



Faculty of Computer Science and Information Technology

**REAL-TIME MALAYSIAN SIGN LANGUAGE SENTENCE-LEVEL GESTURE
RECOGNITION ON ANDROID USING GRU-CTC MODEL**

TIANG YU EN

Bachelor of Computer Science with Honours (Network Computing)

2025

**REAL-TIME MALAYSIAN SIGN LANGUAGE SENTENCE-LEVEL GESTURE
RECOGNITION ON ANDROID USING GRU-CTC MODEL**

TIANG YU EN

This project is submitted in partial fulfilment of
the requirements for the degree of Bachelor of Computer Science with Honours
(Network Computing)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

2025

Acknowledgement

First and foremost, I would like to thank God for giving me the strength, knowledge, ability, and opportunity to complete this project. Throughout this journey, I have been fortunate to have my dearest supervisor, Ts. Dr. Lim Phei Chin, whose passion for BIM recognition has inspired me to pursue this project. Her exceptional guidance and support, including providing opportunities to learn from external parties and workshops, have been invaluable.

A very special thanks to the Malaysian Federation of the Deaf, especially Ms. Goh Soo Leng, for her exceptional support in providing us with the data. Without this data, the project would not have been a success.

I would also like to express my gratitude to all the staff and lecturers of the Faculty of Computer Science and Information Systems (FCSIT) for providing me with the opportunity to learn and expand my knowledge, preparing me to be future-ready.

Lastly, I am deeply thankful to my family, friends, and course mates for their unwavering support throughout my studies. Thank you.

Table of Contents

Acknowledgement.....	4
Abstract	7
Abstrak	8
List of Tables	9
List of Figure	10
List of Abbreviations	12
List of Equations	13
Chapter 1 Introduction	14
1.1 Background.....	14
1.2 Problem Statement	17
1.3 Scope	18
1.4 Aims and Objectives	18
1.5 Significance of Project.....	19
1.6 Expected Outcome.....	19
1.7 Summary	19
Chapter 2 Literature Review.....	20
2.1 Introduction	20
2.2 Current Scenario Analysis	20
2.3 Review of Related-works	22
2.3.1 Alphabet-Level and Word-Level Language Recognition System.....	22
2.3.2 Sentence-Level Language Recognition System.....	25
2.4 Review of Related Application/ System	28
2.4.1 Sign Language AI	28
2.4.2 Sign Language Translator AI	30
2.4.3 Vivo Sign Language Interpreter (手语翻译官).....	32
2.5 Analysis and Discussion	34
2.6 Summary	37
Chapter 3 Methodology.....	38
3.0 Introduction	38
3.1 Proposed Architecture	38
3.2 Dataset Creation	40
3.3 Data Preprocessing and Feature Extraction.....	42
3.4 Model Selection and Justification.....	42

3.5 Prototype Implementation.....	45
3.6 Evaluation Metric and Validation Technique	46
3.7 Summary	48
Chapter 4 Experimental Results and Discussion	49
4.1 Introduction	49
4.2 Dataset Description and Preparation	49
4.2.1 Video Collection.....	49
4.2.2 Video Folder	50
4.2.3 Video Preprocessing	51
4.2.4 Extraction of Frames.....	53
4.2.5 Data Augmentation	54
4.3 Algorithm Implementation.....	57
4.3.1 Tool used	57
4.3.2 Prototype Implementation.....	64
4.4 Training and Testing Procedures	67
4.4.1 Gated-Recurrent Unit + Connectionist Temporal Classification Model.....	67
4.4.2 Input Features	67
4.4.3 GRU-based Architecture.....	68
4.4.4 Output and Loss.....	68
4.4.5 Decoding Strategy	68
4.4.6 Model Validation and Testing	69
4.5 Interpretation of Experimental Results	70
4.5.1 Parameter Configuration Testing Result	70
4.5.2 Model Validation and Testing	72
4.5.3 Comparative Performance Analysis	75
4.6 Summary	79
Chapter 5. Conclusions and Future Work	80
5.1 Limitations	80
5.2 Future Work	80
5.3 Conclusions	81
References.....	82

Abstract

The unique structure of sign language sentences, relying on transitions between gestures rather than isolated words, renders traditional word-based or fingerspelling methods inefficient for natural communication. Despite the growing need for accessibility tools, a significant gap exists in real-time recognition systems for Malaysian Sign Language (BIM), particularly at the sentence level, and in the availability of portable mobile translator applications. This paper presents a vision-based gesture recognition system for real-time, sentence-level BIM translation using a Gated Recurrent Unit (GRU) combined with Connectionist Temporal Classification (CTC) to handle temporal dependencies and variable-length sequences. The model was evaluated using 5-fold cross-validation, achieving an average WER of 0.0275 with a standard deviation of 0.0173, and a final test WER of 0.0132 after full-dataset training, confirming its strong generalization capability. Integrated into an Android mobile application, the system was tested in real-time on unseen users, achieving a 69% accuracy and average WER at 0.30, highlighting its potential for practical deployment while also revealing challenges related to inter-user variation and gesture similarity. This work contributes toward bridging the accessibility gap for the Malaysian Deaf community through a portable, vision-based real-time translation solution.

Abstrak

Struktur unik ayat dalam bahasa isyarat, yang bergantung pada peralihan antara gerakan dan bukannya perkataan yang diasingkan, menjadikan kaedah berasaskan perkataan atau ejaan jari tradisional kurang berkesan untuk komunikasi yang semula jadi. Walaupun keperluan terhadap alat kebolehcapaian semakin meningkat, masih wujud jurang ketara dalam sistem pengecaman masa nyata untuk Bahasa Isyarat Malaysia (BIM), khususnya pada peringkat ayat, serta dalam ketersediaan aplikasi penterjemah mudah alih yang boleh digunakan. Kajian ini membentangkan satu sistem pengecaman gerak isyarat berasaskan penglihatan untuk penterjemahan BIM secara masa nyata pada peringkat ayat, menggunakan Gated Recurrent Unit (GRU) yang digabungkan dengan Connectionist Temporal Classification (CTC) bagi mengendalikan pergantungan temporal dan jujukan panjang berubah. Model ini telah dinilai menggunakan kaedah 5-fold cross-validation, mencapai purata Kadar Ralat Perkataan (WER) sebanyak 0.0275 dengan sisihan piawai 0.0173, dan WER ujian akhir sebanyak 0.0132 selepas latihan menggunakan keseluruhan set data, mengesahkan keupayaan generalisasi model yang kukuh. Sistem ini telah diintegrasikan ke dalam aplikasi Android dan diuji secara masa nyata pada pengguna yang tidak dikenali, mencapai ketepatan 69%, dan purata WER 0.30 sekali gus menunjukkan potensinya untuk digunakan secara praktikal dan pada masa yang sama mendedahkan cabaran berkaitan variasi antara pengguna dan persamaan antara gerak isyarat. Kajian ini menyumbang ke arah merapatkan jurang kebolehcapaian bagi komuniti Pekak di Malaysia melalui penyelesaian penterjemahan masa nyata berasaskan penglihatan yang mudah alih.

List of Tables

Table 1	Summary of Related Work: Alphabet-Level and Word-Level Language Recognition System
Table 2	Summary of Related Work: Sentence-Level Recognition
Table 3	Comparison of Related Application
Table 4	Selected Sentences for Classification
Table 5	Data Augmentation
Table 6	Comparison Between the Result of Using Early Stopping
Table 7	5-fold cross-validation result
Table 8	Accuracy and Prediction Iterations for Specific Sign Language Sentence

List of Figure

Figure 2.1	User-friendly interface designed for all age group
Figure 2.2	Instruction for better ASL recognition
Figure 2.3	Sign language dictionary with short videos
Figure 2.4	Wrong desired output from the sign “I love you”
Figure 2.5	Get desired output after 5th attempt
Figure 2.6	Landing Page
Figure 2.7	Instruction for better KSL recognition
Figure 2.8	List of currently recognizable words
Figure 2.9	System able to recognise one hand
Figure 2.10	The landing page explains that if the recognized sentence is in sign language, it will automatically be translated into spoken language.
Figure 2.11	Instruction for better CSL recognition
Figure 2.12	Sign Language Dictionary with different category
Figure 2.13	Demonstration of the avatar performing the sign, with options for the user to adjust the speed and repetition
Figure 2.14	Text is generated at the bottom-left corner after the sign is performed
Figure 2.15	The home page displays the sign history, including multiple attempts
Figure 3.1	Overall Project Flowchart
Figure 3.2	Video Dataset from MFD
Figure 3.3	Definition of the 21 landmarks
Figure 3.4	The structure of GRU cell (Subramanian et. al., 2022)
Figure 3.5	The CTC model (Hannun, 2017)
Figure 3.6	The proposed model GRU + CTC
Figure 3.7	Flowchart for Android Application
Figure 4.1	Malaysian Sign Language in Medical Context dataset from Swinburne University of Technology Sarawak (SUTS) (arkuuu21, 2023)
Figure 4.2	Dataset from the Malaysia Federation of the Deaf (MFD) and the Society of the Deaf (SSD)
Figure 4.3	Directory structure of the video dataset
Figure 4.4	Distribution of video frames counts

Figure 4.5	Json file with tokenized sentence
Figure 4.6	Use of Mediapipe in frames extraction
Figure 4.7	Updated CSV file with augmented NumPy files
Figure 4.8	Python Logo
Figure 4.9	Jupyter Logo
Figure 4.10	Jupyterlab Logo
Figure 4.11	OpenCV Logo
Figure 4.12	Matplotlib Logo
Figure 4.13	MediaPipe Logo
Figure 4.14	NumPy Logo
Figure 4.15	PyTorch Logo
Figure 4.16	Scikit Learn Logo
Figure 4.17	Android Studio Logo
Figure 4.18	Trained model and vocabulary file is placed in assets directory
Figure 4.19	Sign Translate Page
Figure 4.20	Learn Sign Language Page
Figure 4.21	WER of Sentence with different GRU Layers and Units
Figure 4.22	Train VS Validation WER with Different Parameters Configuration
Figure 4.23	Final Model Training - Train vs. Test WER
Figure 4.24	User-independent testing with one of the signers

List of Abbreviations

ASL	American Sign Language
BIM	Malaysian Sign Language
CER	Character Error Rate
CNN	Convolutional Neural Networks
CSL	Chinese Sign Language
CSLR	Continuous Sign Language Recognition
CTC	Connectionist Temporal Classification
FSL	Filipino Sign Language
GRU	Gate Recurrent Unit
KSL	Korean Sign Language
LRCN	Long-term Recurrent Convolutional Network
LSTM	Long Short-Term Memory
MFD	Malaysian Federation of the Deaf
NSL	Nepali Sign Language
RFID	Radio-frequency Identification
RNN	Recurrent Neural Networks
SDGs	Sustainable Development Goals
SLR	Sign Language Recognition
SOS	Sum of Standard Deviations
SSD	Sarawak Society for the Deaf
STTN	Spatial-Temporal Fusion Transformer Network
WER	Word Error Rate

List of Equations

1. *Word Error Rate (WER)* – Equation (1)

Chapter 1 Introduction

1.1 Background

According to the World Federation of the Deaf, there are more than 70 million deaf people worldwide (United Nations, 2017). Over 80 percent of them reside in underdeveloped nations. They employ around 300 different sign languages among themselves. In Malaysia, the Deaf population comprises around 7.4% of the total population. There are about 52,000 children and teenagers who were born deaf within this group of people (Xi, 2023).

The main challenge for the Deaf community is the barrier of language, not sound (Wolfe et al., 2023). This is particularly true in the Malaysian context, where families often speak multiple languages, further complicating communication. Moreover, sign languages, including Malaysia Sign Language (BIM) or Bahasa Isyarat Malaysia (BIM), have linguistic processes that cannot be expressed in a linear order like spoken languages, as they rely on the properties of visual-gestural channels (Wolfe et al., 2023). BIM, like other sign languages, has a limited vocabulary and constructs sentences differently from spoken languages. For example, in BIM, the spoken phrase "How old are you?" is interpreted as "[You][age][what]." Common BIM grammatical structures, used by the Deaf community, include subject (s), object (o), and verb (v) variations, such as 'ov' (rice eat), 'vo' (eat rice), 'sov' (Ali rice eat), 'svo' (Ali eat rice), and 'ovs' (rice Ali eat).

Chinese Sign Language (CSL) illustrates how sign language structure can differ significantly from spoken language syntax, highlighting that sign language is an independent language with its own unique expressive logic. For instance, while "I Love You" in CSL follows the direct structure "[I][Love][You]," more complex sentences, like "I am going to Shanghai," are structured as "[Shanghai][I][Go]" rather than "[I][Go][Shanghai]." This difference reflects a visually logical sequence rather than spoken language order. Xu Cong (2021), Executive

Deputy Director of the Chinese Association of the Deaf's Sign Language Committee, explains that sign language and spoken language, such as Chinese, are fundamentally distinct, with Deaf individuals naturally acquiring sign language independently of spoken syntax. Attempts to impose spoken language structure on sign language, similar to how "Chinglish" arises from blending English and Chinese, result in "Signed Chinese," an unnatural fusion that can hinder clear communication (回形针 PaperClip, 2021).

This insight is relevant for languages like BIM, as imposing spoken Malay or English syntax onto BIM could disrupt its natural flow. Sign language also differs in its ability to convey multi-layered information simultaneously, with elements like subject, action, and direction expressed instantly. This structure aligns with visual processing and contrasts with the linear nature of spoken languages, underscoring the need to preserve sign language's unique structure for accurate, natural communication (回形针 PaperClip, 2021).

Given this complexity and the distinct linguistic logic of sign language, access to trained interpreters is crucial to bridge communication between the Deaf community and broader society. In Malaysia, there is a significant shortage of certified sign language interpreters. In 2021, only 95 certified interpreters were registered with the Malaysian Federation of the Deaf (MFD) (Faithhour, 2022). By 2023, Palanisamy, project lead for the "Health care needs of the Deaf (HEARD)" project, highlighted that there was only one Malaysian Sign Language interpreter for every 800 Deaf individuals in Malaysia, emphasizing the substantial gap in interpreter availability and the need for improved services in this area (Monash University Malaysia, 2023).

To support language learning and accessibility, the MFD has produced valuable BIM (Bahasa Isyarat Malaysia) resources. These include 18 reference books containing 10,000 signs

and two mobile applications (BIM Sign Bank, n.d.). Additionally, the MFD developed the BIM Sign Bank as an official reference, benefiting students, teachers, parents, and the general public. This resource categorizes signs into 12 sections, covering topics from general life to health. In collaboration with Guidewire Gives Back, MFD launched the “BIM by MFD” mobile application in July 2021, expanding access to sign language resources (BIM Sign Bank, n.d.). With an established vocabulary, learners can more easily progress to sentence formation, a natural step in language acquisition.

Several assistive technologies have also emerged to address the needs of the Deaf community. For example, MFD's own My Bahasa Isyarat Malaysia (MyBIM) app and Digi Telecommunications' EDDY (Edugame & Dictionary for Deaf) by BAXS Technology provide interactive language support (Malaysian Wireless, 2017). Devices like SignAloud gloves, launched in 2016, aim to automate sign language interpretation. While these tools show potential, they often lack practicality for daily use, as they require additional motivation to learn and carry.

Although most previous research on BIM recognition has achieved good accuracy, it has primarily focused on static sign language with a limited vocabulary, which is insufficient for practical, daily use (Khan et al., 2021). To address this limitation, in 2022, Padmanand and Lim (2022) proposes a deep-learning approach for real-time Malaysian Sign Language (BIM) recognition using 3D handpose estimation and action models. With TensorFlow and MediaPipe, the model detects 3D hand keypoints, enabling real-time interpretation of BIM and offering a potential solution to the shortage of interpreters in Malaysia. This aligns with the authors in (Khan et al., 2021), who suggest a need for a real-time BIM translation system to bridge the communication gap between the Deaf community and broader society.

Sign language sentences are structured with gestures separated by movement epenthesis, making sentence-level recognition complex and time-intensive, especially when using methods like fingerspelling or isolated word recognition (Al Abdullah et al., 2024). The authors noted that the research in this area is still relatively recent and limited, with continuous sign language translation at the word level remaining a significant challenge. Recognizing entire sentences adds another layer of difficulty, yet achieving accurate sentence-level recognition could substantially improve the practicality and real-life usability of sign language translation systems.

This project aims to develop an application for real-time sentence-level BIM gesture recognition using AI technology. It can significantly contribute to several Sustainable Development Goals (SDGs). By helping the Deaf communicate more effectively, the project promotes equal access to resources and services (SDG 10). Improved communication also supports inclusive education by making learning materials more accessible to people with hearing impairments (SDG 4), ultimately providing them with the confidence to join the workforce (SDG 8).

1.2 Problem Statement

The unique structure of sign language sentences relies on transitions between gestures rather than isolated words, making word-based or fingerspelling methods inefficient for natural communication (Meng et al., 2019; Khan et al., 2021). In the context of Malaysian Sign Language (BIM), there is a lack of real-time recognition systems capable of accurately interpreting sentence-level gestures and there are not many mobile translator applications for BIM on the market (Ke et al., 2022). This gap in technology hinders seamless interactions between Deaf and hearing individuals, particularly in essential settings like medical consultations.

1.3 Scope

The *Malaysian Sign Language Medical Dataset* (Arekku21, n.d.) from Kaggle will be processed using Google's MediaPipe Hand Landmarker model, combined with a Gated Recurrent Unit (GRU) and Connectionist Temporal Classification (CTC) for sentence-level recognition.

The five selected sentences are:

No.	Sentence
1	Doktor mahu saya pergi buat x-ray / the doctor wants me to go for x-ray
2	Saya muntah dan penat kerana mengandung / I'm vomiting and tired because I'm pregnant
3	Saya rasa batuk dan sakit tekak / I feel cough and sore throat
4	Saya membuat pembedahan untuk mata / I did eye surgery
5	Badan saya berasa gatal-gatal / my body feels itchy

Dataset video from MFD and Society Deaf (SSD) with repetition of each sentence at normal speed and slow speed at 10 times for each different speed. Each signer will have a total of 100 repetitions for all sentences.

1.4 Aims and Objectives

Aim

To develop a mobile application for real-time sentence-level BIM gesture recognition.

Objectives

1. To study current challenges and techniques for translating continuous hand and pose gestures effectively.
2. To implement a sentence-level real-time BIM gesture recognition model.
3. To evaluate the word error rate (WER) of the sentence-level real-time BIM translation application.

1.5 Significance of Project

This project demonstrates the feasibility of recognize BIM sign gestures in sentences level, serving as a potential solution toward more comprehensive translation systems. By addressing the practical need for basic communication in medical consultation, this project aims to facilitate interactions between Deaf and hearing individuals, thereby fostering inclusivity.

1.6 Expected Outcome

A prototype mobile application for real-time gesture recognition of Malaysian Sign Language (BIM) in sentence-level.

1.7 Summary

In this chapter, there are several components are used to explain the introduction of Real-Time Malaysia Sign Language Gesture Recognition in sentence-level such as problem statement, project scope, aim and objective, project significance, expected outcome and project schedule. The main problem to solve is the lack of real-time gesture recognition system to translate sentence-level BIM. Next, the aim and objectives are set to target on how the project will solve the problem. The project scope further explains on what is the focus of the project. Then, the expected outcome is described where there will be an application that can do real-time BIM gesture recognition in sentence-level. The significance of the project is also mentioned to demonstrate the feasibility of recognize BIM sign gestures in sentences level, serving as a potential solution toward more comprehensive translation systems.

Chapter 2 Literature Review

2.1 Introduction

This chapter reviews related work and applications in sign language recognition and translation. It highlights existing solutions' features, performance metrics, and limitations to identify gaps in current research. The insights derived from this review form the foundation for designing a robust real-time gesture recognition system for Malaysian Sign Language (BIM) at the sentence level.

2.2 Current Scenario Analysis

Sign language recognition (SLR) has been a prominent research focus in sign language processing for decades (Abdullah et al., 2024). Current systems aim to convert sign language gestures into readable text or audible speech, supporting inclusivity and accessibility for the Deaf community. These systems typically address both static signs, such as fingerspelling alphabets and numbers, and dynamic signs, which are more practical for real-world conversations as they convey words and sentences through sequential frames. However, while much progress has been made, translating dynamic signs, especially at the sentence level, remains a significant challenge.

Recent advancements in SLR have leveraged deep learning techniques to improve gesture recognition accuracy, but many systems primarily target isolated word recognition (Abdullah et al., 2024). Sentence-level recognition, which requires understanding the contextual and temporal relationships between consecutive gestures, is still in its infancy. Current approaches often struggle with overlapping frame sequences and ambiguity in dynamic transitions, particularly when multiple signs share similar starting frames (Luqman & El-Alfy, 2021). These limitations hinder the development of robust systems capable of practical application in real-world scenarios.

Given these challenges, there is a growing need for SLR systems that can handle the complexities of sentence-level gestures, including accurate feature extraction, seamless sequence alignment, and contextual interpretation. Addressing these gaps is essential for improving SLR's reliability and making it a viable tool for everyday communication.

2.3 Review of Related-works

2.3.1 Alphabet-Level and Word-Level Language Recognition System

Convolutional Neural Networks (CNNs) have emerged as a popular approach for sign language recognition due to their ability to extract meaningful features from raw data. Pre-trained models like Inception V3, trained on the ImageNet dataset, have demonstrated strong classification performance, even with limited training sets (Tupal et al., 2022).

However, while CNNs are effective for static images, sign language involves dynamic, sequential gestures that require temporal pattern recognition. To address this, Recurrent Neural Networks (RNNs) have been widely employed in tasks such as language modeling. Traditional RNNs, however, face challenges like vanishing and exploding gradient issues. Advanced variants, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), overcome these limitations. LSTM manages both short-term and long-term memory using a gating mechanism to focus on relevant information, while GRU simplifies the process by exposing the entire memory. The choice between LSTM and GRU often depends on the dataset and application.

Hybrid models that combine CNNs and RNNs have gained traction for sign language recognition, leveraging CNNs for feature extraction and RNNs for temporal sequence modeling. For instance, Tupal et al. (2022) achieved 86.75% accuracy in recognizing Filipino Sign Language (FSL) using a CNN-GRU combination. Similarly, Tran et al. (2023) demonstrated 99% accuracy with a model integrating one CNN layer, one LSTM layer, and dense layers. Advancements in lightweight models have further improved efficiency and real-time application. Prince et al. (2023) proposed a system combining MediaPipe Holistic with GRU, achieving 94.4% accuracy, outperforming LSTM and SimpleRNN, and demonstrating faster processing with fewer parameters.

Existing BIM recognition research have also integrated with RNNs. Padmanand and Lim (2022) used MediaPipe Hands for extracting 3D hand keypoints and integrated them with LSTM models, achieving an accuracy of 84.45% for BIM recognition. Similarly, Quzaimi (2023) utilized a hand landmark-based CNN model but noted challenges in obtaining large labeled datasets and optimizing computational resources. Despite these challenges, their system achieved 92% accuracy, although the limited dataset of 300 instances was identified as a constraint.

Practical applications have also been developed to enhance accessibility. Saiful Bahri et al. (2023) introduced an Android application for BIM recognition, incorporating MobileNet and SSD MobileNet V2 models. The app achieved 99.75% accuracy for recognizing BIM letters and 61.6% for common words. Additional features, such as speech-to-text functionality, enabled real-time communication, making the solution practical for bridging communication gaps without requiring human interpreters. The study highlighted future directions, including improving gesture recognition with attention models and enhancing speech recognition through audio-visual integration and lip-reading.

These advancements demonstrate the evolving potential of combining deep learning techniques and practical applications to address the complexities of sign language recognition. Future research should focus on optimizing models for low-resource settings, exploring attention mechanisms, and integrating multimodal capabilities for more robust and accessible solutions. The finding is summarized in *Table 1* below.

The paper	Target Language	Challenge to solve	Signs details	Input Data	Model	Performance Metric
(Padmanand, K., & Lim, P.-C., 2022)	Malaysian Sign Language (BIM)	Limited resources and research in BIM	• 6 words and 26 alphabets • 5 volunteers with repetition of 3 times for each sign.	MediaPipe Holistic for extracting and drawing 3D hand keypoints	Sequential model using Sequential API + 2-LSTM layers	Accuracy 84.45%
(Tupal et al., 2022)	Filipino Sign Language (FSL)	Lacking in FSL recognition	20 words signed by 4 deaf participants with at least 5 times repetitions.	MediaPipe to remove background	InceptionV3 Convolutional Neural Network (feature extraction) + LSTM or GRU + dense layer	Accuracy • LSTM 16 = 63.8554% • LSTM 32 = 54.2168% • GRU 16 = 86.7469% • GRU 32 = 81.9277%
(Maheswara et al., 2023)	Bahasa Isyarat Indonesia (BISINDO)	Examines the influence of facial landmarks on RNN models	20 words signed by 3 volunteers with 120 times repetitions	MediaPipe Holistic to capture face and hand key points	LSTM and GRU	Accuracy With face key points • LSTM 63% • GRU 65% Without face key points • LSTM 70% • GRU 83%
(Prince et al., 2023)	Makaton Sign Language (MSL)	lack of technology for accurately recognizing and understanding Makaton signs	3 words repeated 60 times	MediaPipe Holistic to capture hand key points	LSTM, GRU and Simple RNN	Accuracy • GRU 94.4% • LSTM 91.6% • SimpleRNN 91.6%
(Quzaimi et al., 2023)	Malaysian Sign Language (BIM)	High development cost using CNN due to the need of large amount of labelled data and high computing power.	24 alphabets recorded 300 times for each alphabet. 80% training data. 20% testing	MediaPipe Holistic for extracting and drawing 3D hand keypoints	Artificial Neural network (ANN)	Accuracy 92%
(Saiful Bahri et al., 2023)	Malaysian Sign Language (BIM)	Inadequate recognisable character database	• 29 letters in the datasets, including delete, nothing, and space 3000 images in each class, total 87000 images from Kaggle, self-generated dataset and Deaf/mute teachers' dataset. • 5 words with 100 images for each word	Hand gesture images with different position and light intensity	CNN model = SSD MobileNet V2 FPNLite 320×320 from TensorFlow 2 Detection Model Zoo	• 99.78% accuracy for alphabet recognition • 61.60% accuracy for word recognition
(Tran et al., 2023)	American Sign Language (ASL)	lack of fingers information or effective way to process data without compensate the performance in real-time.	20 words signs with 200 repetitions for each word	MediaPipe Hand and MediaPipe Pose (left and right elbow; left and right wrist)	One CNN layer (feature extraction) + one LSTM layer (sequence learning) + dense layer	99.5% accuracy

Table 1. Summary of Related Work: Alphabet-Level and Word-Level Language Recognition System

2.3.2 Sentence-Level Language Recognition System

In 2019, Meng et al. introduced the first sentence-level language recognition system utilizing Radio-frequency Identification (RFID) technology. This system collects phase sequences from signals detected by RFID devices, leveraging the signals' pure phase properties to perform sign language segmentation and recognition through a novel sum of standard deviations (SOS) feature. Extensive testing across various scenarios demonstrated high recognition accuracy and robustness, with average accuracy ranging from 96% to 98.11% under diverse multipath conditions.

Other notable non-vision-based sentence-level sign language recognition works include those by Wang et al. (2023) and Gu et al. (2024). Both studies utilized convolutional neural networks (CNNs) combined with recurrent neural networks (RNNs) and connectionist temporal classification (CTC) for sentence recognition and translation. CTC, a popular method for sentence recognition, eliminates the need for manual alignment during training through its specialized loss function. Wang et al. (2023) employed acoustic signals, achieving a low word error rate (WER) of 0.9%. Similarly, Gu et al. (2024) achieved a comparable WER of 0.93%. However, Gu's model faced challenges during real-time testing, with WER increasing to 72.9%, likely due to a limited participant pool that hindered the model's ability to generalize. Gu's research also introduced end-to-end translation by integrating an encoder-decoder structure, achieving a WER of 16.63% for the translated output.

In the domain of vision-based recognition, Akandeh (2022) proposed two frameworks for continuous sign language recognition (CSLR) from video sequences. These sentence-level SLR models process video data directly, bypassing the need for prior segmentation. The first model employed a Long-term Recurrent Convolutional Network (LRCN), which processed raw video frames using an 18-layer LRCN with a CTC layer for classification. This approach

operated without leveraging prior linguistic knowledge. The second model, termed the Multi-Cue Network, incorporated linguistic features such as hand shape, hand position, and hand movement. By utilizing Mediapipe to extract 2D hand landmarks, the system created hand skeletons, which were processed using a convolutional LSTM (CONV-LSTM). Additional features, like hand positions relative to the head, were analyzed through separate LSTMs before being integrated into the Multi-Cue Network. Both models were evaluated on the RWTH-PHOENIX Weather dataset, achieving a relatively high WER of 35, despite extensive parameter fine-tuning.

Bhusal and Baral (2024) proposed a system for recognizing Nepali Sign Language (NCSL) by combining the Mediapipe Holistic framework with a Spatial-Temporal Fusion Transformer Network (STTN). Tested on a dataset of 40 sentences recorded at Shree Deaf School in Bhairahawa, Nepal, their model achieved a WER of 20.23%. The STTN model enhanced feature extraction by simultaneously capturing spatial and temporal information. Patching video frames further minimized computational overhead and enabled efficient end-to-end training.

In the context of sentence-level recognition for BIM, Veron et al. (2025) utilized a combination of the VGG16 model with LSTM, GRU, and an encoder-decoder model. Their findings demonstrated that the GRU outperformed the LSTM in several metrics, including BLEU, ROUGE, and accuracy. Table 2 below summarizes the findings of sentence-level recognition.

The paper	Target Language	Challenge to solve	Signs details	Input Data	Model	Performance Metric
(Meng et al., 2019)	Chinese Sign Language (CSL)	- Recognize sentence-level sign language - Sign language segmentation	9 basic sentence-level sign language in 2 robust scenarios of History Museum with 8 volunteers repeat 100 times for each sign.	Vision- based system (RGB images commercial RFID device used)	Random Forest (RF) classifier	Accuracy between 96% and 98.11%
(Akandeh, Atra, 2022)	German Sign Language (DGS)	Overcome lighting conditions, complex background, and camera position	RWTH-PHOENIX-Weather dataset, consists of 5672 sentences for training with 65,227 signs and 799,006 frames in total.	Black and white skeleton images using Mediapipe from raw data	- Raw Input Sign-CTC (RSign-C) - Multi-Cue Sign-C (MCSign-C)	45.1% Word Error Rate (WER) for Rsign-C 35.3% WER for MCSign-C
(Wang et al., 2023)	American Sign Language (ASL)	Aiming for more ubiquitous, low cost, and accurate ASLR system	30 sentences with 10 participants signing 10 times for 2 sessions	Acoustic signals	CNN + GRU + CTC	0.9% WER
(Gu et al., 2024)	American Sign Language (ASL)	Implementing sign language recognition & end-to-end sign language translation	300 ASL sentences signed by 3 volunteers with 20 times repetitions	Motion with different in amplitude and trajectory	SLR: CNN + Bi-directional LSTM + CTC End-to-end translation: Encoder-decoder structured model	SLR: 0.93% WER End-to-end translation: 16.63% WER Sentence-level user-interdependent validation-SLR: 72.90% WER
(D. Bhusal and D. S. Baral, 2024)	Nepali Sign Language (NSL)	Limited resources and research in NSL CSLR	1640 videos of Nepali continuous sign language gestures, covering 40 classes	Frame extraction with Mediapipe Holistic	Spatial-Temporal transformer model, coupled with Mediapipe Holistic	20.32% WER
(Veron et al., 2025)	Malaysian Sign Language (BIM)	Promote inclusive education	10 sentences with 7 participants signing the 3 times for each sentence	BIM video processed using VGG-16 model	CNN + LSTM/GRU + encode-decoder model	BLEU-3: LSTM 0.4645 GRU 0.5050 BLEU-4: LSTM 0.3378 GRU 0.3643 ROUGE-L: LSTM 0.8350 GRU 0.8692 Accuracy: LSTM 0.7010 GRU 0.7580

Table 2. Summary of Related Work: Sentence-Level Recognition

2.4 Review of Related Application/ System

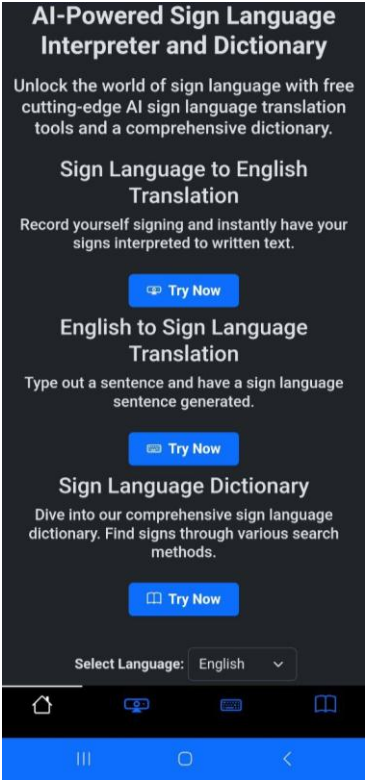
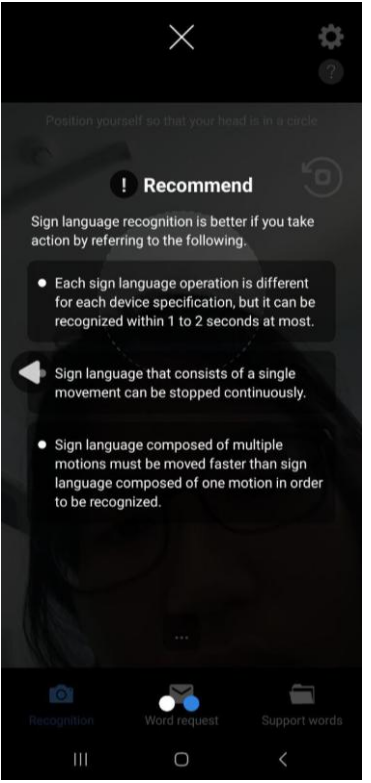
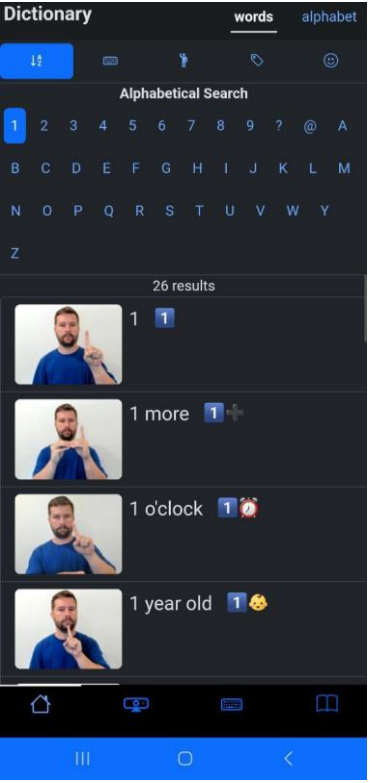
2.4.1 Sign Language AI

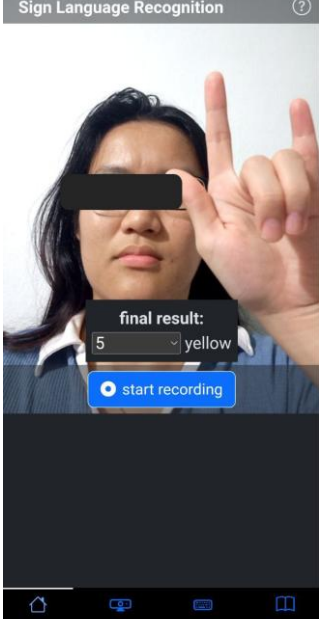
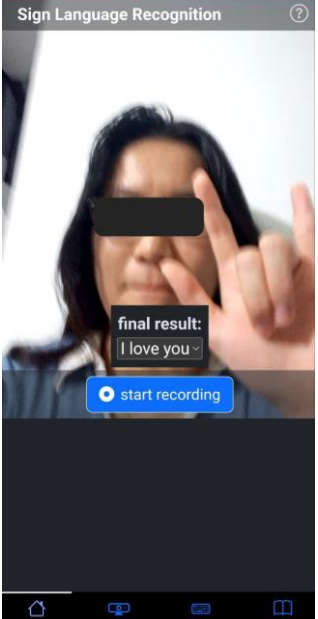


Sign Language AI is a mobile application developed by Kelly (2023) and released on February 25, 2022. It supports two-way communication, including text-to-sign language and sign language-to-text. The application offers several notable features, including real-time translations with unparalleled accuracy, a dictionary with easy-to-follow tutorials, and multilingual support to overcome language barriers. The application focuses on American Sign Language (ASL), recognizing over 2,600 signs and the 26 letters of the alphabet. Users can also access the sign language dictionary section to view all the recognized signs, each accompanied by a 10-20 second video tutorial to help users master the art of signing.

The mobile application was tested with the sign "I love you," and it was observed that multiple attempts were necessary to achieve accurate recognition. The interface and results are summarized in Figures 1–5 below.

Summary for Sign Language AI

		
<i>Figure 2.1.</i> User-friendly interface designed for all age group	<i>Figure 2.2.</i> Instruction for better ASL recognition	<i>Figure 2.3.</i> Sign language dictionary with short videos

	
<i>Figure 2.4.</i> Wrong desired output from the sign “I love you”	<i>Figure 2.5.</i> Get desired output after 5th attempt

2.4.2 Sign Language Translator AI



Sign Language Translator AI is a mobile application developed by Nam (2024) and released on July 24, 2022. Although it does not support two-way conversations, the application offers several key features, including real-time translations using the smartphone camera's bitmap data with text output, the ability to create sign language videos for developers, a list of currently recognizable words, and an engine that dynamically adjusts the recognition range based on the smartphone's performance. The application focuses on Korean Sign Language (KSL) and is currently capable of recognizing 279 frequently used signs. Users can also access the sign language dictionary section to view all the recognized signs. Additionally, the application has created over 300,000 training data files.

Due to a lack of familiarity with Korean Sign Language and the absence of videos in the sign language dictionary section, testing the application was not feasible. Notably, the application is designed to recognize gestures performed with one hand rather than two. While the application demonstrates the capability to detect sign language, its accuracy could not be verified under these conditions. The interface and results are summarized in Figures 6–9 below.

Summary for Sign Language Translator AI

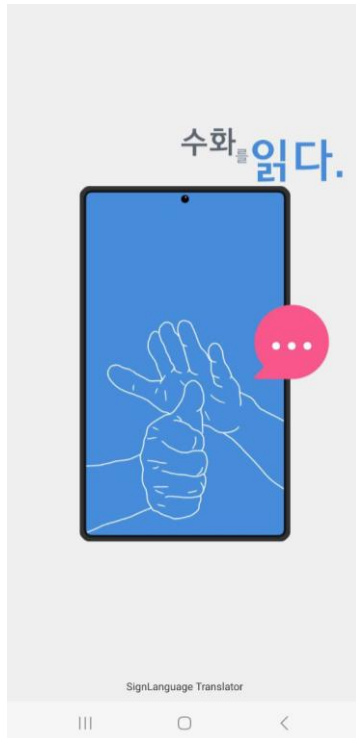


Figure 2.6. Landing Page

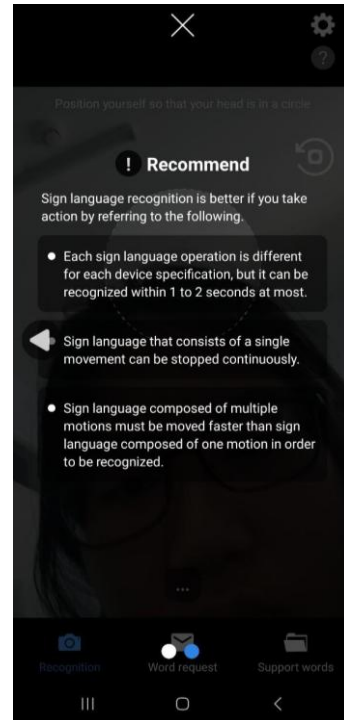


Figure 2.7. Instruction for better KSL recognition



Figure 2.8. List of currently recognizable words

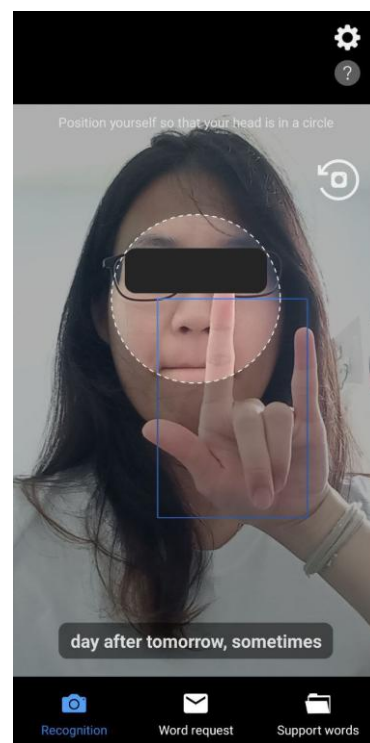


Figure 2.9. System able to recognise one hand

2.4.3 Vivo Sign Language Interpreter (手语翻译官)



The *Vivo Sign Language Interpreter* (手语翻译官) is a mobile application developed by the Vivo OriginOS Technology R&D team (Chand et al., 2024). It supports two-way communication, including speech-to-text, text-to-speech, text-to-sign language, and sign language-to-text. In addition to its two-way communication capabilities, the application offers a sign language dictionary with avatar demonstration at 3 different speeds. The application focuses on Chinese Sign Language (CSL) and it can recognize 1,200 sign language vocabulary, with over 80 percent accuracy rate. This is equivalent to the level of Chinese HSK Level 4 where it has covered more than 8,000 words in the Chinese general dictionary.

The application was tested with the phrase "I love you." Unlike American Sign Language (ASL), where "I love you" is expressed as a single sign, CSL divides this phrase into three distinct words. The application demonstrated high accuracy in recognizing individual words, such as "I," which was identified without errors. However, challenges were encountered when forming a complete sentence, as multiple attempts with consistent speed were required to achieve accurate recognition. The interface and results are summarized in Figures 10–15 below.

Summary for Vivo Sign Language Interpreter (手语翻译官)

 手语识别进行语音播报	 1、请将上半身置于检测框内	
Figure 2.10. The landing page explains that if the recognized sentence is in sign language, it will automatically be translated into spoken language.	Figure 2.11. Instruction for better CSL recognition	Figure 2.12. Sign Language Dictionary with different category
		
Figure 2.13. Demonstration of the avatar performing the sign, with options for the user to adjust the speed and repetition.	Figure 2.14. Text is generated at the bottom-left corner after the sign is performed.	Figure 2.15. The home page displays the sign history, including multiple attempts

2.5 Analysis and Discussion

Based on results and trends in word-level recognition, models combining CNN-RNN or MediaPipe with RNN architectures, particularly GRU, have demonstrated higher accuracy. While Padmanand and Lim (2022) utilized LSTM in their research, GRU has not yet been tested for BIM word-level recognition.

Studies specifically focused on sentence-level BIM recognition remain limited. Researchers have primarily explored related challenges such as segmentation, lighting variations, and resource constraints. Despite leveraging diverse input types, including skeleton data, and extracted frames, existing models continue to exhibit significant limitations.

Notably, Veron et al. (2025) conducted the only known research on sentence-level BIM recognition, demonstrating that GRU outperforms LSTM for this task. This aligns with findings from word-level recognition studies by Tupal, Maheswara, and Prince, which reported GRU accuracies of 86.75%, 83%, and 94.4%, respectively—surpassing LSTM performance by 3% to 20% (see summary in *Table 1* and *Table 2*).

The fundamental difference between GRU and LSTM lies in their architecture. LSTM employs two additional gates (forget and input gates) for managing temporal information over extended sequences, making it more effective for complex languages but also more resource-intensive. GRU, with its simpler architecture and single update gate, is less complex, easier to train, and highly effective for simpler sign languages. This makes GRU more suitable for applications where computational efficiency and smaller datasets are key considerations.

Additionally, recent works on sentence-level recognition (e.g., Akandeh, Atra, Wang, Gu, Bhusal, and Baral) highlight the use of CTC for handling segmentation and variable-length sequences. CTC eliminates the need for manual segmentation by learning to align sequences with transcription labels directly, enabling efficient end-to-end training.

To address the identified gaps and improve computational efficiency, this study proposes the use of a MediaPipe + GRU + CTC model. This approach aims to evaluate its effectiveness in advancing sentence-level BIM recognition, particularly in scenarios where resource constraints and real-time application are critical considerations.

While the proposed approach is promising, it is important to acknowledge potential limitations. First, the quality and diversity of the dataset could significantly impact model performance. A limited dataset may result in the model overfitting or underperforming in real-world scenarios, especially in diverse environments with varying lighting and backgrounds.

Second, the reliance on MediaPipe for feature extraction could pose challenges when dealing with occlusions or complex hand gestures, potentially affecting recognition accuracy. MediaPipe's landmark detection might struggle with signs involving intricate finger movements or signs performed at extreme angles (Tupal et al., 2022).

Third, while GRU models are computationally efficient, they may face limitations in capturing highly complex temporal dependencies compared to LSTM, which could impact sentence-level recognition accuracy for longer or intricate sentences.

Finally, real-time implementation of the MediaPipe + GRU + CTC model on mobile platforms may require optimization to ensure smooth performance without compromising accuracy. Resource-constrained devices, such as older smartphones, may struggle to handle the processing demands of real-time gesture recognition.

Addressing these limitations will involve curating a diverse and representative dataset, optimizing model parameters, and exploring hybrid approaches such as testing with LSTM to improve robustness.

In terms of mobile application development, three mobile applications targeting sign languages (ASL, KSL, and CSL) were reviewed in this section. Each application offers unique functionalities but suffers from limitations. *Sign Language AI* provides a simple interface with an extensive dataset but sacrifices accuracy. *Sign Language Translator AI* supports limited vocabulary (279 signs) and restricts recognition to single-hand gestures. *Vivo Sign Language Interpreter* demonstrates promising accuracy (over 80%) but requires improvements in sentence-level recognition to enhance usability in real-world contexts. All the applications highlight the need for combining advanced AI techniques with extensive datasets to support continuous and contextual sign language translation. With the review of the related application, the pros taken into consideration are to have a simple interface and provide sign bank for user to learn. The features comparison is summarized in *Table 3*.

The mobile application	Target language	Number of recognised signs	Features	Performance	
				Pro	Con
(Kelly, 2023)	American Sign Language (ASL)	over 2,600 signs and 26 alphabets	Sign-to-Text; Text-to-Sign	Simple interface with vast data	Low accuracy rate
(Nam, 2024)	Korean Sign Language (KSL)	279 frequently used signs	Sign-to-Text only	Able to direct collab and contribute with developer	Only able to recognise one hand
(Vivo Software, 2024)	Chinese Sign Language (CSL)	1,200 sign language vocabulary	Speech-to-Text; Text-to-Speech; Text-to-Sign; Sign-to-Text	Over 80% accuracy rate (Chand et al., 2024)	Require improvement in accuracy rate for sentence level recognition

Table 3: Comparison of Related Application

2.6 Summary

Recent advancements in word-level recognition demonstrate that combining CNN-RNN or MediaPipe with GRU achieves higher accuracy than LSTM. While GRU has not been tested for BIM word-level recognition, Veron et al. (2025) showed its superiority over LSTM for BIM sentence-level recognition. Studies on sentence-level BIM remain limited, addressing challenges like segmentation and resource constraints. Recent works highlight the use of connectionist temporal classification (CTC) for efficient end-to-end training by eliminating the need for manual segmentation and handling variable-length sequences. Therefore, implementation of a MediaPipe + GRU + CTC model is proposed to address these gaps and evaluate its effectiveness for sentence-level BIM recognition. A simple interface and sign bank feature will be available for user after the review of mobile applications.

Chapter 3 Methodology

3.0 Introduction

This chapter outlines the methodology for developing a BIM sentence-level gesture recognition mobile application, designed to translate Malaysian Sign Language (BIM) into structured Malay (BM) text. The approach follows a **research methodology, focusing quantitative methods** and consists of four key stages: data preprocessing, sequence modelling, project implementation, and performance evaluation. In the preprocessing stage, frames are extracted from raw video using OpenCV, and hand key points are detected using Google MediaPipe. The sequence modelling stage employs an RNN GRU with CTC to capture temporal dependencies and handle unaligned sequence-to-sequence learning. The trained model is then integrated into an Android mobile application featuring a "capture sign" function and a "sign bank" for storing recognized gestures. Finally, model performance is assessed using Word Error Rate (WER), comparing results from controlled training and user-independent testing to evaluate real-world accuracy.

3.1 Proposed Architecture

The proposed application mainly focuses on user who uses sign language during medical consultation. User shall use this application to get the translation of captured BIM signs in English and Malay in text format.

There are two main concerns in order to realise the real-time sentence-level recognition. First is the efficient architecture where low computational overhead is needed to ensure smooth performance on mobile devices and secondly is the capability for temporal data to handle sequential gesture information to classify sentences.

Firstly, the user will be asked to give permission to allow the application to access the device camera. The capture and recognition of the hand gestures is done through Mediapipe,

which is mostly approved by several works on its high accuracy of recognition of hand gestures and lightweight which is optimized for mobile devices. Once the hand gestures have been captured and normalized, they are simultaneously fed to the Gate Recurrent Unit (GRU) Model in conjunction with Connectionist Temporal Classification (CTC) to handle sequence-to-sequence learning without requiring explicit alignment. The model will then be converted from a PyTorch model into the TorchScript format and import into Android Studio. The translated gestures are captured using camera in real time and return translated text output. The main purpose of this system is to achieve portability without cutting back on accuracy.

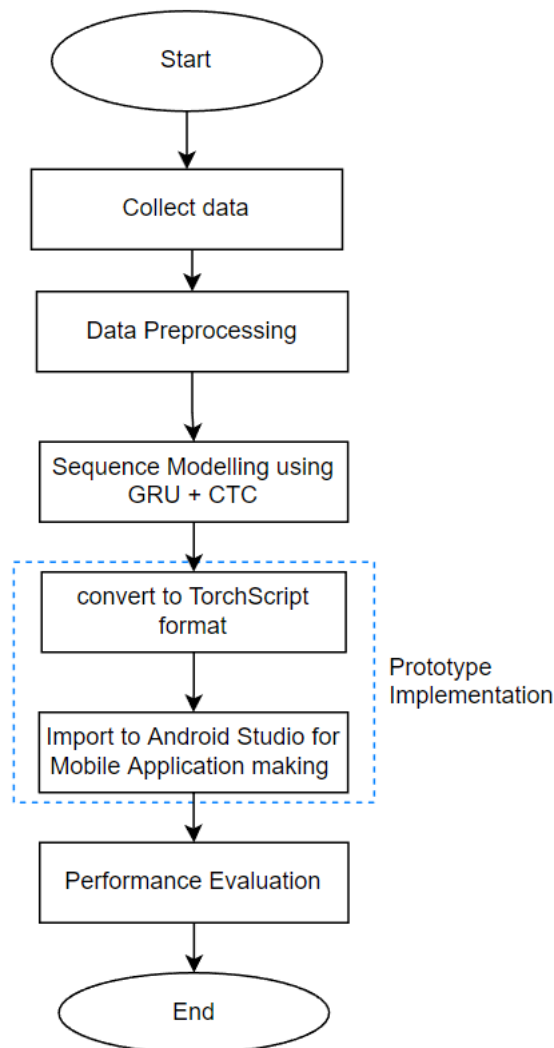


Figure 3.1. Overall Project Flowchart

3.2 Dataset Creation

The dataset consisted of 5 sentences for classification, aiming at medical used scenarios to simulate real-world scenarios. All the sentences selected are less than 10 words to minimized the Word Error Rate (WER). The 5 selected sentences are as shown in Table 4.

No.	Sentence	Average Duration
1	Doktor mahu saya pergi buat x-ray / the doctor wants me to go for x-ray	5-33 seconds
2	Saya muntah dan penat kerana mengandung / I'm vomiting and tired because I'm pregnant	5-10 seconds
3	Saya rasa batuk dan sakit tekak / I feel cough and sore throat	5-13 seconds
4	Saya membuat pembedahan untuk mata / I did eye surgery	5-13 seconds
5	Badan saya berasa gatal-gatal / my body feels itchy	5-13 seconds

Table 4. Selected Sentences for Classification

The research was proposed to Malaysia Federation of The Deaf (MFD) (shown in *Figure 17*) and Society Deaf (SSD). A total of nine signers contributed to these datasets, and the videos were recorded using webcams at 30 FPS in indoor setting. The duration for all the collected video is around five to 33 seconds.

Ethical considerations were a cornerstone of the data collection process for this research, ensuring the respectful and transparent involvement of all participants. Prior to any data collection, all signers were provided with and signed a comprehensive consent form. This form clearly outlined the research title, "Real-Time Malaysian Sign Language Gesture Recognition (Sentence-Level)," and explicitly stated that the data collected would be used solely for research purposes. Participants were informed that all personal data would be treated with strict confidentiality, and they provided their agreement for their information to be used anonymously in statistical or research reports and case studies. Furthermore, the consent form clarified that signers waived any copyright claims should the material be published in scholarly journals or online. Importantly, participants were informed of their right to withdraw consent at any time

by contacting the researchers, whose contact details were readily available on the form. This thorough consent process ensured that all participants were fully informed and willingly participated in the study, upholding the highest ethical standards.

With consent provided, each of the sentences were captured with a repetition of ten times at normal speed and ten times at slow speed, hence, each sentences have a total of 100 repetitions.

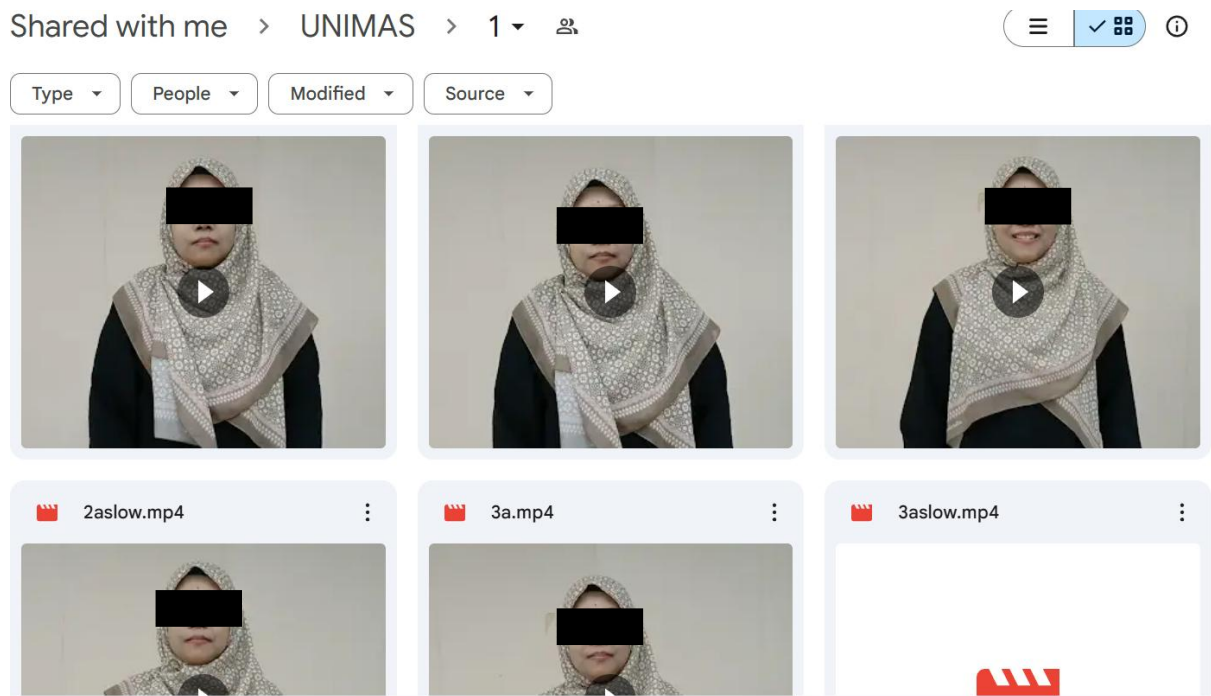


Figure 3.2. Video Dataset from MFD

Other than data collection from the MFD, novel educational dataset focuses on medical context from Swinburne University of Technology Sarawak is also utilise. The dataset consists of 1040 videos with 40 unique glosses and 10 sets of text annotations in Malay including English translation (Veron et al., 2025). The utilised sentences from are transcribed in BM/ENG which are constructed from the glosses data set, so that they correspond to the approximate meanings of the sign language.

3.3 Data Preprocessing and Feature Extraction

The raw video data is first segmented into frames using OpenCV, which are then processed by Google MediaPipe's Hand Landmarker for real-time key point extraction. This tool identifies specific hand gestures and provides both hand landmarks and recognized gestures as outputs.

MediaPipe employs a machine learning model to analyze both static images and continuous video streams. Its output includes hand landmarks in image and world coordinates, hand orientation (left or right), and categorized gestures for multiple hands. The system operates using two pre-packaged model bundles: (1) the Hand Landmark Model, which detects hand presence and geometry, including 21 landmark points per hand (as shown in Figure 18); and (2) the Gesture Classification Model, which interprets gestures based on the detected hand geometry.

Each hand is represented by 126 key points per frame, calculated as 21 key points $\times$ 3 dimensions (x, y, z) $\times$ 2 hands. These extracted key points serve as essential features for training the sign language recognition model. The dataset is split into 80% for training, 10% for validation and 10% for testing to ensure robust model performance.

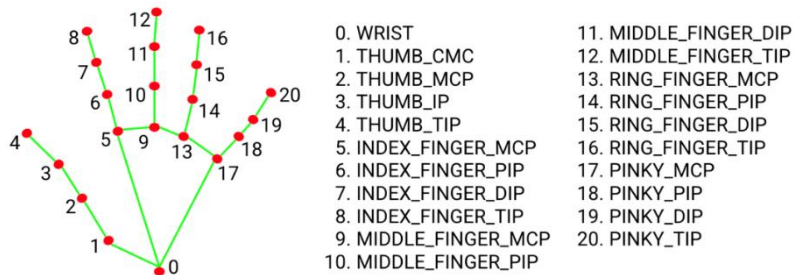


Figure 3.3. Definition of the 21 landmarks

3.4 Model Selection and Justification

The output from MediaPipe serves as input to the RNN-based GRU model. Each RNN layer includes dropout to prevent overfitting. Based on Tupal et al. (2022), using 16 units achieves

optimal performance, as increasing the unit count can degrade performance and prolong training time.

GRUs process sequential data step-by-step, maintaining a hidden state that captures temporal relationships, which is essential for understanding gesture evolution and how earlier frames

Due to the small dataset and short sequences, the number of GRU layers are experimented in this project. The input layer defines the shape of the input data but does not perform any computation; here, the input features consist of 126 key points. The based setting will be two GRU layers, where the first GRU layer processes the input sequence and transforms it into a more compact representation. The second GRU layer captures deeper sequential dependencies, enhancing feature extraction. The structure of GRU cell is summarized in Figure 3.4.

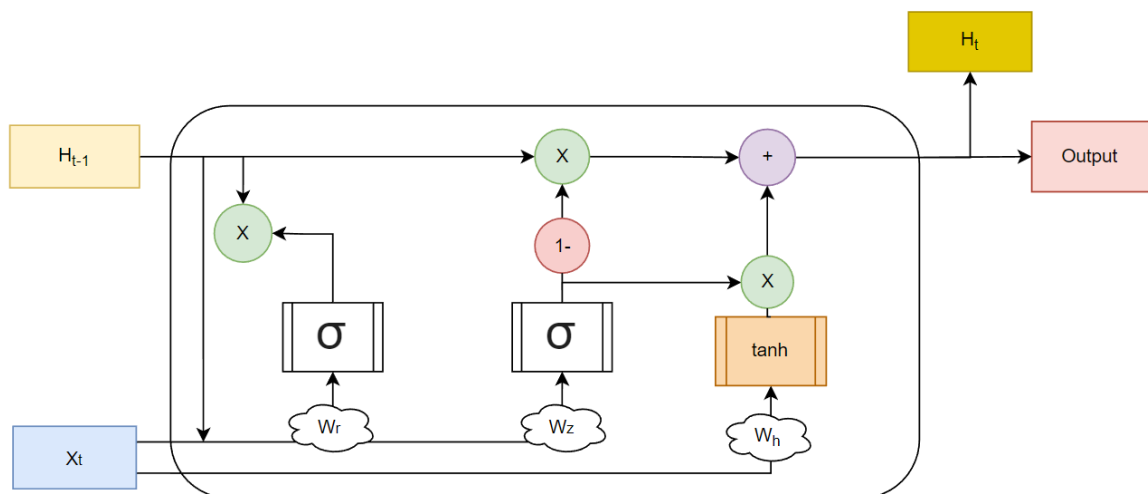


Figure 3.4. The structure of GRU cell (Subramanian et. al., 2022)

The GRU's output is passed to a Connectionist Temporal Classification (CTC) classifier, which generates gloss label sequences. CTC is a specialized loss function and decoding algorithm for sequence-to-sequence tasks without strict alignment between input and

label sequences. This makes it ideal for sign language, as it segments continuous signing into individual words and handles unaligned data where gestures do not directly map to single frames.

The sequence is processed through a recurrent neural network (RNN), which outputs a probability distribution over possible characters at each time step, including a blank symbol (ϵ). By marginalizing over multiple possible alignments, the final predicted output is obtained. *Figure 3.5* illustrates how CTC enables models to learn mappings from variable-length inputs to outputs without requiring explicit alignment labels.

In this work, CTC maps gesture sequences to text while managing variable input lengths and redundant predictions. For instance, the model may output repeated words or gaps (e.g., "Doktor Doktor Doktor - - mahu - saya"), but the CTC layer condenses this into a coherent sentence, such as "Doktor mahu saya pergi buat x-ray." The project model is summarized in *Figure 3.6*.

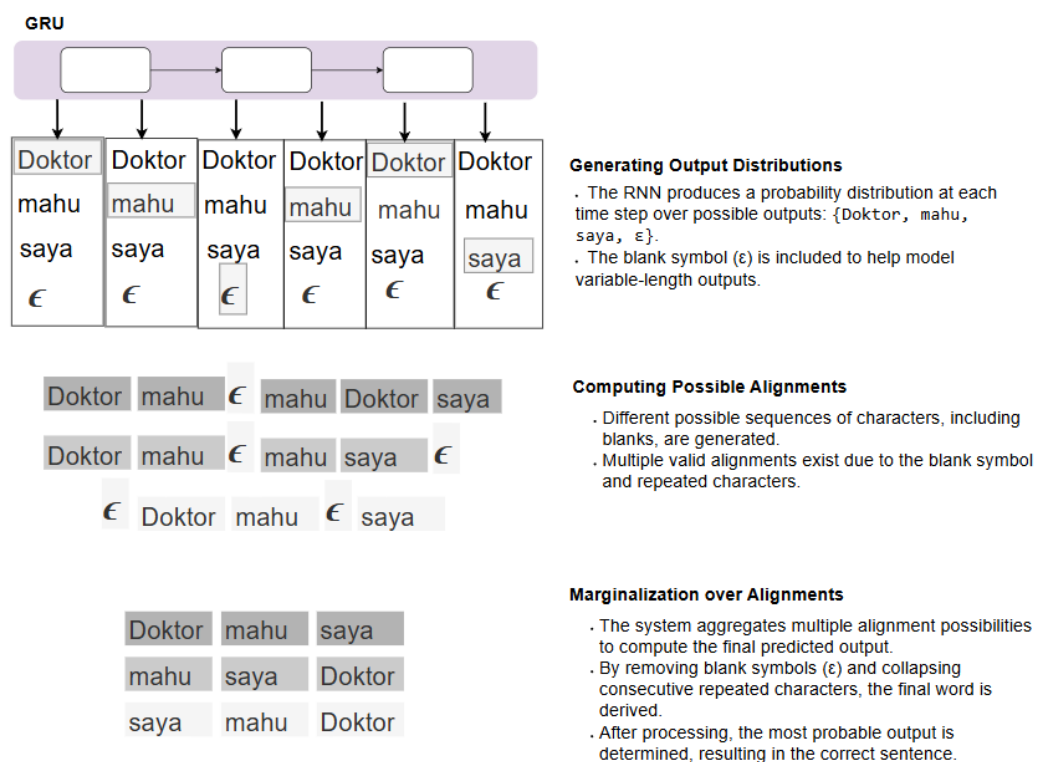


Figure 3.5. The CTC model (Hannun, 2017)

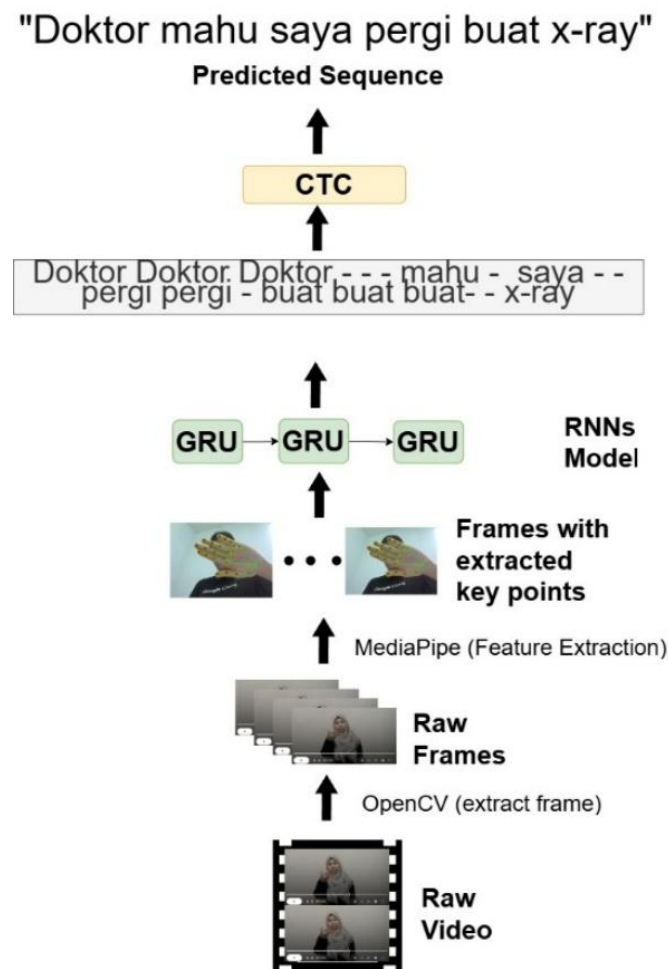


Figure 3.6. The proposed model GRU + CTC

3.5 Prototype Implementation

The trained model will be converted from a PyTorch model into the TorchScript format for optimized performance. It will then be imported to the mobile application created using Android Studio, integrating MediaPipe for sentence recognition. The mobile application will be a simple interface with main page, capture sign page, and a sign dictionary with short video page. The process flow of the Android application of this BIM recognition is presented in *Figure 3.7*.

To use the application, several stages and features user need to fulfil such as:

1. User needs to turn on the internet connection.
2. Download and install the apps on their smartphone.

3. User must allow the apps to use the camera.

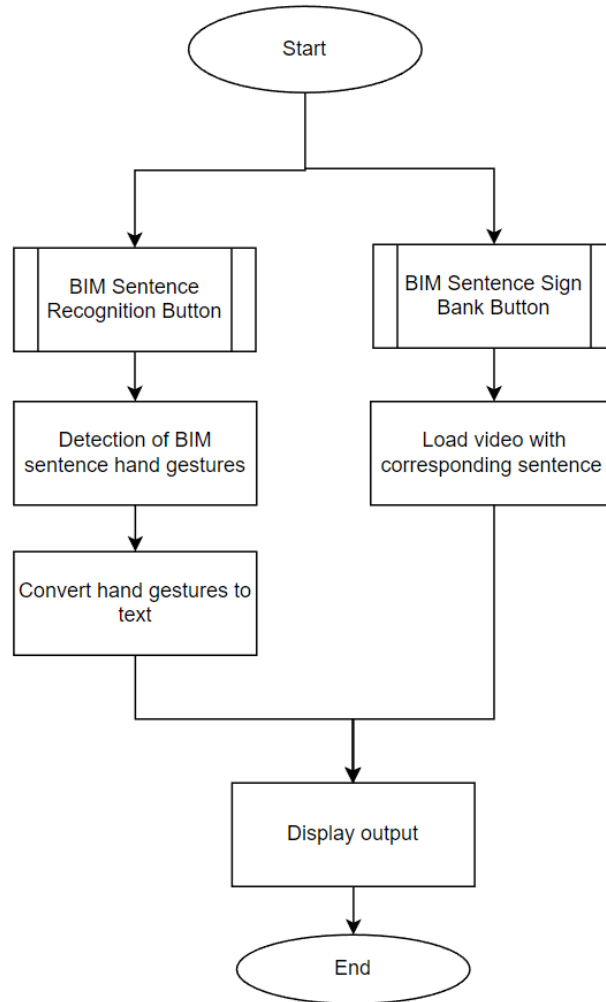


Figure 3.7. Flowchart for Android Application

3.6 Evaluation Metric and Validation Technique

To evaluate the performance and generalization ability of the sign language recognition model, a two-stage evaluation process was employed. This process began with k-fold cross-validation during model development and hyperparameter tuning to ensure robust model performance and prevent overfitting. This was followed by a final evaluation on a dedicated test dataset.

For the primary evaluation metric, the Word Error Rate (WER) was adopted as it is a standard metric for speech recognition and machine translation systems (Hou et al., 2019; Akandeh, Atra, 2022; Wang et al., 2023; D. Bhusal and D. S. Baral, 2024). It serves as a key

metric for assessing CSLR systems, comparing the model's output—based on current parameters—with the actual sentence vector in accurate sign language. While WER is a standard benchmark for system comparison and performance evaluation, it provides limited insight into the specific nature of translation errors, which needs deeper analysis.

The word error rate can then be computed as:

$$WER = \frac{D+I+S}{D+C+S} = \frac{D+I+S}{N}, \quad (1)$$

where,

- D is the number of deletions,
- I is the number of insertions,
- S is the number of substitutions,
- C is the number of correct words,
- N is the number of words in the reference ($N=S+D+C$)

The WER of a translation system should be as low as possible.

Character Error Rate (CER) is a commonly used metric similar to Word Error Rate (WER), with the key difference being that CER is based on character counts rather than word counts. In this study, CER was not calculated due to the limited number of finger-spelled words. However, in larger datasets containing more finger-spelled words, calculating CER becomes crucial for effective evaluation.

To assess performance, two types of metrics are computed, firstly is the trained model WER where it evaluates the model on test set data; and secondly is the User-Independent Testing accuracy where it tests the model on unseen user data to analyze real-time performance differences.

3.7 Summary

This section summarizes the methodology for developing a BIM sentence-level gesture recognition mobile application, which consists of four main stages. First, data preprocessing involves extracting frames from raw video using OpenCV and processing them with Google MediaPipe to obtain hand key points. Second, in sequence modeling, an RNN GRU with CTC is employed to capture temporal relationships within gesture sequences and handle unaligned sequence-to-sequence tasks. Third, project implementation integrates the trained model into an Android mobile application, which includes a "capture sign" feature and a "sign bank" for storing recognized gestures. Finally, performance evaluation compares the trained model's WER and user-independent testing accuracy to assess its accuracy in both controlled and real-time scenarios.

Chapter 4 Experimental Results and Discussion

4.1 Introduction

This chapter presents the experimental results obtained from the proposed Real-Time Malaysian Sign Language Gesture Recognition system, building upon the methodologies and model architecture detailed in the preceding chapters. Following the data collection and preprocessing steps, which involved segmenting raw video data into frames using OpenCV and extracting key points with Google MediaPipe's Hand Landmarker, the processed features were fed into the developed RNN-based GRU model. This chapter will delve into the performance evaluation of the model using the Word Error Rate (WER) metric, as established for assessing sentence-level recognition accuracy.

4.2 Dataset Description and Preparation

4.2.1 Video Collection

The main data source for this study was the Malaysian Sign Language (BIM) dataset in a medical context, obtained from Kaggle user arkuuu21 (2023). The dataset consists of 1040 videos with 40 unique glosses and 10 sets of text annotation in Malay with English translation. For this project, we utilized five of the ten available text annotation sets. The dataset included recordings from seven sign language signers, each repeating every sentence three times. The first two repetitions employed a formal approach, while the third utilized an informal approach, deemed more practical and intuitive by sign language experts. The videos were recorded in the Green Screen Studio at Swinburne University of Technology Sarawak (SUTS) using high-definition (HD) webcams with a resolution of 1920x2080 at 30 frames per second (FPS). Datasets from the Malaysia Federation of the Deaf (MFD) and the Society of the Deaf (SSD) were also utilized. A total of nine signers contributed to these datasets, and the videos were recorded using webcams at 30 FPS in indoor setting. The duration for all the collected video is around three to 20 seconds, the sample video files are shown in Figure 4.1 and Figure 4.2.

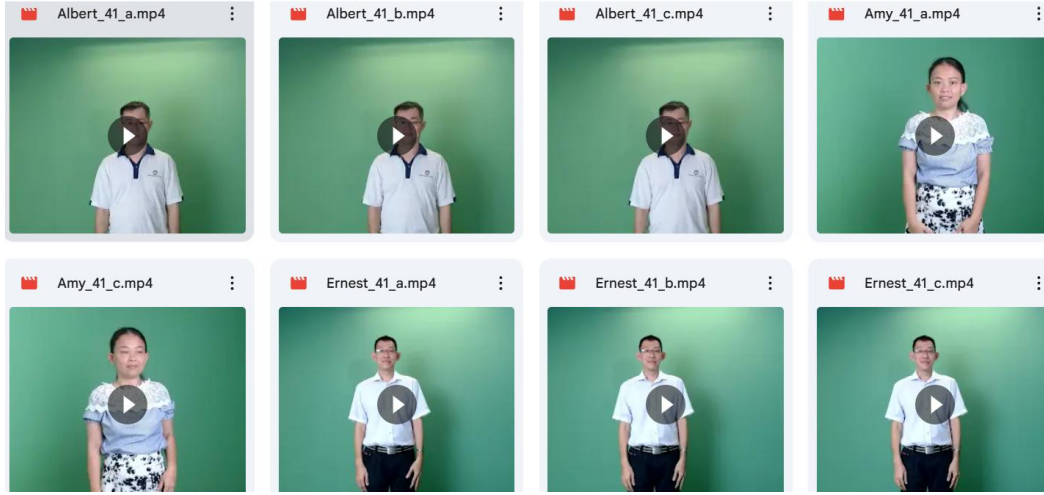


Figure 4.1. Malaysian Sign Language in Medical Context dataset from Swinburne University of Technology Sarawak (SUTS) (arkuuu21, 2023)

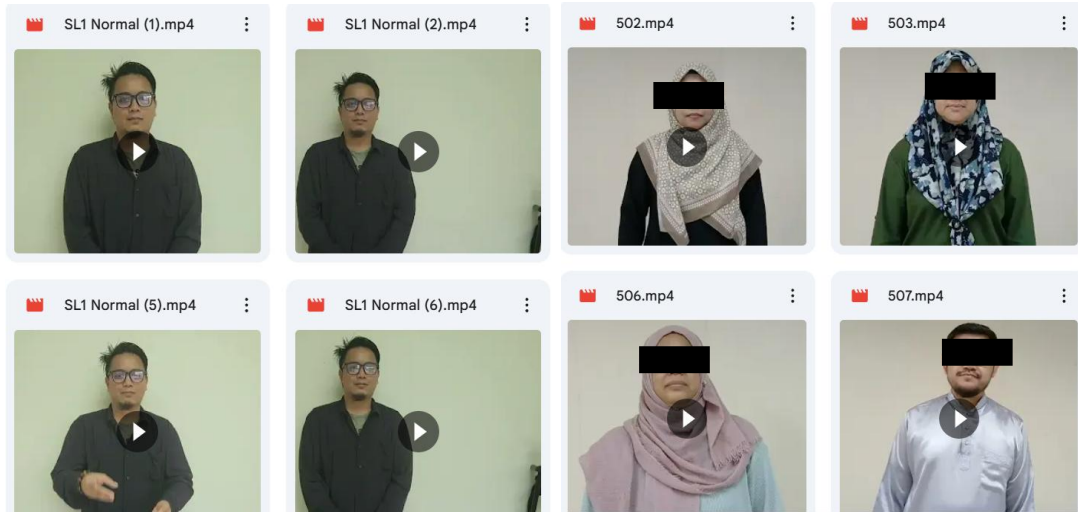


Figure 4.2. Dataset from the Malaysia Federation of the Deaf (MFD) and the Society of the Deaf (SSD)

4.2.2 Video Folder

The dataset's file structure consisted of a top-level directory named `video_dataset`. Within this directory, each unique sentence was represented by a separate subdirectory (e.g., `sentence_1`, `sentence_2`, and so on). Each subdirectory contained the video recordings corresponding to that specific gloss. The video files within each gloss directory were named using a numerical identifier (e.g., `101.mp4`, `102.mp4`), allowing for easy identification of individual recordings. For instance, the subdirectory `sentence_1` contained video files related to the first gloss, such as `101.mp4` and `102.mp4`. Similarly, videos for the second gloss were located in the `sentence_2`

subdirectory (e.g., 201.mp4). This structure allowed for logical grouping of video data based on the expressed sentence and is illustrated in Figure 4.3.

```
video_dataset/
  .ipynb_checkpoints/
  sentence_1/
    101.mp4
    102.mp4
    103.mp4
    ... (and 31 more files)
  sentence_2/
    201.mp4
    202.mp4
    203.mp4
    ... (and 32 more files)
  sentence_3/
    301.mp4
    302.mp4
    303.mp4
    ... (and 30 more files)
  sentence_4/
    401.mp4
    402.mp4
    403.mp4
    ... (and 30 more files)
  .ipynb_checkpoints/
  sentence_5/
    501.mp4
    502.mp4
    503.mp4
    ... (and 30 more files)
  .ipynb_checkpoints/
```

Figure 4.3. Directory structure of the video dataset

4.2.3 Video Preprocessing

To prepare video data for sign language analysis, a pipeline was developed to extract meaningful features from each video. Instead of directly converting videos to a torch tensor format due to potential processing overhead, this pipeline focused on extracting hand landmarks using the MediaPipe library. This process generated a sequence of 3D coordinates for 21 hand landmarks across each frame of the video. To handle variations in signing speed and video length, Dynamic Time Warping (DTW) was employed to align the landmark sequences to a reference sequence within each sentence category. Subsequently, to ensure consistent input for a machine learning model, distribution plot (in Figure 4.4) was analysed to determine the maximum number of frames and sequence length for model configuration based on the number of frames per video. After several experiments with 50, 75, 100, 200 frames, the aligned sequences were interpolated to a fixed length of 50 frames, as the hand landmarks closely aligned with the original video the most. This resulted in a structured dataset where each video was represented by a NumPy array of consistent dimensionality.

The pipeline generated an output directory mirroring the input dataset structure. For each sentence folder in the original dataset, a corresponding folder was created in the output directory to store the processed landmark sequences as individual .npz files. The naming convention of these files was derived from the original video filenames, with the .mp4 extension replaced by .npz. OpenCV was utilized to read each video frame by frame, enabling the extraction of hand landmarks. Furthermore, the number of frames in the original videos was implicitly considered during the DTW alignment and explicit interpolation steps to achieve a uniform sequence length for subsequent analysis.

A CSV file was generated, with two columns featuring “label” and “filename” to store sentence labels and the processed landmark sequences. A tokenizer was constructed using the label column from the CSV file with the tokenizer library to create a word2idx.json file (Figure 4.5).

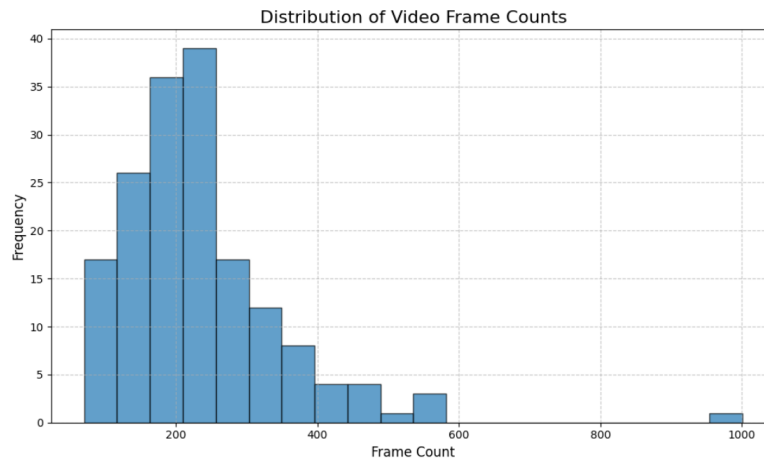


Figure 4.4. Distribution of video frames counts

```

root
  badan 1
  batuk 2
  berasa 3
  buat 4
  dan 5
  doktor 6
  gatal-gatal 7
  kerana 8
  mahu 9
  mata 10
  membuat 11
  mengandung 12
  muntah 13
  pembedahan 14
  penat 15
  pergi 16
  rasa 17
  sakit 18
  saya 19
  tekak 20
  untuk 21
  x-ray 22
  <BLANK> 0

```

Figure 4.5. Json file with tokenized sentence

4.2.4 Extraction of Frames

The “extract_landmarks_from_video” function was defined to process video files to extract hand joint coordinates using the MediaPipe library. It initializes a Hands object with specific detection and tracking confidence thresholds, and a maximum of two hands. For each frame of the input video, the function detects hand landmarks. If hands are detected, the 3D coordinates (x, y, z) of 21 landmarks per hand are flattened into a feature vector. If only one hand is detected, the feature vector is padded with zeros for the second hand to maintain a consistent size. In the absence of any detected hands, a warning is printed, and the feature vector for that frame is filled with zeros for both hands. The function accumulates these frame-wise feature vectors into a sequence, which is then converted into a NumPy array and returned. This array represents the temporal evolution of hand landmarks within the video. OpenCV (cv2) is used to read the .np video frames, and MediaPipe (mp) handles the hand landmark detection. The .np landmarks were crosschecked with the original video to verify the accuracy of landmark extraction (Figure 4.6).

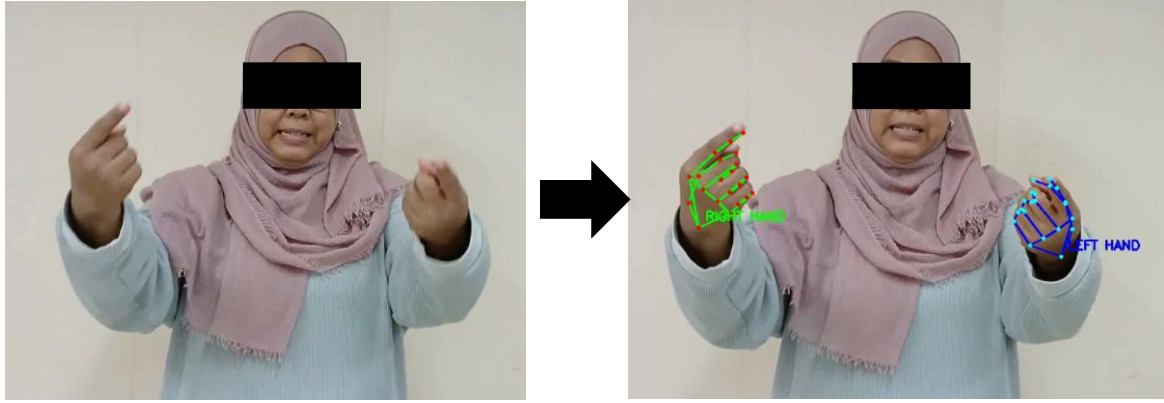


Figure 4.6. Use of Mediapipe in frames extraction

4.2.5 Data Augmentation

Data augmentation was implemented to expand the hand landmark dataset and enhance the robustness of sign language recognition models. .npy files containing landmark sequences located in the "output" directory were being processed. For each original sequence, three augmentation techniques, namely, horizontal flipping, adding noise and random rotation were applied and summarised in Table 5.

The original and the four augmented versions of each landmark sequence were saved as separate .npy files in a newly created "new_output" directory, mirroring the input directory structure. This augmentation strategy increases the dataset size by a factor of five for each original sample, providing a richer training set for subsequent model development. The updated CSV file is shown in Figure 4.7.

After augmentation, the dataset expanded from 168 data to 840 data points. The subsequent data preparation involved scanning the augmented data directory ('new_output'), mapping the file paths to their corresponding sentence labels using a predefined mapping, and creating a comprehensive 'dataset.csv' file. This file was then subjected to a stratified splitting process, dividing the data into training (80%), validation (10%), and testing (10%) sets, while

preserving the class proportions. The resulting datasets were saved as ‘train.csv’ (672 samples), ‘val.csv’ (84 samples), and ‘test.csv’ (84 samples) for subsequent model training and evaluation.

	filename	label
1	sentence_1/101.npy	dokter mahu saya pergi buat x-ray
2	sentence_1/101_combo.npy	dokter mahu saya pergi buat x-ray
3	sentence_1/101_flip.npy	dokter mahu saya pergi buat x-ray
4	sentence_1/101_noise.npy	dokter mahu saya pergi buat x-ray
5	sentence_1/101_rot.npy	dokter mahu saya pergi buat x-ray
6	sentence_1/102.npy	dokter mahu saya pergi buat x-ray
7	sentence_1/102_combo.npy	dokter mahu saya pergi buat x-ray
8	sentence_1/102_flip.npy	dokter mahu saya pergi buat x-ray
9	sentence_1/102_noise.npy	dokter mahu saya pergi buat x-ray
10	sentence_1/102_rot.npy	dokter mahu saya pergi buat x-ray
11	sentence_1/103.npy	dokter mahu saya pergi buat x-ray
12	sentence_1/103_combo.npy	dokter mahu saya pergi buat x-ray
13	sentence_1/103_flip.npy	dokter mahu saya pergi buat x-ray
14	sentence_1/103_noise.npy	dokter mahu saya pergi buat x-ray

Figure 4.7. Updated CSV file with augmented NumPy files

Table 5. Data Augmentation

Original video		
		
Augmentation Technique	Description	Augmented video
Horizontal Flipping	Simulating mirrored viewpoints of the signing hand by inverting the x-coordinates of the landmarks	
Additive Noise	Add random noise within a range of ± 0.02 to all landmark coordinates (x, y, z). This aims to improve the model's resilience to minor tracking inaccuracies.	
Random Rotation	Applying a random in-plane (XY) rotation with a maximum angle of ± 15 degrees to the hand landmarks. This helps the model generalize to slight variations in the signer's orientation	
Combined Augmentation	A sequential application of horizontal flipping, additive noise (with a range of ± 0.02), and random rotation (with a maximum angle of ± 15 degrees) to generate more diverse synthetic data.	

4.3 Algorithm Implementation

4.3.1 Tool used

1. Python Programming Language (version 3.11.9)



Figure 4.8. Python Logo

Source: <https://docs.python.org/3/faq/general.html#what-is-python>

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. The way Python is interpreted makes it fast to develop programs. Because it is high level, a developer can focus on the problem to be solved rather than on details of the implementation. It comes with a large standard library, hence, many common tasks requiring significant programming in other languages can be done in Python (Python Software Foundation, n.d.).

Python was utilized as the primary programming language due to its versatility and extensive library support, which are crucial for tasks such as:

- **Image and Video Processing:** Libraries like OpenCV were used for capturing, processing, and analyzing video frames of sign language gestures.
- **Machine Learning and Deep Learning:** Libraries such as TensorFlow and PyTorch were employed for building and training models to recognize sign language patterns.
- **Data Handling and Manipulation:** Libraries like NumPy were used for efficient data manipulation and analysis of the sign language dataset.

2. Jupyter Notebook (version 7.3.2)



Figure 4.9. Jupyter Logo

Source: <https://jupyterlab.readthedocs.io/en/latest/>

Jupyter Notebook is an open-source web application that allows create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. Jupyter Notebooks provide an interactive environment where code can be written and executed in blocks (cells), with the output displayed directly below the code. This facilitates an iterative and exploratory approach to data science and scientific computing. Its support for various programming languages (through kernels), rich media output (including plots and images), and the ability to integrate Markdown and LaTeX for documentation make it a powerful tool for communication and collaboration (The Jupyter Project, n.d.).

Jupyter Notebook provided an interactive and organized environment, used for data exploration, preprocessing, model development, visualization of results. Its cell-based structure allowed for iterative experimentation and clear documentation of the different stages of the project.

3. Jupyter Lab (version 4.3.5)



Figure 4.10. Jupyterlab Logo

Source: <https://jupyterlab.readthedocs.io/en/latest/>

JupyterLab is the next-generation web-based user interface for Project Jupyter. It offers all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and powerful extensible environment. JupyterLab enables users to arrange and customize their workspace with multiple notebooks, code consoles, terminals, and other components side by side or tabbed. It includes features like drag-and-drop cell rearranging, multiple views of the same notebook, and improved support for large notebooks. Its extensibility allows developers to create and integrate new components, enhancing its functionality for diverse workflows in data science, scientific computing, and education (The Jupyter Project, n.d.).

JupyterLab facilitated in managing multiple notebooks and scripts simultaneously, comparing different model implementations side-by-side, utilizing integrated terminal for system commands. Its flexible interface and organizational features contributed to a more efficient and streamlined development process.

4. OpenCV (version 4.11.0)



Figure 4.11. OpenCV Logo
Source: <https://opencv.org/>

OpenCV is open source, contains over 2500 algorithms, and is operated by the non-profit Open-Source Vision Foundation. Since June 2000 (OpenCV, n.d.). OpenCV (Open-Source Computer Vision Library) was utilized for video processing. Specifically, the `cv2.VideoCapture` function was used to access video files, and the `get(cv2.CAP_PROP_FPS)`

method was employed to determine the frame rate of the video sequences. This information about the video's temporal characteristics was extracted for potential analysis.

5. Matplotlib (version 3.10.1)



Figure 4.12. Matplotlib Logo

Source: <https://matplotlib.org/>

Matplotlib was created by neurobiologist John Hunter to work with EEG data. It is a comprehensive library for creating static, animated, and interactive visualizations in Python, used for data exploration and presentation (Matplotlib, n.d.). In this project, it was employed to visualize the trend of frames per second (FPS) across different video files. This allowed for a visual assessment of the temporal consistency of the input data.

6. Mediapipe (Hand Landmarker version 0.10.21)



Figure 4.13. MediaPipe Logo

Source: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

MediaPipe is a framework developed by Google for building multimodal applied machine learning pipelines. It provides pre-built and customizable solutions for tasks like hand tracking, pose estimation, and more. The MediaPipe Hand Landmarker task was employed to detect and track key anatomical points (landmarks) of hands within image data. This functionality allowed for the precise localization of hand features, providing both image-based and world-coordinate representations of these landmarks, as well as determining the handedness (left or right) of each detected hand across static images and continuous video streams (Google, n.d.). The `mp.solutions.hands` module was used to instantiate a Hands object, configured to detect and track up to two hands in the video frames. Parameters such as

`min_detection_confidence` and `min_tracking_confidence` were set to control the sensitivity of the hand tracking model. The `extract_landmarks_from_video` function was designed to process video frames and extract the 3D coordinates of key hand landmarks.

7. Numpy (version 1.26.4)



Figure 4.14. NumPyLogo
Source: <https://numpy.org/doc/stable/>

NumPy is a fundamental package for numerical computation in Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It can efficiently store and manipulating the numerical data representing hand landmarks, video frames, and other relevant data to performing mathematical operations necessary for data processing and machine learning tasks (NumPy, n.d.). The extracted sequences of hand landmarks were converted into NumPy arrays for streamlined processing. Furthermore, NumPy's capabilities were utilized within the `interpolate_to_fixed_length` function to perform linear interpolation, ensuring that all gesture sequences had a consistent length of 50 frames, a crucial step for subsequent analysis or model training.

8. Fastdtw

FastDTW is an efficient approximation of the Dynamic Time Warping (DTW) algorithm. It achieves optimal or near-optimal sequence alignments with a linear time and memory complexity ($O(N)$), significantly reducing the computational resources compared to the standard DTW algorithm's quadratic complexity ($O(N^2)$). This efficiency is gained through a multi-level approach that iteratively refines alignment solutions starting from a lower resolution (Salvador & Chan, 2007). FastDTW addresses the challenge of comparing gesture sequences

that might have variations in speed or timing. It is a crucial component in data processing pipeline, enabling the alignment and comparison of temporal data (hand movements).

9. Pytorch (version 2.6.0)



Figure 4.15. PyTorch Logo

PyTorch is an open-source machine learning framework that provides a flexible and efficient platform for building and training neural networks. In this project, PyTorch served as the primary deep learning tool, enabling the implementation of a Gated Recurrent Unit (GRU) based classifier (GRUClassifier). The `torch.nn` module was utilized to define the architecture of this neural network, including the GRU layer (`nn.GRU`) for processing sequential data and a linear layer (`nn.Linear`) for the final classification. The `torch.optim` module provided the Adam optimization algorithm (`optim.Adam`) used to update the model's weights during training. Furthermore, PyTorch's `torch.utils.data` module facilitated the creation of data loaders (`DataLoader`) and the management of dataset subsets (`Subset`) for efficient training and validation. The framework's tensor operations and automatic differentiation capabilities were crucial for the forward and backward passes during the training process, and `torch.device` allowed for seamless execution on either a CPU or a GPU, leveraging hardware acceleration when available.

10. scikit-learn (version 1.6.1)



Figure 4.16. Scikit Learn Logo

Scikit-learn is a Python library for machine learning, offering a wide range of tools for tasks such as classification, regression, clustering, and model selection. Scikit-learn's `model_selection` module was specifically utilized for implementing k-fold cross-validation through the `KFold` class. This technique is essential for robustly evaluating the performance of the GRU classifier and for tuning its hyperparameters. By splitting the dataset into multiple folds, training the model on a subset, and validating on the remaining fold iteratively, k-fold cross-validation provides a more reliable estimate of the model's generalization ability compared to a single train-validation split. The results from each fold were then aggregated to obtain an average performance metric for each hyperparameter combination explored during the grid search.

11. Android Studio (version 21.0.5)



Figure 4.17. Android Studio Logo

Android Studio served as the primary Integrated Development Environment (IDE) for this project. As the official IDE for Android application development, it provided tools essential for coding, debugging, testing, and compiling the Android application. Key features utilized included its intelligent code editor, built-in emulators for testing across various device

configurations, a flexible build system based on Gradle, and integrated version control tools. The tool is utilised to create the real-time sign language gesture recognition application.

4.3.2 Prototype Implementation

4.3.2.1 Android Integration

The trained PyTorch model was integrated into an Android application for real-time sign language recognition using camera input and MediaPipe hand tracking.

4.3.2.2 PyTorch Model Export

The trained GRU + CTC model for Malaysian Sign Language sentence recognition was developed and validated using PyTorch in Python. Once the model was trained, it was exported to the TorchScript format (.pt) for use on Android.

The model was traced with sample input of shape $[T, 126]$ (where T is the number of frames, and 126 corresponds to flattened hand landmark coordinates: $21 \text{ landmarks} \times 2 \text{ hands} \times 3 \text{ dimensions}$). It is then exported using `torch.jit.trace()` or `torch.jit.script()` and saved as `android_model.pt` and added to the Android app's assets directory (shown in Figure 4.18).

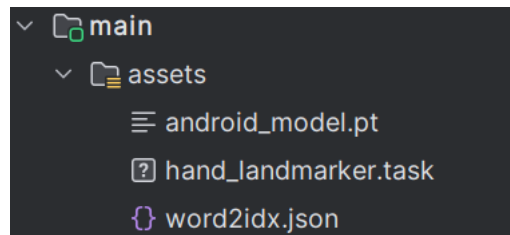


Figure 4.18. Trained model and vocabulary file is placed in assets directory

4.3.2.3 Kotlin Inference via ModelRunner.kt

A custom Kotlin class, `ModelRunner.kt`, was created to handle model loading, preprocessing, inference, and decoding. PyTorch Android API was used to load the .pt model and each input sequence with `FloatArray` of shape $[T, 126]$ is normalized and padded to a fixed length at 50 frames. During inference, the model will output raw logits, then a CTC beam search decoder was implemented in Kotlin using the loaded `word2idx.json` mapping.

4.3.2.4 Real-Time CameraX + MediaPipe Setup

In order to use real-time recognition, CameraX was used to access the phone camera stream in real time. MediaPipe HandLandmarker was used to detect and track both hands. The “HandLandmarker.task” file (shown in Figure 4.18) from MediaPipe was integrated and used to extract 3D hand landmarks.

When implementing the real-time recognition, the camera feed will be processed via ImageProxy. It will then be converted to Bitmap and passed into MediaPipe's MPIOImage to further extract List<Landmark> from both hands and flatten them to a FloatArray of size 126 ($21 \times 2 \times 3$).

4.3.2.5 Frame Buffering and Decoding Logic

To perform sentence-level prediction, a ring buffer of 50 frames was used to collect sequential landmark data. Buffer starts filling only when the "Start" button is pressed. When full, the buffer is passed to ModelRunner.kt for inference. Resulting sentence is displayed in the UI and updated in real time. It is implemented by using a MutableList<FloatArray> to store 126-element vectors which will automatically clear when “End” is pressed or after prediction is made.

4.3.2.6 UI Rendering with HandOverlayView

A custom view (HandOverlayView.kt) overlays the hand landmarks detected by MediaPipe onto the camera preview. It draws lines connecting joints and circles for each landmark point and handles both hands using MediaPipe's handedness classification (as shown in Figure 4.19). Integrated directly above PreviewView in the layout using a FrameLayout.

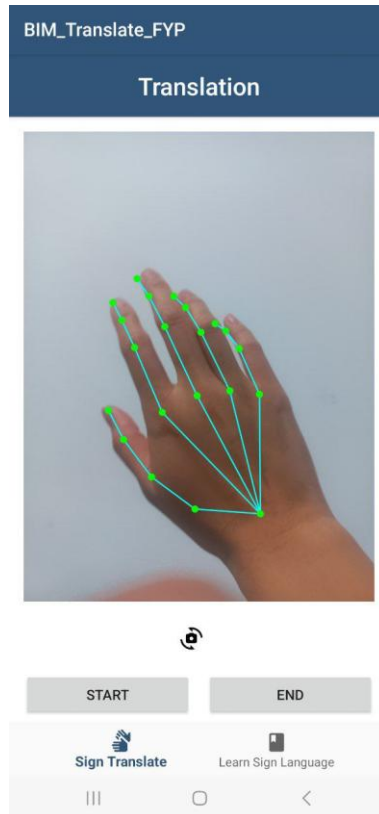


Figure 4.19. Sign Translate Page

4.3.2.7 Learn Sign Language Page

The *Learn Sign Language* page provides users with a list of five pre-recorded videos demonstrating Malaysian Sign Language sentences. Each video is displayed alongside its corresponding sentence for easy reference. The purpose of this page is to support users in learning common BIM sentences visually. This page is implemented using a RecyclerView, and each list item contains a video player on the left and the sentence label on the right. Figure 4.20 below shows the Learn Sign Language page layout.



Figure 4.20. Learn Sign Language Page

4.4 Training and Testing Procedures

4.4.1 Gated-Recurrent Unit + Connectionist Temporal Classification Model

The model architecture employed is Gated Recurrent Unit (GRU)-based sequence model designed for sentence-level recognition of Malaysian Sign Language (BIM) using hand landmark features. The model is trained using Connectionist Temporal Classification (CTC) loss, which enables the alignment between input sequences and variable-length output word sequences without requiring pre-segmented data.

4.4.2 Input Features

Each input sample is a sequence of 126-dimensional hand landmark vectors extracted from video frames. This corresponds to 21 landmarks per hand (left and right), each represented by (x, y, z) coordinates: $21 \text{ landmarks} \times 2 \text{ hands} \times 3 = 126 \text{ features per frame}$.

4.4.3 GRU-based Architecture

The core of the model is a Bidirectional GRU layer, which captures both past and future temporal dependencies in the input gesture sequence. The base configuration uses a hidden size of 256 units, resulting in a 512-dimensional output per time step due to the concatenation of forward and backward outputs. This is then passed to a fully connected linear layer, which maps the temporal features into the vocabulary space, followed by a Softmax function applied at each time step to produce a probability distribution over all possible vocabulary words plus a CTC blank token.

To assess the effect of model capacity on performance, several configurations of the GRU architecture are experimented with. These include variations in the number of hidden units (32, 64, 128, 256, 512) and the number of stacked GRU layers (1, 2, 3). The aim is to evaluate the trade-off between model complexity, accuracy, and inference efficiency for real-time applications.

4.4.4 Output and Loss

The output shape of the model is **(B, T, V)**:

- **B**: Batch size
- **T**: Sequence length (time steps)
- **V**: Vocabulary size (including the blank token)

The model is trained using CTC Loss, which computes the alignment between input and target sequences and handles varying sequence lengths and timing variations. The loss uses the padding token index as the blank symbol.

4.4.5 Decoding Strategy

During inference, beam search decoding is employed instead of the greedy decoding approach, where greedy decoding only selects the token with the highest probability at each time step and remove consecutive duplicate tokens. Beam search considers multiple candidate sequences at

each time step rather than selecting only the most likely token. This increases the likelihood of finding the optimal output sequence, especially in ambiguous cases.

The decoding steps are as follows:

1. At each time step, the decoder maintains a set of the top k most probable partial sequences (beams), where k is the beam width.
2. For each beam, it extends the sequence with possible next tokens and updates the cumulative probabilities.
3. After processing all time steps, the sequence with the highest overall probability is selected as the final output.
4. It then duplicates and blank tokens are removed in post-processing.

Beam search enhances recognition quality by improving prediction accuracy, particularly for longer or more complex gesture sequences.

4.4.6 Model Validation and Testing

The performance of the GRU + CTC model is evaluated with different parameters configuration which impacts the recognition accuracy. Different numbers of units and layers in GRU will be tested and evaluated with the WER of the sentence-level data set. These include variations in the number of hidden units (32, 64, 128, 256, 512) and the number of stacked GRU layers (1, 2, 3). The aim is to evaluate the trade-off between model complexity, accuracy, and inference efficiency for real-time applications.

To evaluate the performance and generalization ability of sign language recognition model, a two-stage evaluation process was employed: k-fold cross-validation during model development and hyperparameter tuning, followed by a final evaluation on a dedicated test dataset.

4.5 Interpretation of Experimental Results

4.5.1 Parameter Configuration Testing Result

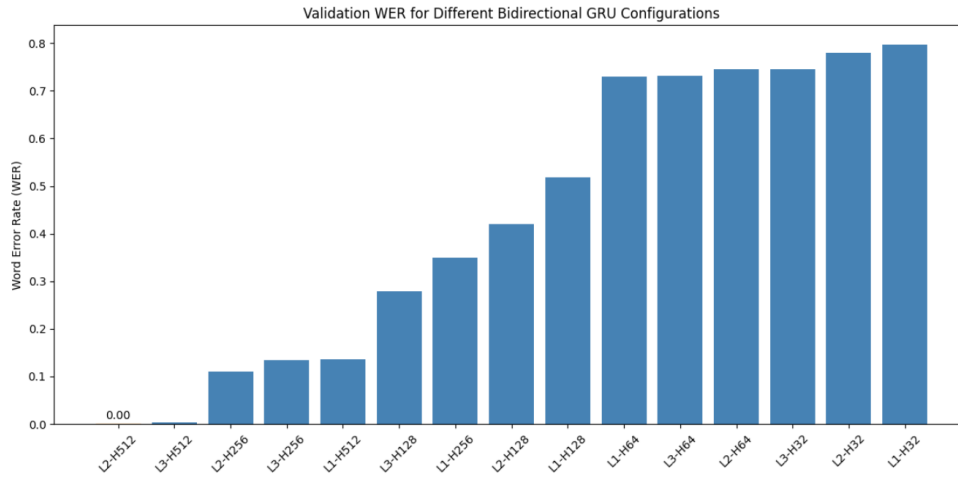


Figure 4.21. WER of Sentence with different GRU Layers and Units

Word Error Rate (WER) was adopted as it is a standard metric for speech recognition and machine translation systems (Hou et al., 2019; Akandeh, Atr, 2022; Wang et al., 2023; D. Bhusal and D. S. Baral, 2024). It serves as a key metric for assessing CSLR systems, comparing the model's output—based on current parameters—with the actual sentence vector in accurate sign language. A lower WER indicates higher accuracy, meaning the system made fewer mistakes in transcribing the gestures. Conversely, a higher WER signifies poorer performance. Figure 4.21 shows the increasing of WER across different configurations. Despite different hidden layers, it is shown that the hidden unit of 32 shows higher WER followed by hidden unit 64, then 128 with WER ranging from 0.8 to 0.3. Hidden unit 256 and 512 shows good WER rates at under 0.15. Since the graph only shows the validation set performance, the overfitting of these settings cannot be observed, hence, the first three best results, which are L2+H512, L3+H512 and L2+H256 are taken to do Train versus Validation plot for observation and is shown in Figure 4.22.

In Figure 4.22, the L2+H512 (2 GRU layers, 512 hidden size) graph shows extremely low training WER (Final Train WER: 0.00, Final Validation WER: 0.0044), and validation WER

also drops significantly. There is a very small gap between training and validation performance, indicating very little to no overfitting and well generalized. The L3+H512 (3 GRU layers, 512 hidden size) shows slight increase in final training WER compared to validation WER. There is a small performance gap, but both values are low (Final Train WER: 0.0424, Final Validation WER: 0.0110). The model also had some fluctuation (e.g., Epoch 19 WER jumped), showing slight signs of instability but no strong overfitting. Lastly, the L2+H256 (2 GRU layers, 256 hidden size) shows larger gap between training and validation WER and slower convergence. This may indicate there are some overfitting and weaker generalization.

The experiment is then continued by using the best configuration, L2+H512 with its consistently decreasing WER across epochs and minimal gap between training and validation WER.

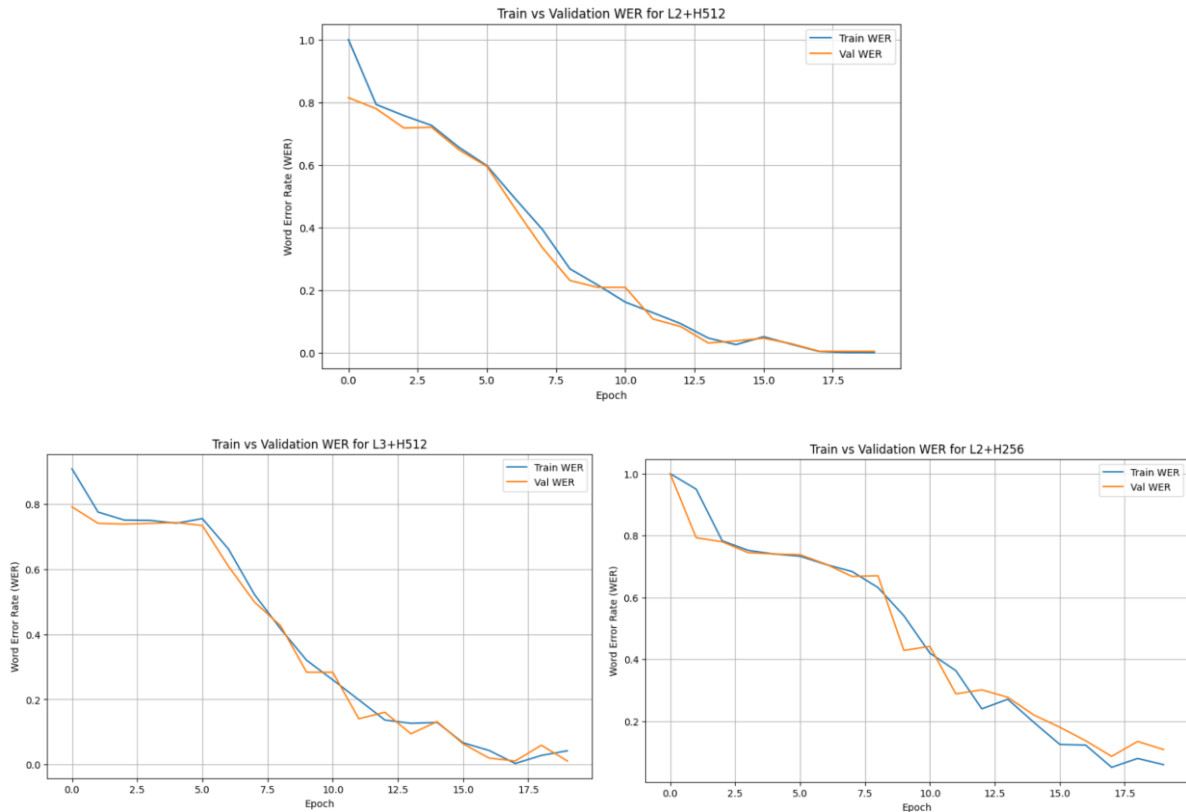


Figure 4.22. Train VS Validation WER with Different Parameters Configuration

4.5.2 Model Validation and Testing

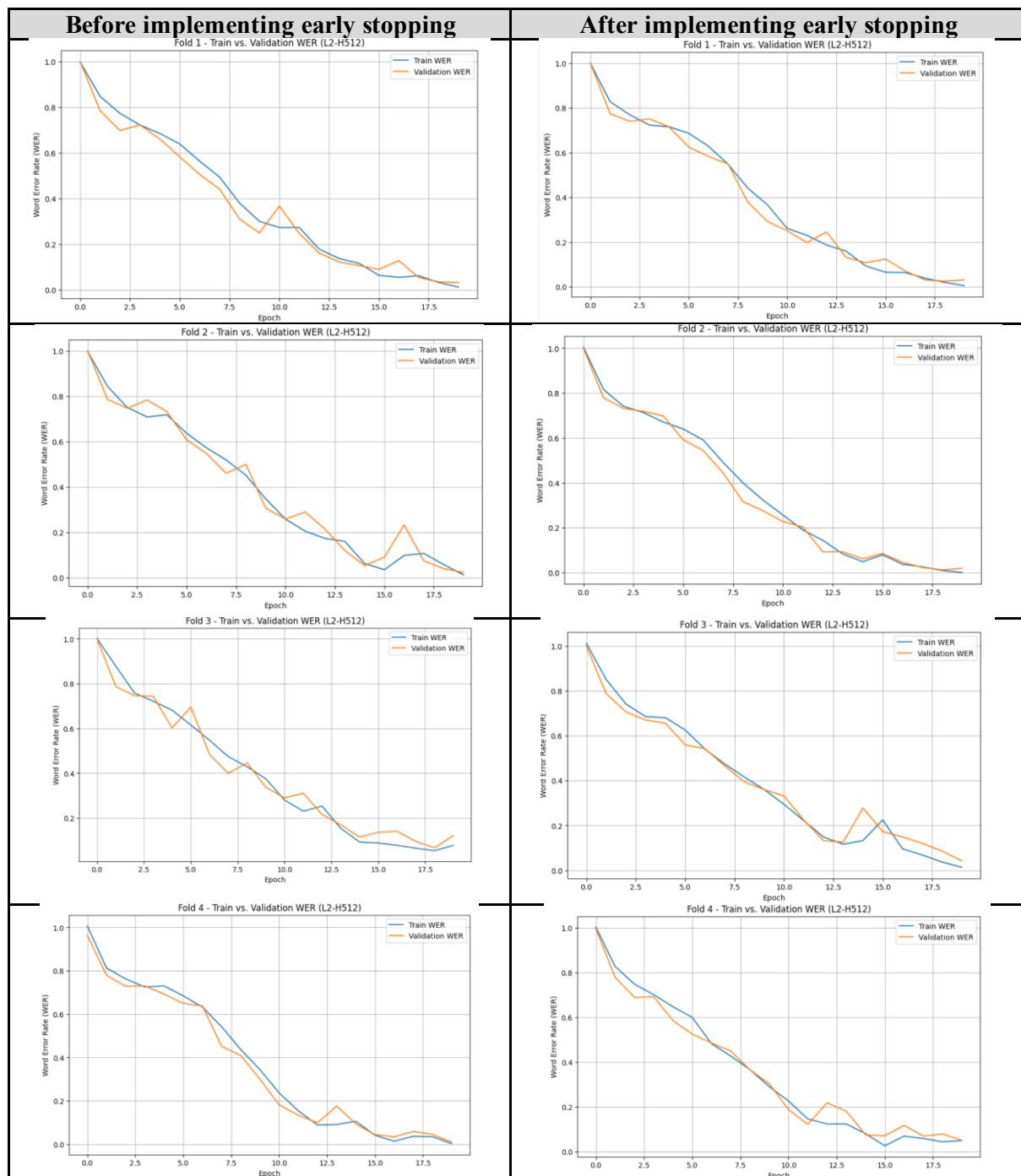
To evaluate the performance and generalization ability of sign language recognition model, k-fold cross-validation is implemented. During initial cross-validation experiments, a common trend was observed across several fold: while the training WER continued to decrease, the validation WER tended to plateau or slightly increase in the later epochs. This suggested that the model was beginning to overfit the training data, learning noise and specific patterns that did not generalize well to the unseen validation data within those folds.

To address the issue of overfitting observed during the initial cross-validation runs, early stopping is implemented as a regularization technique in training procedure. Early stopping is designed to halt the training process at the point where the model's performance on a validation set starts to degrade, even if the training performance is still improving.

“EARLY_STOPPING_PATIENCE” value was set to 5 epochs. This means that after each epoch, the validation WER is monitored. If the validation WER did not improve (i.e., it was not lower than the best validation WER seen so far) for a consecutive number of epochs equal to the patience value, the training process for that specific fold was terminated early. The model state corresponding to the best validation WER achieved during training was saved and used for evaluation. Table 6 illustrates this trend: while the training WER approaches near-zero in later epochs within certain folds, the corresponding validation WERs remain low and do not exhibit a dramatic increase, which indicates that the model is learning generalizable features rather than merely memorizing the training data.

Table 7 shows the 5-fold cross-validation result. The 5-fold cross-validation yielded a promising average best validation Word Error Rate (WER) of 0.0275, with a standard deviation of 0.0173. This low average WER suggests that the model demonstrates a strong ability to generalize to unseen data within the training distribution. The relatively small standard

deviation indicates that the model's performance is consistent across different splits of the training data thereby enhancing confidence in its robustness. The best validation performance was generally observed in the later stages of training (around epochs 19-20), indicating that the model effectively learned the underlying patterns in the sign language data over the course of training, and the early stopping mechanism prevented significant overfitting beyond this point.



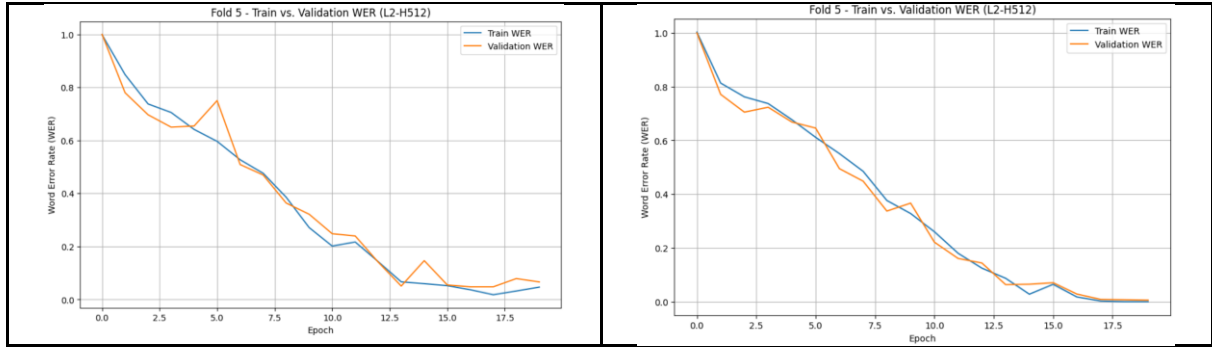


Table 6. Comparison Between the Result of Using Early Stopping

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average	Standard Deviation
Best Validation WER	0.0261	0.0121	0.0425	0.0510	0.0056	0.0275	0.0173
Epoch at Best Val WER	19	19	20	20	20	19.6	0.89

Table 7. 5-fold cross-validation result

Based on the favourable results obtained from the k-fold cross-validation with early stopping, final training proceeded with the entire training dataset. The same model architecture (GRUCTCModel2 with L2-H512), the same training hyperparameters (learning rate, optimizer, batch size), and the same early stopping strategy with a patience of 5 were utilised.

For this final training run, test dataset was used as the validation set for the early stopping mechanism. This approach is crucial as it ensures the final model's training is guided by its performance on truly unseen data (the test set), thereby providing a more reliable indication of its generalization ability. The final test result showed a Word Error Rate of 0.0132 and is shown in Figure 4.23. Furthermore, the achieved test WER is consistent with, or represents an improvement upon, the average best validation WER of 0.0275 obtained during the k-fold cross-validation procedure, indicating robust generalization capabilities. Analysis of the training and validation (test) WER curves reveals a concurrent decreasing trend, supporting the model's ability to effectively generalize beyond the training corpus.

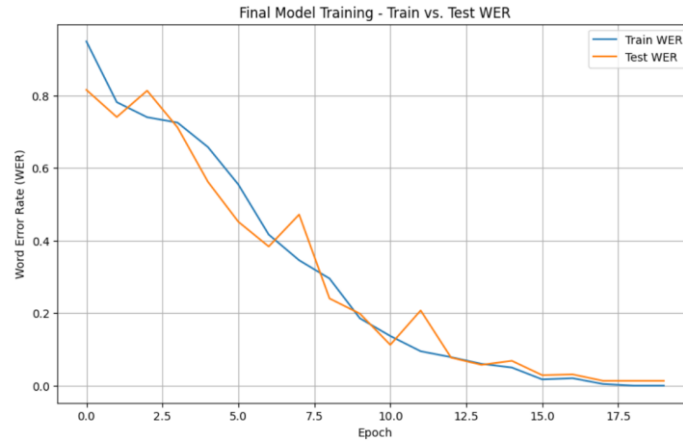


Figure 4.23. Final Model Training - Train vs. Test WER

4.5.3 Comparative Performance Analysis

During real-time testing, to investigate the performance and evaluate the model's consistency, user independent testing was conducted. Three signers from SSD were utilised and the dataset from them were not trained in the model. The three signers perform sign language in different indoor environments with five iterations for each sentence. Figure 4.24 show the testing with one of the signers and Table 8 details the accuracy for individual sentences. Sentence 2, 'saya muntah dan penat kerana mengandung,' exhibited the lowest accuracy, frequently being confused with Sentence 3 ('Saya rasa batuk dan sakit tekak'), Sentence 4 ('Saya membuat pembedahan untuk mata'), and Sentence 5 ('badan saya berasa gatal-gatal'). This may be attributed to the visual similarities in the signed actions for 'batuk' and 'pembedahan' to 'muntah,' particularly the hand placement near the center of the throat with upright, pinched fingers. Additionally, the sign for 'penat' (tired) was often confused with 'badan' (body), both involving similar two-hand gestures touching the shoulders.

Sentences 1 and 4 each achieved an average accuracy of 66.67%, primarily due to confusion with a single distinct incorrect sentence. Specifically, Sentence 1 occasionally showed confusion with the word 'mahu' (want), whose sign shares a similar hand placement to 'badan,' with both hands touching the shoulders. Despite these confusions, both sentences exhibited a consistent trend: accuracy improved when key words such as 'dokter,' 'x-ray,'

‘pembedahan,’ and ‘mata’ were signed at a slower pace and sustained longer, facilitating better gesture capture. Finally, Sentences 3 and 5 demonstrated comparatively greater stability, potentially due to their more expansive and distinct hand movements.

The real-time testing phase highlighted the importance of optimal beam width and window size configurations, with the beam width 5 and window size 50 setting yielding the highest accuracy of 80%. While the model demonstrated strong performance for certain sentences, particularly those with more distinct hand movements (Sentences 3 and 5), challenges arose from the inherent visual similarities between specific signs. The confusion observed, especially between 'muntah,' 'batuk,' and 'pembedahan', and between 'penat' and 'badan', indicates areas where further model training or feature engineering might be beneficial.

Moreover, there is a significant performance gap arises from critical differences in data quality, environmental conditions, and user variability between the controlled offline environment and real-time application. Real-time testing, conducted with data from users whose signing styles, hand shapes, and motion speeds were not encountered during training, highlighted a key challenge in model generalization. The model proved sensitive to these unforeseen variations. Furthermore, real-world usage introduced noise into the input sequences from factors such as camera frame drops, motion blur, background clutter, lighting variations, and occasional inaccuracies in MediaPipe's landmark detection. These factors directly affected the consistency and completeness of the hand landmarks, frequently leading to misclassifications by the model. Lastly, a significant limitation was the model's training on a relatively small set of predefined sentences, constrained by both time and dataset availability. This inherent constraint limits the system's current flexibility to handle spontaneous or more extensive BIM vocabulary in practical, real-world communication scenario.

Sentence	SSD Signer	Predicted Sentences per Iteration	Word Error Rate (WER)	Average WER	Average Accuracy
Doktor mahu saya pergi buat x-ray	1	doktor mahu saya pergi buat x-ray	0	0.278	66.67%
		badan saya berasa gatal-gatal	1		
		doktor mahu saya pergi buat gatal-gatal	0.17		
		doktor mahu saya pergi buat x-ray	0		
		doktor mahu saya pergi buat x-ray	0		
	2	doktor mahu saya pergi buat x-ray	0		
		badan saya berasa gatal-gatal	1		
		doktor mahu saya pergi buat x-ray	0		
		doktor mahu saya pergi buat x-ray	0		
		doktor mahu saya pergi buat x-ray	0		
	3	badan saya berasa gatal-gatal	1		
		badan saya berasa gatal-gatal	1		
		doktor mahu saya pergi buat x-ray	0		
		doktor mahu saya pergi buat x-ray	0		
		doktor mahu saya pergi buat x-ray	0		
Saya muntah dan penat kerana mengandung	1	saya muntah dan penat kerana mengandung	0	0.40	60%
		saya rasa batuk dan sakit tekak	1		
		saya muntah dan penat kerana mengandung	0		
		badan saya berasa gatal-gatal	1		
		saya muntah dan penat kerana mengandung	0		
	2	badan saya berasa gatal-gatal	1		
		saya muntah dan penat kerana mengandung	0		
		saya membuat pembedahan untuk mata	1		
		saya muntah dan penat kerana mengandung	0		
		saya muntah dan penat kerana mengandung	0		
	3	saya muntah dan penat kerana mengandung	0		
		saya rasa batuk dan sakit tekak	1		
		badan saya berasa gatal-gatal	1		
		saya muntah dan penat kerana mengandung	0		
		saya muntah dan penat kerana mengandung	0		
Saya rasa batuk dan sakit tekak	1	saya rasa batuk dan sakit tekak	0	0.20	80%
		saya muntah dan penat kerana mengandung	1		
		saya rasa batuk dan sakit tekak	0		
		saya rasa batuk dan sakit tekak	0		
		saya rasa batuk dan sakit tekak	0		
	2	saya rasa batuk dan sakit tekak	0		
		saya rasa batuk dan sakit tekak	0		
		saya rasa batuk dan sakit tekak	0		
		saya muntah dan penat kerana mengandung	1		
		saya rasa batuk dan sakit tekak	0		
	3	saya rasa batuk dan sakit tekak	0		
		saya rasa batuk dan sakit tekak	0		
		saya membuat pembedahan untuk mata	1		
		saya rasa batuk dan sakit tekak	0		
		saya rasa batuk dan sakit tekak	0		

		saya rasa batuk dan sakit tekak	0		
Saya membuat pembedahan untuk mata	1	badan saya berasa gatal-gatal	1	0.33	66.67%
		saya membuat pembedahan untuk mata	0		
		saya membuat pembedahan untuk mata	0		
		badan saya berasa gatal-gatal	1		
		saya membuat pembedahan untuk mata	0		
	2	badan saya berasa gatal-gatal	1		
		saya membuat pembedahan untuk mata	0		
		badan saya berasa gatal-gatal	1		
		saya membuat pembedahan untuk mata	0		
		saya membuat pembedahan untuk mata	0		
	3	badan saya berasa gatal-gatal	1		
		saya membuat pembedahan untuk mata	0		
		saya membuat pembedahan untuk mata	0		
		saya membuat pembedahan untuk mata	0		
		saya membuat pembedahan untuk mata	0		
Badan saya berasa gatal-gatal	1	badan saya berasa gatal-gatal	0	0.27	73.33%
		saya membuat pembedahan untuk mata	1		
		badan saya berasa gatal-gatal	0		
		badan saya berasa gatal-gatal	0		
		badan saya berasa gatal-gatal	0		
	2	saya membuat pembedahan untuk mata	1		
		badan saya berasa gatal-gatal	0		
		badan saya berasa gatal-gatal	0		
		saya membuat pembedahan untuk mata	1		
		badan saya berasa gatal-gatal	0		
	3	saya membuat pembedahan untuk mata	1		
		badan saya berasa gatal-gatal	0		
		badan saya berasa gatal-gatal	0		
		badan saya berasa gatal-gatal	0		
		badan saya berasa gatal-gatal	0		
Overall Average				0.30	69.33%

Table 8. Accuracy and Prediction Iterations for Specific Sign Language Sentence

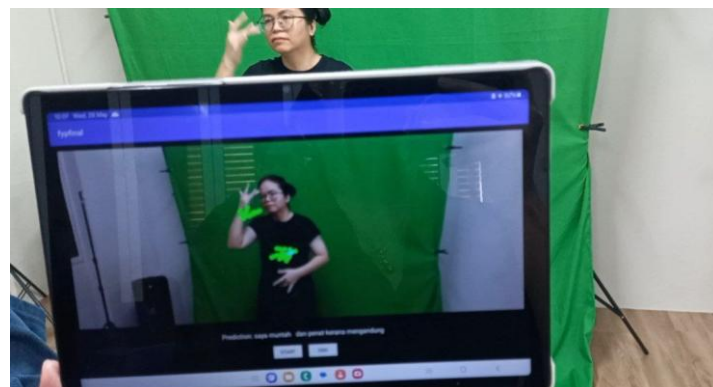


Figure 4.24. User-independent testing with one of the signers

4.6 Summary

This multi-stage model development and integration enables real-time, on-device sign language translation with feedback, leveraging MediaPipe efficient landmark detection and PyTorch lightweight inference on mobile. The Android app combines machine learning, real-time processing, and interactive UI to deliver an accessible translation tool for BIM users.

The developed real-time Malaysian Sign Language (BIM) recognition system successfully demonstrated sentence-level recognition, achieving promising results under controlled conditions. During offline validation, the system attained a remarkably low Word Error Rate (WER) of 0.0275, indicating strong model accuracy on pre-processed and controlled input sequences. However, a notable performance disparity was observed in real-time testing with unseen users, where the system achieved an accuracy of only 69%, highlighting a key challenge in model generalization and sensitive to unforeseen variations.

Chapter 5. Conclusions and Future Work

5.1 Limitations

This project faced several limitations that impacted the overall accuracy and generalization of the model. Firstly, is the limited dataset diversity, because each signer performs the same sentence differently. Due to time and resource constraints, the dataset could not capture the full range of variations in hand shapes, motions, and transition patterns. Furthermore, with the nature of inter-individual variability, the accuracy of the model decreased significantly when exposed to signers not present in the training set, indicating a lack of robustness to diverse signing styles.

Next, the model's performance was influenced by environment sensitivity such as different signing speeds, lighting conditions, and distances between the user and the camera. These variations introduced inconsistencies in landmark detection. Lastly, mobile hardware constraints is also part of the challenge. Real-time performance was affected by the processing power, lighting adaptation, and power efficiency of mobile devices, which are typically resource-constrained compared to desktop environments.

5.2 Future Work

Several improvements can be made to enhance the performance, usability, and scalability of the sign language recognition system. Firstly, is the Natural Language Processing (NLP) Integration. By incorporating NLP techniques, it can help generate more natural and grammatically correct sentences from recognized gesture sequences, improving the fluency of the translated output. Further work is needed to optimize the model for energy efficiency and real-time responsiveness on a wide range of Android devices with varying hardware capabilities.

Next, a user interface can be developed to allow users to contribute sign language data directly to the developers. This community involvement will help expand the dataset with diverse signing styles and scenarios. Moreover, recruiting more participants from various demographic backgrounds and including multiple repetitions across environments will improve the model's generalization capabilities.

5.3 Conclusions

A real-time Malaysian Sign Language (BIM) recognition system was developed using MediaPipe for hand landmark detection and a GRU-based deep learning model with Connectionist Temporal Classification (CTC) for sequence prediction. The system was successfully integrated into an Android mobile application, enabling sentence-level real-time recognition of a limited set of predefined BIM sentences. Evaluation was conducted through both offline validation and real-time testing. During 5-fold cross-validation, the model achieved an average Word Error Rate (WER) of 0.0275 with a standard deviation of 0.0173, indicating consistent performance across different data splits. A final test WER of 0.0132 was obtained after full-dataset training and tested with test set, further confirming the model's strong generalization capability. In real-time testing with unseen users, the system reached 69% accuracy and average WER of 0.30, demonstrating acceptable user-independent performance.

Despite promising results, the model's accuracy was constrained by the limited diversity and volume of the dataset. Inter-individual variability in signing style, speed, and articulation significantly affected model performance. Additionally, real-world factors such as lighting, distance from the camera, and hardware limitations posed challenges to consistent gesture recognition across devices. It demonstrates the feasibility of deploying a real-time sign language recognition system on mobile platforms, highlighting the potential of deep learning and computer vision in improving communication accessibility for the deaf and hard-of-hearing communities.

References

- Akandeh, Atra. (2022). Sentence-Level Sign Language Recognition Framework. *arXiv preprint arXiv:2211.14447*
- Al Abdullah, B. A., Amoudi, G. A., Alghamdi, H. S. (2024). Advancements in Sign Language Recognition: A Comprehensive Review and Future Prospects. *IEEE Access*, vol. 12, pp. 128871-128895. doi: 10.1109/ACCESS.2024.3457692
- Arkuuu21. (2023). *MSL Medical Segmentation and Classification* [Data set]. Kaggle. Retrieved May 9, 2025, from <https://www.kaggle.com/datasets/arkuuu21/msl-medical>
- Bhusal D. & Baral, D. S. (2024). Sentence-Level Nepali Sign Language Recognition. *2024 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2024*, pp. 972-979. doi: 10.1109/ICICT60155.2024.10544519
- BIM Sign Bank. (n.d.). Bim signbank - bahasa ISYARAT Malaysia / malaysian sign language. <https://www.bimsignbank.org/about-us>
- Chand, N., Sengupta, A., & Bhattacharya, S. (2024). Vivo's sign language interpreter function revolutionizes language accessibility. Techgenyz, <https://techgenyz.com/vivos-sign-language-interpreter/>
- Faithour. (2022). *Malaysia is a country with many spoken languages, but there is still a long way to go for Malaysian sign language (BIM)*. <https://www.faithour.com/bim-deaf/>
- Google. (n.d.). *MediaPipe Hand Landmarker*. Google AI for Developers. Retrieved April 30, 2025, from https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker
- Gu, Y., Oku, H., & Todoh, M. (2024). American Sign Language Recognition and Translation Using Perception Neuron Wearable Inertial Motion Capture System. *Sensors*, 24(2), 453. <https://doi.org/10.3390/s24020453>

- Hannun, A. (2017). Sequence modeling with CTC. Distill. <https://doi.org/10.23915/distill.00008>
- Hou, J., Li, X.-Y., Zhu, P., Wang, Z., Wang, Y., Qian, J., & Yang, P. (2019). Signspeaker. *The 25th Annual International Conference on Mobile Computing and Networking*, 1–15. <https://doi.org/10.1145/3300061.3300117>
- Johari, R. T., Ramli, R., Zulkoffli, Z., & Saibani, N. (2023). MYWSL: Malaysian words sign language dataset. *Data in Brief*, vol. 49, p. 109338. doi:10.1016/j.dib.2023.109338
- Ke, S.C., Mahamad, A.K., Saon, S., Fadlilah, U., Handaga, B. (2022). Malaysian Sign Language Translator for Mobile Application. *In: Mahyuddin, N.M., Mat Noor, N.R., Mat Sakim, H.A. (eds) Proceedings of the 11th International Conference on Robotics, Vision, Signal Processing and Power Applications. Lecture Notes in Electrical Engineering, vol 829. Springer, Singapore.* https://doi-org.remotexs.unimas.my/10.1007/978-981-16-8129-5_139
- Khan R. U. et al. (2021). Intelligent Malaysian sign language translation system using convolutional-based attention module with residual network. *Computational Intelligence and Neuroscience*, vol. 2021, no. 1. doi:10.1155/2021/9023010
- Luqman, H. & El-Alfy, E.-S. M. (2021). Towards hybrid multimodal manual and non-manual Arabic sign language recognition: Marsl database and pilot study. *Electronics*, vol. 10, no. 14, p. 1739. doi:10.3390/electronics10141739
- Maheswara, Y. D., Al-Sulthon, M. A., Wicaksono, P. A., Afifah, K., & Prihatiningrum, N. (2023). Real-time bisindo sign language recognition: A dynamic approach with GRU and LSTM models leveraging MediaPipe. *2023 6th International Seminar on Research*

- of Information Technology and Intelligent Systems (ISRITI)*, 226–232.
<https://doi.org/10.1109/isriti60336.2023.10467586>
- Malaysian Wireless. (2017). Digi: Learn basic Malaysian sign language with Eddy.
<https://www.malaysianwireless.com/2017/12/digi-malaysian-sign-language-eddy/>
- Matplotlib. (n.d.). *Matplotlib*. Retrieved April 30, 2025, from <https://matplotlib.org/>
- Meng, X. et al. (2019). Sentence-Level Sign Language Recognition Using RF signals. *2019 6th International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC), Beijing, China, 2019*, pp. 1-6. doi: 10.1109/BESC48373.2019.8963177
- Monash University Malaysia. (2023). *Health Equity Project for malaysian deaf community wins nature inclusive health research award – Malaysia*.
<https://www.monash.edu.my/pages/latest/articles/2023/health-equity-project-for-malaysian-deaf-community-wins-nature-inclusive-health-research-award>
- Nam, C. (2024). Sign Language Translator AI [Mobile app]. Google Play Store.
<https://play.google.com/store/apps/details?id=com.siyaroll.sltranslator&hl=en>
- NumPy. (n.d.). *NumPy*. Retrieved April 30, 2025, from <https://numpy.org/doc/stable/>
- OpenCV. (n.d.). *OpenCV*. Retrieved April 30, 2025, from <https://opencv.org/>
- Padmanand, K., & Lim, P. -C. (2022). Malaysian Sign Language Recognition Using 3D Hand Pose Estimation. *2022 International Conference on Digital Transformation and Intelligence (ICDI), Kuching, Sarawak, Malaysia, 2022*, pp. 214-218. doi: 10.1109/ICDI57181.2022.10007170
- Paul Kelly @terpai. (2023). *Sign Language AI* [Mobile app]. Google Play Store.
<https://play.google.com/store/apps/details?id=ai.terp.www.twa&hl=en>

- Prince, N. R., Thayil, R. O., Anto Praveena, M. D., & Prince Mary, S. (2023). Makaton Sign Language recognition. *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)*, 360–363. <https://doi.org/10.1109/iccpct58313.2023.10245903>
- Python Software Foundation. (n.d.). *Python documentation: General Python FAQ*. Retrieved April 18, 2025, from <https://docs.python.org/3/faq/general.html#what-is-python>
- Quzaimi, M. A., Ramli, A. F., & Basarudin, H. (2023). Malaysian sign language alphabet recognition through hand landmarks feature and machine learning. *2023 International Conference on Engineering Technology and Technopreneurship (ICE2T)* (pp. 183-186). IEEE. <https://doi.org/10.1109/ICE2T58637.2023.10540543>
- Saiful Bahri, I. Z., Saon, S., Mahamad, A. K., Isa, K., Fadlilah, U., Ahmadon, M. A. B., & Yamaguchi, S. (2023). Interpretation of Bahasa Isyarat Malaysia (BIM) Using SSD-MobileNet-V2 FPNLite and COCO mAP. *Information*, 14(6), 319. <https://doi.org/10.3390/info14060319>
- Salvador, S., & Chan, P. (2007). *FastDTW: Toward accurate dynamic time warping in linear time and space*. Google Code Archive. Retrieved April 30, 2025, from <https://code.google.com/archive/p/fastdtw/>
- Subramanian, B., Olimov, B., Naik, S. M., Kim, S., Park, K. H., & Kim, J. (2022). An integrated mediapipe-optimized GRU model for Indian sign language recognition. *Scientific reports*, 12(1), 11964. <https://doi.org/10.1038/s41598-022-15998-7>
- The Jupyter Project. (n.d.). *JupyterLab documentation*. Retrieved April 18, 2025, from <https://jupyterlab.readthedocs.io/en/latest/>

- Tran, K. B., Nguyen, U. D., & Huynh, Q. T. (2023). Continuous sign language recognition using MediaPipe. *2023 International Conference on Advanced Technologies for Communications (ATC)*, 493–498. <https://doi.org/10.1109/atc58710.2023.10318855>
- Tupal, I., Cabatuan, M., & Manguerra, M. (2022). Recognizing Filipino sign language with INCEPTIONV3, LSTM, and gru. *2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 1–5. <https://doi.org/10.1109/hnicem57413.2022.10109284>
- United Nations. (2017). *International Day of Sign Languages*. <https://www.un.org/en/observances/sign-languages-day>
- Veron, Z. L. H., Aaron, K. K. L., Ida, P. X. L., Alec, V. G. A., Lee, S. H., Patrick, H. H. T. (2025). Two-Way Malaysian Sign Language Communication System for Inclusive Education. *2024 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*.
- Vivo Software. (2024). 手语翻译官 [Mobile app]. V-Appstore. <https://accessibility.vivo.com/accessibility/translate>
- Wang, Y., Li, F., Xie, Y., Duan, C., & Wang, Y. (2023). HearASL: Your smartphone can hear American Sign Language. *IEEE Internet of Things Journal*, 10(10), 8839–8852. <https://doi.org/10.1109/JIOT.2022.3232337>
- Wolfe, R. et al. (2023). Special issue on sign language translation and avatar technology. *Universal Access in the Information Society*. doi:10.1007/s10209-023-01014-w

Xi, L. G. (2023. November, 1). Gesture of goodwill to bridge the gap. *The Star*.
<https://www.thestar.com.my/metro/metro-news/2023/11/01/gesture-of-goodwill-to-bridge-the-gap>

回形针 PaperClip. (Jan 26, 2021). *Vol.154 手语的秘密*. [Video]. YouTube.
<https://www.youtube.com/watch?v=vpkHTLhuDqg&t=44s>