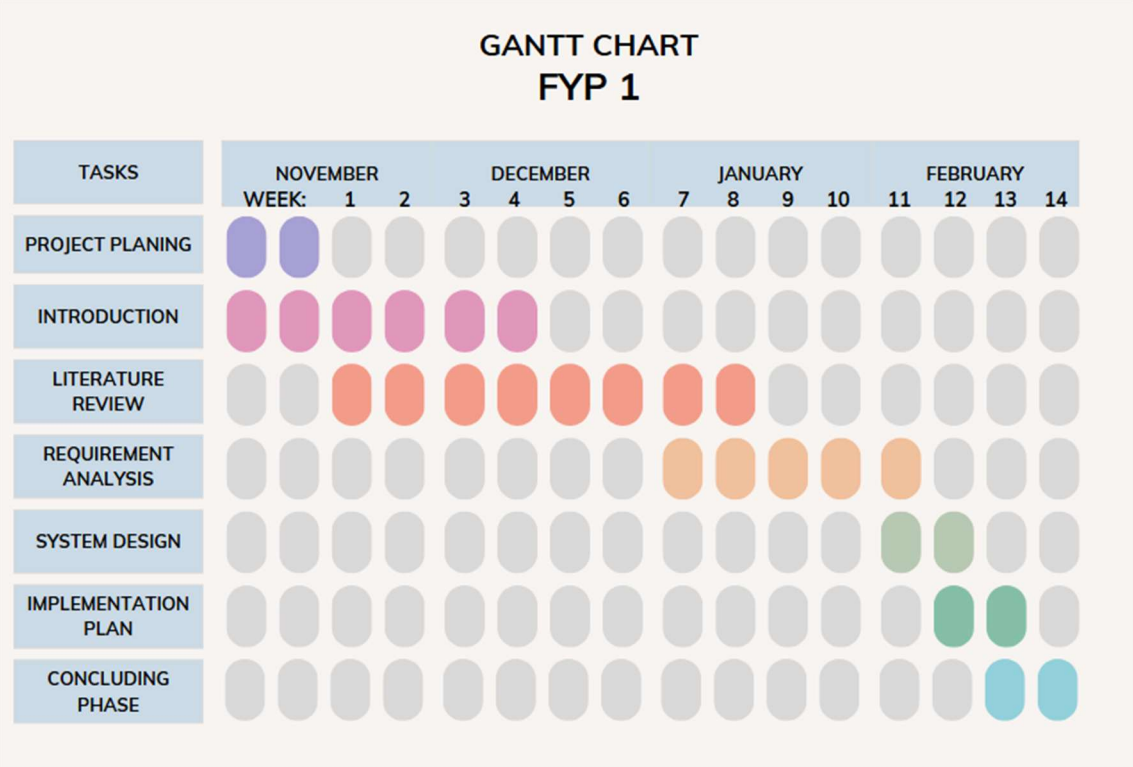
16. Samaan, G. H., Wadie, A. R., Attia, A. K., Asaad, A. M., Kamel, A. E., Slim, S. O., Abdallah, M. S., & Cho, Y.-I. (2022). MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition. *Electronics*, *11*(19), 3228. https://doi.org/10.3390/electronics11193228

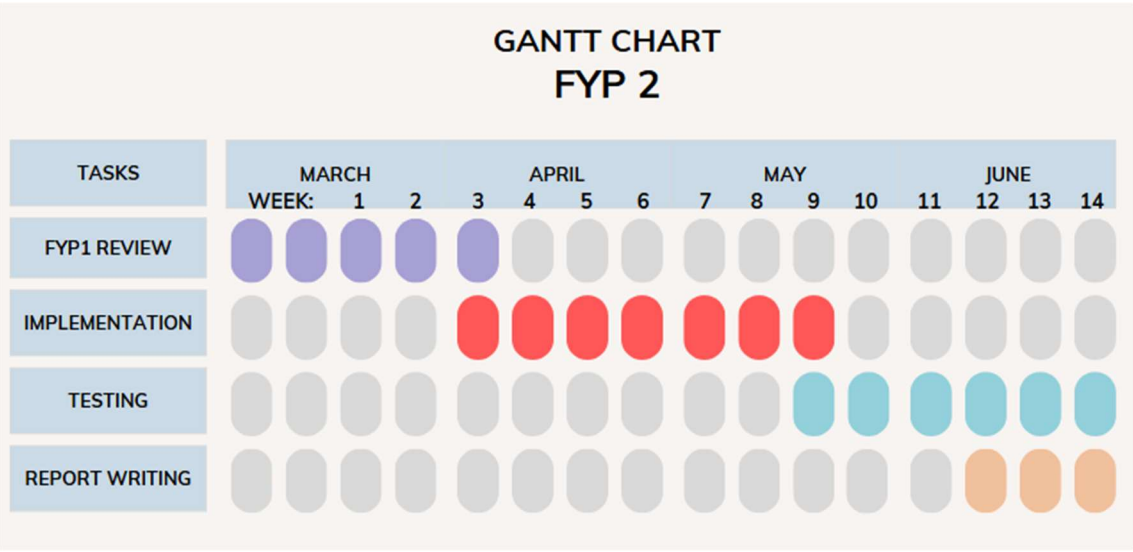
17. Srivastava, S., Gangwar, A., Mishra, R., & Singh, S. (2022). Sign Language Recognition System Using TensorFlow Object Detection API. *Communications in Computer and Information Science*, 634–646. https://doi.org/10.1007/978-3-030-96040-7_48

18. Vaishnavi Karanjkar, Rutuja Bagul, Singh, R. R., & Rushali Shirke. (2023). A Survey of Sign Language Recognition. *INTERANTIONAL JOURNAL of SCIENTIFIC RESEARCH in ENGINEERING and MANAGEMENT*, *07*(10), 1–11. https://doi.org/10.55041/ijsrem26316

19. Wadhawan, A., & Kumar, P. (2019). Sign Language Recognition Systems: A Decade Systematic Literature Review. *Archives of Computational Methods in Engineering*. https://doi.org/10.1007/s11831-019-09384-2

Appendix A: Gantt Charts





Appendix B: FYP1 & FYP2 Meeting Logs

MMU MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 20/11/2024	**Meeting No.: 1**
Meeting Mode: ~~Online~~ / Face-to-Face	
Project ID: FYP01-DS-T2430-0073	**Project Type:** ~~Research based/Application based/~~ Application & Research based
Project Title: Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID: 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- I have prepared my project proposal and submitted it to my supervisor Dr. Nasser
- I have done some background study before the first meeting

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning~~ / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Start literature review based on my project title
- Aim to read about 15-20 papers
- Summarize key findings
- Create a comparative table noting features included or excluded.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Need to narrow down project as it is too broad
- Think about evaluation approach

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- The student needs to commence the literature review to establish a solid research foundation.
- The project needs to be narrowed down to ensure a more focused and manageable research approach.
- The student should develop a clear evaluation approach.

Supervisor's Signature

Student's Signature

Co-Supervisor's Signature
(if applicable)

Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 4/12/2024	**Meeting No.: 2**
Meeting Mode: ~~Online~~ / Face-to-Face	
Project ID: FYP01-DS-T2430-0073	**Project Type:** ~~Research based/Application based/~~ Application & Research based
Project Title: Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID: 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Read around 10 research papers
- Drafted Chapter 1 (introduction) and Chapter 2 (literature review) portion of the interim report
- Created a gantt chart

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning~~ / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Continue working on Chapter 2
- Include current work/applications on sign language recognition

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Need to improve progress on chapter 2

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- The student needs to continue working on Chapter 2, include current work and applications on sign language recognition.
- The student needs to improve progress on Chapter 2 to maintain alignment with the project timeline.
- Student may start working on Chapter 3.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 18/12/2024	**Meeting No.: 3**
Meeting Mode: ~~Online~~ / Face-to-Face	
Project ID: FYP01-DS-T2430-0073	**Project Type:** ~~Research based/Application based/~~ Application & Research based
Project Title: Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID: 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / ~~Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- I have finished my first draft of Chapter 1 (Introduction) and Chapter 2 (Literature Review) of the interim report and emailed it to my supervisor Dr Nasser

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework~~/ Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Enhance the problem statement in Chapter 1 by adding more references
- Elaborate more on the methodology section in Chapter 1
- Include additional subtopics for Chapter 2 to broaden the background study
- Start Chapter 3 (Requirement Analysis)

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- The report needs to be extended to include more detailed content and meet the required length.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- Clearly define the research gap identified in existing applications to justify your study.
- Clearly define both functional and non-functional requirements for the project.
-

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 8/1/2024	**Meeting No.: 4**
Meeting Mode: ~~Online~~ / Face-to-Face	
Project ID: FYP01-DS-T2430-0073	**Project Type:** ~~Research-based/Application-based/~~ Application & Research based
Project Title: Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID: 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework~~ / Design or Research Methodology / Prototype Development or Proof-of-Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- I have done a draft for the introduction, existing system analysis, proposed system overview, feasibility analysis, requirements gathering of chapter 3
- I have reworked my chapter 1 and chapter 2

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review /~~ Requirement Analysis or Theoretical Framework~~ / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Provide more details on LSTM
- Refine and improve Chapter 3.
- Change from point form to academic writing.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- The report should have more references and around 60 pages

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- The student should enhance report with more figures, diagrams and equations
- Have more detailed diagrams
- The student should change writing style to academic writing

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 16/1/2024	**Meeting No.: 5**
Meeting Mode: Online / ~~Face-to-Face~~	
Project ID: FYP01-DS-T2430-0073	**Project Type:** ~~Research-based/Application-based/~~ Application & Research based
Project Title: Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID: 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review /~~ Requirement Analysis or Theoretical Framework ~~/ Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finished draft of Chapter 3
- Changed writing style

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework /~~ Design or Research Methodology ~~/ Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Start chapter 4
- Add descriptions to diagrams/figures

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Switch use case and data flow diagram from Chapter 3 to Chapter 4

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- The student should relate diagrams and figures to the content
- The student should elaborate more on machine learning model

Supervisor's Signature

Student's Signature

Co-Supervisor's Signature
(if applicable)

Company Supervisor's Signature
(if applicable)

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 22/1/2024	**Meeting No.: 6**
Meeting Mode: ~~Online~~ / Face-to-Face	
Project ID: FYP01-DS-T2430-0073	**Project Type:** ~~Research based/Application based/~~ Application & Research based
Project Title: Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID: 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: ~~Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework~~ / Design or Research Methodology / ~~Prototype Development or Proof of Concept / Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finished draft of Chapter 1, 2, and 3 for my supervisors review
- Started on Chapter 4

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks~~: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework /~~ Design or Research Methodology / ~~Prototype Development or Proof of Concept/ Draft Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finish Chapter 4
- Finalize Chapter 1, 2, and 3

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Several subtopics were redundant or can be placed in other chapters

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- The student should finalize their report for submission
- The student should change their from point form to paragraph form in certain subtopics
- The student should add more visuals to enhance their report

.....
Supervisor's Signature

.....
Student's Signature

.....	.....
Co-Supervisor's Signature **(if applicable)**	Company Supervisor's Signature **(if applicable)**

IMPORTANT NOTES TO STUDENTS:

1. Items 1 – 3 are to be completed by the students prior to the meeting. Item 4 is to be completed by the supervisor / co-supervisor / company supervisor.

2. Student has to upload the soft copies of the meeting logs in eBwise and also attach them along with interim (FYP1) report.
   Minimum requirement is SIX Meeting Logs (Period: Week 3 to Week 12). Students can have fortnightly meetings with the supervisor.

3. Log sheets provide the basis for evaluating the General Effort (Project Management, Attitude, and Technical Competency) of the student, by the supervisor and also for checking the attendance requirement of the student, by the FYP Committee.

   This also provide the student with feedback from the supervisor / co-supervisor / company supervisor on the tasks done and provide the plan for the upcoming tasks. This can provide the motivation for the student to give consistent and efficient effort throughout the period of FYP.

4. Student who fails to meet the minimum requirement (six nos.) of log sheets will not be allowed to submit FYP report.

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 18/4/2025	**Meeting No.:** 1
Meeting Mode: **(~~In-person~~ / Online)**	
Project ID: FYP02-DS-T2510-0066	**Project Type:** **~~Research-based / Application-based~~ / Hybrid**
Project Title : Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID : 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Completed FYP 1 review feedback and made necessary adjustments.
- Started collecting the dataset for the system.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or ~~Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Begin system implementation based on the finalized design and requirements.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- The initial LSTM model was too simple and lacked sufficient novelty or contribution for the project. Decided to develop a hybrid model combining LSTM with CNN to increase complexity and improve contribution value.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- Student should begin working on system implementation as per the project plan.
- Student should also start planning for user testing and gather feedback to evaluate system performance and usability.

.....
Supervisor's Signature

.....
Student's Signature

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 7/5/2025	**Meeting No.:** 2
Meeting Mode: **(In-person / ~~Online~~)**	
Project ID: FYP02-DS-T2510-0066	**Project Type:** **~~Research-based / Application-based~~ / Hybrid**
Project Title : Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID : 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Implemented the core system using Python (excluding API/GUI components).
- Collected and prepared the dataset for system development and testing.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or ~~Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Expand the dataset.
- Begin implementing the API and GUI components of the system.
- Start drafting Chapter 5 of the final report.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- The accuracy of the system is not at its best, possibly due to a limited dataset size. Plan to increase the dataset size to enhance the system's accuracy and ensure more reliable performance.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- Student should start Chapter 5.
- Student should finalize implementation and proceed with user testing and feedback.
- Student should increase dataset size to support broader and effective implementation.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 16/5/2025	**Meeting No.:** 3
Meeting Mode: **(In-person / ~~Online~~)**	
Project ID: FYP02-DS-T2510-0066	**Project Type:** **~~Research-based / Application-based~~ / Hybrid**
Project Title : Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID : 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Completed system implementation using Python, including all core functionalities.
- Drafted Chapter 5 of the final report.

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or ~~Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Conduct user testing to evaluate the system's functionality and gather feedback.
- Finalize Chapter 5 of the final report, including results analysis, pseudocode, charts, and discussion of findings.

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Initially included screenshots of the code in the algorithm section in Chapter 5, which was not suitable. Should convert these into structured pseudocode for better clarity.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- Student should finalize Chapter 5 for final report
- Student should diversify the content of Chapter 5 by including pseudocode, charts and output images to enhance clarity and better illustrate the implementation and analysis.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature

.....
Company Supervisor's Signature

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 30/5/2025	**Meeting No.:** 4
Meeting Mode: **(~~In-person~~ / Online)**	
Project ID: FYP02-DS-T2510-0066	**Project Type:** **~~Researeh-based / Application-based~~ / Hybrid**
Project Title : Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID : 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report ~~/ Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize Chapter 5 of the final report, including results analysis, pseudocode, charts, and discussion of findings.
- Created survey questionnaire for user testing feedback

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or ~~Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report~~ / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Start Chapter 6 of the final report
- Collect user testing feedback on the completed system

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- Student should start drafting Chapter 6 of the final report
- Student should start feedback collection for the implemented system for Chapter 6

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 16/6/2025	**Meeting No.:** 5
Meeting Mode: **(~~In-person~~ / Online)**	
Project ID: FYP02-DS-T2510-0066	**Project Type:** **~~Research-based / Application-based~~ / Hybrid**
Project Title : Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID : 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: ~~Implementation~~ / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report ~~/ Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Drafted chapter 6 of the final report

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or ~~Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report~~ / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize Chapter 6 of the final report
- Finalize final report for completion

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- For chapter 6, ensure all components of the system are properly evaluated, especially the main contribution.
- For user-based evaluation, include survey questions and collected responses.
- Perform in-depth analysis and discussion once results are gathered
- Clearly explain how results show the effectiveness of the system.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 20/6/2025	**Meeting No.:** 6
Meeting Mode: **(In-person ~~/ Online~~)**	
Project ID: FYP02-DS-T2510-0066	**Project Type:** **~~Researeh-based / Application-based~~ / Hybrid**
Project Title : Malaysian Sign Language (BIM) Recognition Using Machine Learning	
Student ID : 1211101888	**Student Name:** Shahnaz Binti Husain Sukri
Student Programme and Specialisation: BCS. Data Science	
Supervisor Name: Dr. Alandoli Mohammed Nasser Mohammed	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / ~~Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects)~~ / Draft Final Report ~~/ Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Redrafted Chapter 6 of the final report
- Finalize other Chapters of the final report

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or ~~Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) /~~ Draft Final Report~~ / Final Report Completion~~

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Finalize final report for submission
- Create poster for poster presentation and submission

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

- Student should start finalizing final report for submission
- The student should ensure correct and consistent formatting is used throughout the report especially table/figure titles.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

Appendix C: Turnitin Similarity Index Page

FYP2

ORIGINALITY REPORT

17%	9%	10%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Multimedia University Student Paper	4%
2	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P. Prasad. "Algorithms in Advanced Artificial Intelligence - Proceedings of International Conference on Algorithms in Advanced Artificial Intelligence (ICAAAI-2024)", CRC Press, 2025 Publication	1%
3	www.mdpi.com Internet Source	1%
4	Submitted to University of Westminster Student Paper	1%
5	Submitted to Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur Student Paper	1%
6	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelligent Computing and Communication Techniques - Volume 3", CRC Press, 2025 Publication	<1%
7	Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical and Computer Technologies", CRC Press, 2025 Publication	<1%

8	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challenges in Information, Communication and Computing Technology", CRC Press, 2024 Publication	<1 %
9	dokumen.pub Internet Source	<1 %
10	www.journalpressindia.com Internet Source	<1 %
11	Submitted to Kingston University Student Paper	<1 %
12	Submitted to Southern University College Student Paper	<1 %
13	Submitted to The Maldives National University Student Paper	<1 %
14	Submitted to University of Ulster Student Paper	<1 %
15	Submitted to Babes-Bolyai University Student Paper	<1 %
16	Submitted to University of Sheffield Student Paper	<1 %
17	pastebin.com Internet Source	<1 %
18	Amol Dattatray Dhaygude, Suman Kumar Swarnkar, Priya Chugh, Yogesh Kumar Rathore. "Smart Agriculture - Harnessing Machine Learning for Crop Management", CRC Press, 2024 Publication	<1 %
19	web.archive.org Internet Source	<1 %

20	Ankita Wadhawan, Parteek Kumar. "Sign Language Recognition Systems: A Decade Systematic Literature Review", Archives of Computational Methods in Engineering, 2019 Publication	<1 %
21	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
22	Submitted to University of Hertfordshire Student Paper	<1 %
23	jisem-journal.com Internet Source	<1 %
24	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelligent Computing and Communication Techniques - Volume 2", CRC Press, 2025 Publication	<1 %
25	Benjamin Svendsen, Seifedine Kadry. "Comparative Analysis of Image Classification Models for Norwegian Sign Language Recognition", Technologies, 2023 Publication	<1 %
26	Submitted to University of Bradford Student Paper	<1 %
27	listens.online Internet Source	<1 %
28	hdprediction.commons.gc.cuny.edu Internet Source	<1 %
29	www.ncbi.nlm.nih.gov Internet Source	<1 %
30	Submitted to The University of Texas at Arlington Student Paper	<1 %

31	Submitted to University of South Wales - Pontypridd and Cardiff Student Paper	<1%
32	ijirt.org Internet Source	<1%
33	Pallavi Kaushik, Anmol Gupta, Partha Pratim Roy, Debi Prosad Dogra. "EEG-Based Age and Gender Prediction Using Deep BLSTM-LSTM Network Model", IEEE Sensors Journal, 2019 Publication	<1%
34	Ezekiel Maina, Lilian Wanzare, James Obuhuma, Mildred Ayere, Maurine Kang'ahi, Joel Okutoyi. "Kenyan sign language word-based pose dataset", Data in Brief, 2025 Publication	<1%
35	Submitted to University of Carthage Student Paper	<1%
36	Tasmiya salim Mujawar. "Urban Noise Classification Using Machine Learning Techniques: Comparative Analysis and Future", Open Science Framework, 2025 Publication	<1%
37	medikom.iocspublisher.org Internet Source	<1%
38	repositorio.ufsm.br Internet Source	<1%
39	Submitted to University of Glasgow Student Paper	<1%
40	dspace.bracu.ac.bd Internet Source	<1%
41	www.ksqm.org Internet Source	<1%
	www.uyik.org	

42	Internet Source	<1%
43	Waleed Umer, Imran Mehmood, Yazan Qarout, Maxwell Fordjour Antwi-Afari, Shahnawaz Anwer. "Deep learning-based fatigue monitoring of construction workers using physiological signals", Automation in Construction, 2025 Publication	<1%
44	odr.chalmers.se Internet Source	<1%
45	Submitted to Brunel University Student Paper	<1%
46	doaj.org Internet Source	<1%
47	harbinengineeringjournal.com Internet Source	<1%
48	Submitted to International Islamic University Malaysia Student Paper	<1%
49	ijsrem.com Internet Source	<1%
50	kar.kent.ac.uk Internet Source	<1%
51	rjwave.org Internet Source	<1%
52	www.nature.com Internet Source	<1%
53	Submitted to Al Akhawayn University in Ifrane Student Paper	<1%
54	Purva Chaitanya Badhe, Vaishali Kulkarni. "Artificial Neural Network based Indian Sign	<1%

	Language Recognition using hand crafted features", 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020 Publication	
55	repositorio.ufla.br Internet Source	<1 %
56	sparkbyexamples.com Internet Source	<1 %
57	Submitted to University of Wolverhampton Student Paper	<1 %
58	de Sousa, Lúcia Maria Bessa. "Detecting a Poker Face", Universidade de Aveiro (Portugal), 2024 Publication	<1 %
59	Submitted to University of Greenwich Student Paper	<1 %
60	Submitted to CSU, San Jose State University Student Paper	<1 %
61	Syed Rameem Zahra, Mohammad Ahsan Chishti. "Security and Privacy in the Internet of Things", CRC Press, 2020 Publication	<1 %
62	Submitted to Turun yliopisto Student Paper	<1 %
63	Submitted to University of Central Lancashire Student Paper	<1 %
64	pangea.stanford.edu Internet Source	<1 %
65	www.atbuftejoste.net Internet Source	<1 %
66	www.atlantis-press.com Internet Source	<1 %

67	repositorio.iscte-iul.pt Internet Source	<1 %
68	Asmaa Alayed. "Machine Learning and Deep Learning Approaches for Arabic Sign Language Recognition: A Decade Systematic Literature Review", Sensors, 2024 Publication	<1 %
69	Submitted to Middlesex University Student Paper	<1 %
70	docplayer.net Internet Source	<1 %
71	fastercapital.com Internet Source	<1 %
72	nurseslabs.com Internet Source	<1 %
73	utpedia.utp.edu.my Internet Source	<1 %
74	www.diva-portal.org Internet Source	<1 %
75	Almahri, Hamda Ghalib Awadh Ali. "Arabic Sign Language Recognition: A Deep Learning Approach", The British University in Dubai, 2023 Publication	<1 %
76	Durgesh Kumar Mishra, Nilanjan Dey, Bharat Singh Deora, Amit Joshi. "ICT for Competitive Strategies", CRC Press, 2020 Publication	<1 %
77	arxiv.org Internet Source	<1 %
78	forum.fxsound.com Internet Source	<1 %

79	ijrpr.com Internet Source	<1%
80	propulsiontechjournal.com Internet Source	<1%
81	scholar.sun.ac.za Internet Source	<1%
82	www.nice.org.uk Internet Source	<1%
83	www.ukessays.com Internet Source	<1%
84	zir.nsk.hr Internet Source	<1%
85	Anurag Tiwari, Manuj Darbari. "Emerging Trends in Computer Science and Its Application - Proceedings of the International Conference on Advances in Emerging Trends in Computer Applications (ICAETC-2023) December 21–22, 2023, Lucknow, India", CRC Press, 2025 Publication	<1%
86	C++ Recipes, 2015. Publication	<1%
87	Mohsen Mohammadagha. "Hyperparameter Optimization Strategies for Tree-Based Machine Learning Models Prediction: A Comparative Study of AdaBoost, Decision Trees, and Random Forest", Open Science Framework, 2025 Publication	<1%
88	Qing Chen, A. El-Sawah, C. Joslin, N.D. Georganas. "A dynamic gesture interface for virtual environments based on hidden markov models", IREE International Worksho	<1%

	on Haptic Audio Visual Environments and their Applications, 2005., 2005 Publication	
89	Ronghao Hou, Dongjie Liu, Xiaobo Jin, Jian Weng, Guanggang Geng. "A malware detection method with function parameters encoding and function dependency modeling", PeerJ Computer Science, 2025 Publication	<1 %
90	Sujith Samuel Mathew, Mohammad Amin Kuhail, Maha Hadid, Shahbano Farooq. "The Object-Oriented Approach to Problem Solving and Machine Learning with Python", CRC Press, 2025 Publication	<1 %
91	Vinhas, Manuel Francisco Paulo Sarreira. "Análise de VOCs do ar Exalado Suportada por IA Para o Diagnóstico Precoce de Cancro", Universidade NOVA de Lisboa (Portugal) Publication	<1 %
92	discovery.researcher.life Internet Source	<1 %
93	dos Santos, Sérgio Filipe Paiva da Silva Gonçalves. "Land Cover Automatic Classification Using Deep Learning Techniques Applied to Satellite Imagery", ISCTE - Instituto Universitario de Lisboa (Portugal), 2024 Publication	<1 %
94	etd.lib.metu.edu.tr Internet Source	<1 %
95	ijaseit.insightsociety.org Internet Source	<1 %
96	journalofbigdata.springeropen.com Internet Source	

		<1%
97	nhsjs.com Internet Source	<1%
98	pdffox.com Internet Source	<1%
99	pmc.ncbi.nlm.nih.gov Internet Source	<1%
100	research-repository.st-andrews.ac.uk Internet Source	<1%
101	www.adaface.com Internet Source	<1%
102	www.frontiersin.org Internet Source	<1%
103	www.slideshare.net Internet Source	<1%
104	www.theseus.fi Internet Source	<1%
105	Md Zia Uddin. "Machine Learning and Python for Human Behavior, Emotion, and Health Status Analysis", CRC Press, 2024 Publication	<1%
106	Achin Jain, Arun Kumar Dubey, Meenakshi Gupta, Sarita Yadav, Arvind Panwar, Roger Atanga, Bernardo Lemos, Saurav Mallik. "Brain Stroke Prediction: A PSO Optimized Stacked Ensemble Machine Learning Approach with SMOTEEN Data Balancing", Cold Spring Harbor Laboratory, 2025 Publication	<1%

Appendix D: Prototype Code

sign_language_gui.py

```python
import tkinter as tk
from tkinter import ttk, font
import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
import os
import time
import threading

# Load the model
model = tf.keras.models.load_model('model3.h5')

# Setup MediaPipe
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils

# Define the actions
DATA_PATH = os.path.join('Action_Dataset')
actions = np.array([folder for folder in os.listdir(DATA_PATH) if os.path.isdir(os.path.join(DATA_PATH, folder))])


# MediaPipe detection function
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results

# Drawing landmarks function
def draw_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1))

    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1))

    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1))

    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1))
```

```python
# Extract keypoints function
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros(33*4)
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if results.face_landmarks else np.zeros(468*3)
    left_hand = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() if results.left_hand_landmarks else np.zeros(21*3)
    right_hand = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() if results.right_hand_landmarks else np.zeros(21*3)
    return np.concatenate([pose, face, left_hand, right_hand])

# Probability visualization function
def prob_viz(res, actions, input_frame):
    output_frame = input_frame.copy()
    top_idx = np.argmax(res)
    prob = res[top_idx]
    color = (245, 117, 16)  # Single color for all classes
    cv2.rectangle(output_frame, (0, 60), (int(prob * 100), 90), color, -1)
    cv2.putText(output_frame, actions[top_idx], (0, 85), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)
    return output_frame

# Function to run the sign language detection
def detection_thread():
    global running

    # Set capture device
    cap = cv2.VideoCapture(0)

    # Detection variables
    sequence = []     # collect frames to generate prediction
    sentence = []     # prediction history
    predictions = []
    threshold = 0.96  # confidence threshold

    # Start detection loop
    with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
        while running and cap.isOpened():
            # Read feed
            ret, frame = cap.read()

            # Make detection
            image, results = mediapipe_detection(frame, holistic)
            print(results)

            # Draw landmarks
            draw_landmarks(image, results)

            #make prediction
            keypoints = extract_keypoints(results)
            sequence.append(keypoints)
            sequence = sequence[-30:]

            if len(sequence) == 30:
                res = model.predict(np.expand_dims(sequence, axis=0))[0]
                print(actions[np.argmax(res)])
                predictions.append(np.argmax(res))
```

```
                if np.unique(predictions[-10:])[0] == np.argmax(res):
                    if res[np.argmax(res)] > threshold:
                        if len(sentence) > 0:
                            if actions[np.argmax(res)] != sentence[-1]:    #check if not same action
                                sentence.append(actions[np.argmax(res)])
                        else:
                            sentence.append(actions[np.argmax(res)])

                if len(sentence) > 5:
                    sentence = sentence[-5:]

                #visualization probability
                image = prob_viz(res, actions, image)

            #rendering text
            cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
            cv2.putText(image, ' '.join(sentence), (3,30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)

            # Show to screen
            cv2.imshow('BIM Detection', image)

            # Break loop
            if cv2.waitKey(10) & 0xFF == ord('q'):
                running = False
                break

    # Release resources
    cap.release()
    cv2.destroyAllWindows()

    # Update UI
    root.after(0, update_ui_after_stop)

# Function to start detection in a separate thread
def run_detection():
    global running, detection_thread_obj

    if running:
        return

    # Update UI
    start_button.config(state="disabled")
    status_label.config(text="Running... (Press 'Q' to stop)")

    # Start detection in a separate thread
    running = True
    detection_thread_obj = threading.Thread(target=detection_thread)
    detection_thread_obj.daemon = True
    detection_thread_obj.start()
```

```python
# Function to stop detection
def stop_detection():
    global running
    running = False
    status_label.config(text="Stopping...")

# Function to update UI after stopping
def update_ui_after_stop():
    start_button.config(state="normal")
    status_label.config(text="Ready")

# Function to handle key press events
def on_key_press(event):
    # Check if Q or q was pressed
    if event.char.lower() == 'q' and running:
        stop_detection()

# Create the main application window
root = tk.Tk()
root.title("BIM Recognition System")
root.geometry("450x300")
root.configure(bg="#f0f0f0")

# Bind key press event to the root window
root.bind('<Key>', on_key_press)

# Set global style
style = ttk.Style()
style.configure("TButton", font=("Arial", 14), padding=10)

# Create header
header_font = font.Font(family="Arial", size=18, weight="bold")
header = tk.Label(root, text="BIM Recognition System", font=header_font, bg="#f0f0f0")
header.pack(pady=20)

# Create instruction text
instructions = tk.Label(
    root,
    text="Press 'Start' to begin sign language detection.\nPress the 'Q' key to stop detection.\nPress 'Quit' to exit the application.",
    font=("Arial", 12),
    bg="#f0f0f0",
    justify="center"
)
instructions.pack(pady=10)

# Create button frame
button_frame = tk.Frame(root, bg="#f0f0f0")
button_frame.pack(pady=20)

# Create Start button
start_button = ttk.Button(button_frame, text="Start", command=run_detection)
start_button.pack(side="left", padx=10)
```

```python
# Create Quit button
quit_button = ttk.Button(button_frame, text="Quit", command=root.destroy)
quit_button.pack(side="left", padx=10)

# Create status label
status_label = tk.Label(root, text="Ready", font=("Arial", 10), bg="#f0f0f0")
status_label.pack(pady=10)

# Initialize global variables
running = False
detection_thread_obj = None

if __name__ == "__main__":
    root.mainloop()
```