

**REAL-TIME HAND GESTURE RECOGNITION SYSTEM TO
INTERPRET SIGN LANGUAGE**

CHONG SIOW YEN

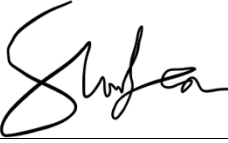
A project report submitted in fulfilment of the
requirements for the award of Bachelor of Science (Honours) Software
Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

June 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Chong Siow Yen

ID No. : 19UEB03392

Date : 9 February 2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**REAL-TIME HAND GESTURE RECOGNITION SYSTEM TO INTERPRET SIGN LANGUAGE**” was prepared by **CHONG SIOW YEN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr. Chia Kai Lin

Date : 9 February 2023

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, Chong Siow Yen. All right reserved.

ACKNOWLEDGEMENTS

I would like to express my gratitude towards Dr. Chia Kai Lin who suggests the title, *Real-Time Hand Gesture Recognition System To Interpret Sign Language*, which would potentially benefit the communication between people with disabilities (deaf/hard of hearing) and the non-disabled person. She supervises me throughout my Project and provides useful suggestions to enhance the report as well as in image processing.

Secondly, I would like to thank the moderator, Dr. Lee Ming Jie for being active in the Microsoft Teams and always there to answer my inquiries on the Project. Next, I appreciate Dr. Faranak Nejati, my moderator for her useful advice during presentation of *PROJECT I* and *II*, as well as Dr Wong Whee Yen, the lecturer for the course, *UECS3599 PROJECT I*, in providing explanations of each part of the Project which aids me in understanding better on what to expect for each chapter. Finally, I would like to thank my parents who are huge sponsors of me finishing the studies and loved ones for supporting and motivating me this far.

ABSTRACT

Sign Language plays a major part in communications among individuals with hearing impairments or hearing-challenged individuals, a medium which allows them to participate in society. Through sign language, it allows them to get access to real-world news and access important information. Its significance can be proven when sign language interpreters appear on screen alongside the news anchor in our daily news broadcastings, with the addition of closed captions.

The central component of sign language is the hand gestures. It is used to communicate words, phrases, and ideas, with each gesture having a specific meaning. Hand gestures in sign language can be divided into two main gestures, static gestures which do not involve movement, such as the alphabet-fingering or thumbs up gesture. while dynamic gestures are hand movements which involve changing of hand shape or position, such as waving, pointing and more. The applications of hand gestures in sign language include in communication, education, interpretation, accessibility in the form of visual format, and cultural events to translate lyrics, plays and musicals to sign language that is easily understood by individuals with hearing impairments or have difficulty in hearing.

The current phenomenon reflects that most hearing people are unaware of sign language and do not take the time to learn it, leading to miscommunication and poor understanding between the hearing and the deaf community. They also view sign language as less of importance compared to spoken language. It is expected that with the existence of a real-time sign language interpretation tool would educate these ignorant people and increase their acceptance towards the individuals with hearing impairments or hearing-challenged individuals' culture.

The building of this System To Interpret Sign Language aims to bridge the communication gap between the hearing and individuals with hearing impairments or hearing-challenged individuals. It shall act as a complement rather than a replacement to the existing sign language interpreter to enhance the communication and understanding among the communities. To achieve that, a tool will be developed to ease communication between the individuals with hearing impairments or hearing-challenged individuals. The sensors and cameras will detect the hand gestures, then translate the hand gestures to captions. There are several studies on sign language recognition systems, which built with techniques, which includes OpenCV, OpenCV with Mediapipe, LeapMotion Controller (LMC) with training of Convolutional Neural Network (CNN), which will be discussed further in the literature review.

The project will follow five major steps: pre-processing, feature extraction, segmentation and dimension reduction, classification, and model evaluation. The system is built with OpenCV Mediapipe in addition of Neural Network model with TensorFlow and Keras API. The results demonstrate that the system can detect 10 alphabets (A, E, H, I, L, N, O, S, T, U) and 7 vocabulary words (Best, Birthday, Please, Happy, Hearing, Like, Feel), with accuracies of 96% and 53% respectively. This project serves as a valuable tool in fostering communication and understanding between these communities.

TABLE OF CONTENTS

REAL-TIME HAND GESTURE RECOGNITION SYSTEM TO INTERPRET SIGN LANGUAGE	1
DECLARATION	2
APPROVAL FOR SUBMISSION	3
© 2023, Chong Siow Yen. All right reserved.	4
ACKNOWLEDGEMENTS	5
ABSTRACT	6
TABLE OF CONTENTS	8
LIST OF TABLES	13
LIST OF FIGURES	14
CHAPTER 1	18
1. INTRODUCTION	18
1.1. General Introduction	18
1.2. Importance of the Study	20
1.3. Problem Statement	21
1. Dependency on trained interpreters, which may not always be available or accessible to the deaf and hard of hearing community.	21
2. Communication gap between individuals with and without hearing impairments in real-time.	21
3. Inadequacy of the existing systems to work with variability in sign language	22
1.4. Aim	22
1.5. Objective	22
1. To design and develop an efficient algorithm for real-time hand gesture recognition that can detect and interpret sign language.	22
2. To evaluate the efficiency of the hand gesture recognition system in real-time communication scenarios.	22

3.	To identify potential challenges in the adoption of the system and develop solutions to address them.	23
1.6.	Research Questions	23
1.7.	Proposed Solution	24
	Figure 1.6.1: Workflow Diagram for the System for recognizing ASL hand gestures	24
	Figure 1.6.2: Model-View-Controller Architecture Pattern in context of system	27
1.8.	Scope and Limitations	30
	Table 1.7.1: Hardware Setup	31
	Table 1.7.2. Software Setup	32
1.9.	Summary	35
	CHAPTER 2	36
	LITERATURE REVIEW	36
2.1	Introduction	36
2.2	Literature Review	37
2.3	OpenCV	39
	2.3.1 Pre-processing	39
	2.3.2 Feature Extraction	40
	2.3.3 Classification	41
	2.3.4 Accuracy	42
	2.3.5 Advantages	42
	2.3.6 Disadvantages	43
2.4	OpenCV with Mediapipe	44
	2.4.1 Pre-processing	45
	2.4.2 Feature Extraction	46
	2.4.3 Segmentation	46
	2.4.4 Classification	47
	2.4.5 Accuracy	48
	2.4.6 Advantages	50
	2.4.7 Disadvantages	50
2.5	LeapMotion Controller (LMC)	51

		10
	2.5.1 Pre-processing	53
	2.5.2 Segmentation	54
	2.5.3 Feature Extraction	55
	2.5.4 Classification	56
	2.5.5 Deep Learning Model / Traditional Machine Learning Model	57
	2.5.6 Accuracy	58
	2.5.7 Complexity	59
	2.5.8 Advantages	59
	2.5.9 Disadvantages	60
2.6	Gloves with sensors	61
	2.6.1 Pre-processing	64
	2.6.2 Segmentation	64
	2.6.3 Feature Extraction	65
	2.6.4 Dimension Reduction	65
	2.6.5 Classification	65
	2.6.6 Deep Learning Model / Traditional Machine Learning Model	66
	2.6.7 Accuracy	66
	2.6.8 Complexity	67
	2.6.9 Advantages	67
	2.6.10 Disadvantages	68
2.7	Skeleton Aware Multimodal SLR framework (SAM-SLR)	69
	2.7.1 Pre-processing	70
	2.7.2 Segmentation	72
	2.7.3 Feature Extraction	72
	2.7.4 Classification	73
	2.7.5 Deep Learning Model / Traditional Machine Learning Model	74
	2.7.6 Accuracy	74
	2.7.7 Complexity	75
	2.7.8 Advantages	77

		11
	2.7.9 Disadvantages	77
	2.8 Summary	78
	CHAPTER 3	91
	METHODOLOGY AND WORK PLAN	91
	3.1 The Proposed Model Workflow	91
	3.2 Pre-processing	93
	3.3 Feature Extraction	96
	3.4 Segmentation and Dimension Reduction	99
	3.5 Classification	101
	3.6 Model Evaluation	104
	3.7 Requirement Specifications	110
	3.8 Tools to use	110
	3.9 Pseudocode for this project	111
	3.10 Work Breakdown Structure of the Project	113
	3.11 Gantt Chart of Project	114
	CHAPTER 4	117
4	SYSTEM PERFORMANCE	117
	4.1 Keypoint classifier detection	117
	4.1.1 Training and Validation Metrics	117
	4.1.2 Confusion Matrix	121
	4.1.3 Classification Report	122
	4.2 Point History Classifier Detection	123
	4.2.1 Training and Validation Metrics	123
	4.2.2 Confusion Matrix	126
	4.2.3 Classification Report	127
	4.3 Comparing OpenCV with Mediapipe and Neural Network with other methodologies in Literature Review (refer Table 2.8)	128
4	TESTING RESULTS	130
	4.1 Introduction	130
	4.1.1 Approach	130
	4.1.2 Expected Achievement	137
	4.2 Test Setup and Data Collection	138

		12
	4.2.1 Test Environment Setup	138
	4.2.2 Gestures Used for Testing	140
	4.2.3 Tester Recruitment and Data Collection	140
4.3	Evaluation and Tester Feedback	142
	a. For testers	142
	b. For self	153
	CHAPTER 5	155
4	CONCLUSIONS AND RECOMMENDATIONS	155
	5.1 Conclusions	155
	5.2 Recommendations for future work	155
	1. Enhance Point History Classifier	155
	2. Gesture Recognition Speed	156
	3. Robustness Across User Positions	156
	4. Complex Gesture Recognition	156
	5. Feedback Integration	156
	REFERENCES	157

LIST OF TABLES

Table 1.7.1:	Hardware Setup	31
Table 1.7.2:	Software Setup	32
Table 2.3.2:	Analysis table for Region of Interest	40
Table 2.3.4:	Analysis table for gesture recognition	42
Table 2.4.5.1:	Precision, F1-score, and Recall were computed for all models during the ML model development and testing	49
Table 2.4.5.2:	Model and corresponding accuracies for 10 class NUS dataset with Training Accuracy (Tr. A), Testing Accuracy (Te. A), Overall Accuracy (O.A), Average Inference Time (A.I.T)	49
Table 2.5.6:	Mean classification accuracy with RGB, LeapMotion and Multi-modality model	58
Table 2.6.7:	Classification accuracy 1st and 2nd version of sign recognition and the system	66
Table 2.7.6:	Performance of multi-stream SL-GCN	74
Table 2.7.7:	Performance baseline results RGB and RGB-D	74
Table 2.8:	Overview of different approaches to hand gesture recognition	78-89
Table 3.6.1:	Confusion Matrix for 10 classes (letters)	105-107
Table 3.6.2:	Confusion Matrix for 7 classes (vocabularies)	108
Table 3.7:	Requirement Specifications	110
Table 4.3.	Accuracy Comparison Literature Review and Current Development	128-129
Table 4.1.1a:	Testers' Profile and Feedback on System	131
Table 4.1.1b:	Static Gestures with contrasting hand signing	133-135
Table 4.1.1b1:	Moving Gestures with contrasting hand signing	135-137
Table 4.3:	Tabulation of Testers Attempt of Sign Language Recognition System	143-147
Table 4.3.1:	Feedback and Comments by Testers	152
Table 4.3.2:	Recognition state under different Lighting and Distances	153

LIST OF FIGURES

Figure 1.6.1:	Workflow Diagram for the Real-Time Hand Gesture Recognition System To Interpret Sign Language	24
Figure 1.6.2:	Model-View-Controller Architecture Pattern in context of system	27
Figure 2.3.1:	The convex hull (in green) and the hand contour (in white) in context of system	39
Figure 2.3.2:	Region of overlapping ROIs	40
Figure 2.3.3.1:	Image captured by camera	41
Figure 2.3.3.2:	Bounding web box without any gesture	41
Figure 2.3.3.3:	Recognition of hand showing number 1	41
Figure 2.3.5.1:	Right interpretation of sign language, 'Love'	43
Figure 2.3.5.2:	Wrong interpretation of sign language, 'Remember' supposed 'Love'	43
Figure 2.4:	The structure of the Sign Language Recognition system	44
Figure 2.4.1:	10 gestures in American Sign Language (ASL) corresponding to the alphabet's "A" to "J."	48
Figure 2.4.1.1:	Wrist and upwards as result of normalisation	45
Figure 2.4.2:	The working of HTM	46
Figure 2.4.4:	Real-time testing results of the gesture recognition System	48
Figure 2.4.5:	Confusion matrix of the model utilizing K-Nearest Neighbours (KNN)	48
Figure 2.5:	Leap Motion Controller (LMC)	51
Figure 2.5.1:	Overall networks and function involved in building the British Sign Language Recognition System	52
Figure 2.5.1.1:	RGB image data of 1s which collected at frequency 0.2s/frame (5 Hertz)	53
Figure 2.5.1.2:	Setup of LMC with 3D graphic display	53
Figure 2.5.2:	Features provided by LMC API	52
Figure 2.5.3:	Bone data of each finger detected by	55

LeapMotion sensor

Figure 2.5.4:	Input image is fed into a fine-tuned VGG16 CNN, where a layer comprising 128 ReLU neurons generates the output, which is then utilized in late fusion with the Leap Motion network (Naglot et al., n.d.)	Figure 2.5.4.1: Architecture of MLP neural network	56
Figure 2.5.4.1:	Architecture of MLP neural network		57
Figure 2.5.6:	Confusion Matrix of Predicted class vs Actual class of system trained using the Back Propagation (BP) algorithm		58
Figure 2.6.1:	Assembly of data-glove		61
Figure 2.6.2:	Flowchart of Sign Language Detection Algorithm		62
Figure 2.6.3:	Improved system with Pressure Sensors		63
Figure 2.6.4:	Overview of modules involved in sign interpretation system		63
Figure 2.6.5:	Normalization formula		66
Figure 2.6.6:	A tilt sensor at upright (vertical) position bent at 50 and a tilt sensor degree		66
Figure 2.6.7:	Classification accuracy 1st and 2nd of sign recognition		66
Figure 2.7:	Concept of Skeleton Aware Multi-modal Sign Language Recognition Framework (SAM-SLR)		69
Figure 2.7.1:	RGB with whole-body keypoints overlay		70
Figure 2.7.2:	Cropping based on hand keypoints by OpenPose may result in distorted crops in cases where certain keypoints are not detected		71
Figure 2.7.2.1:	The depicted samples exhibit differing lengths, with a median duration of approximately 61 frames, equivalent to about 2 seconds, within the training set		72
Figure 2.7.6:	Graph Accuracy against Parameters for 3 experiments: VTN, VTN-HC, VTN-PF		74

Figure 2.7.7:	Visualization of modalities: RGB frames, depth, masked HHA, optical flow and depth flow	75
Figure 2.7.7.1:	Multi-stream SL-GCN model	76
Figure 3.2.1:	Workflow Diagram for Pre-processing (Training)	93
Figure 3.2.2:	Workflow Diagram for Pre-processing (Test)	93
Figure 3.3.1:	Workflow Diagram for Feature Extraction	96
Figure 3.3.2:	Convolutional Neural Network (CNN) in hand landmark identification	97
Figure 3.3.3:	Finalized extracted hand landmarks	98
Figure 3.4.1:	Workflow Diagram for Segmentation and Dimension Reduction	99
Figure 3.5.1:	Workflow Diagram for Classification	101
Figure 3.5.2:	Feedforward neural network (FNN) in Sign Language classification	102
Figure 3.10:	Work Breakdown Structure of the Project	113
Figure 3.11a:	Gantt Chart of Project I	114
Figure 3.11b:	Gantt Chart of Project II	115
Figure 4.1.1a:	Training history of Keypoint classifier (Accuracy over epochs)	117
Figure 4.1.1b:	Training history of Keypoint classifier (Loss over epochs)	119
Figure 4.1.1c:	Key Training Metrics of Keypoint classifier	120
Figure 4.1.2:	Confusion matrix of Keypoint classifier	121
Figure 4.1.3:	Classification Report of Keypoint classifier	122
Figure 4.2.1a:	Training history of Point History Classifier (Accuracy over epochs)	123
Figure 4.2.1b:	Training history of Point History Classifier (Loss over epochs)	124
Figure 4.2.1c:	Key Training Metrics of Point History Classifier	125
Figure 4.2.2:	Confusion matrix of Point History classifier	126
Figure 4.2.3:	Classification Report of Point History classifier	127
Figure 4.1.1a:	Age of Testers	132
Figure 4.1.1b:	Ethnicity of Testers	132

Figure 4.1.1c: Gender of Testers	132
Figure 4.1.1d: Hand Dominance of Testers	132
Figure 4.1.1e: ASL Proficiency	132
Figure 4.1.1f: Experience of ASL Tech	132
Figure 4.2.1a: Tester 1 doing 'Best' gesture being detected	138
Figure 4.2.1b: Tester 2 doing 'Birthday' gesture being detected	138
Figure 4.2.1c: Tester 3 doing 'I' gesture being detected	138
Figure 4.2.1d: Tester 4 doing 'Please' gesture being detected	138
Figure 4.2.1.1aa: 36cm – from table Light condition	139
Figure 4.2.1bb: 53cm – from table Light condition	139
Figure 4.2.1cc: 36cm – from table Dim condition	139
Figure 4.2.1dd: 53cm – from table Dim condition	139
Figure 4.3.1: Bar Chart of Average Duration (s) and Average Speed of Static Gestures (s^{-1})	148
Figure 4.3.2: Bar Chart of Average Duration (s) and Average Speed of Moving Gestures (s^{-1})	149
Figure 4.3.3: Line Graph of Recognized counts of Static Gestures	150
Figure 4.3.4: Line Graph of Recognized counts of Moving Gestures	151

CHAPTER 1

1. INTRODUCTION

1.1. General Introduction

Sign language is a medium that is used in communication in the deaf and hearing-impaired communities, which involves expressions and hand gestures. As reported in 2017, there were about 70 million deaf people that used sign language as their first language (Haj et al., 2017). According to a 2017 article on the United Nations website, sign language is very diverse, with more than 300 different sign languages available worldwide based on the geographical location and culture (United Nations, 2022). Commonly used sign languages include American Sign Language (ASL), Chinese Sign Language (CSL) and many more (Farooq et al., 2019). Despite the variety of sign languages out there, they share some common features such as hand movements, therefore the likeability of understanding between deaf/hard of hearing people of different countries still is high.

However, there is still a social barrier between the hearing people and the deaf/ or hard of hearing people. As most people are not familiar with sign language, therefore this makes real-time communication challenging. In such cases, the signers would need the external assistance of a translator to facilitate the conversation with the hearing, which might cost them some money. A **Real-Time Hand Gesture Recognition System To Interpret Sign Language** would be a cheaper, long-term solution. To achieve this, it requires the knowledge of machine learning, sensing technologies as well as AI concepts (such as Deep Learning) and algorithms to build the system.

Several popular methods have been developed and used for this purpose, including **OpenCV** (Ismail et al., 2021) and **OpenCV with Mediapipe** (Riaz,

2022). OpenCV, a widely used open-source computer vision library, has been preferred for its versatility and robustness in detecting and tracking hand gestures. On the other hand, OpenCV with Mediapipe has gained traction for its ability to provide accurate and efficient hand tracking and gesture recognition in real-time.

Another promising solution for hand-signing recognition is **LeapMotion Controller (LMC) with training of Convolutional Neural Network (CNN)** (Lupinetti et al., 2020). LMC is a small, portable device that tracks hand movements in 3D space, while CNN is an algorithm that can recognize complex hand gestures with high accuracy.

Gloves with sensors are also an attractive option for real-time gesture recognition, providing precise and accurate data for hand tracking and gesture recognition (Pragati et al., 2009). Additionally, the **Skeleton Aware Multimodal SLR framework (SAM-SLR)** has been developed as a multimodal approach to hand gesture recognition, combining visual and skeletal information, and has shown promising results in recognizing complex hand gestures in real-time (Jiang et al., 2021). These techniques have their unique strengths and limitations, depending on the specific application and the accuracy and speed required for real-time gesture recognition.

In this project, a comprehensive review will be provided on the popular methods used in the hand gesture recognition systems to interpret sign language. The strengths and limitations of each technique will be analyzed, and a method that combines the strengths of these techniques will be suggested to improve the accuracy of the sign language hand-gesture prediction system. Additionally, the challenges faced during the implementation of these techniques will be discussed, and possible solutions will be suggested. This

project aims to contribute to the development of a reliable and cost-effective recognition system for sign language interpretation, which would enhance the quality of life for individuals with hearing impairments.

1.2. Importance of the Study

The study on the development of a Recognition System To Interpret Sign Language would bring **independence to individuals with hearing impairments or hearing-challenged individuals without the reliance of a human interpreter**. With the computer vision technology, the system would detect and recognize hand gestures, allowing for real-time interpretation of sign language. This would bring social inclusion by bringing together the sign language users and people who do not understand sign language in their daily communications.

Secondly, this study also would increase the **efficiency of existing systems by reviewing the methodologies used and effectiveness of existing sign language interpretation systems in terms of their sign language capturing, recognition, translation, and representation**. By comparing the performance of the different systems, in aspects of accuracy, and errors, the study can identify areas where the current interpreter systems could be enhanced. This could ensure a more accurate representation of the original signed message.

Lastly, the study would help make **communication more accessible, and more cost-effective to the deaf and hard of hearing** when the sign language interpretation system becomes more widely available with the existence of a real-time interpreter system. With its usage in various settings, such as in workplaces, schools, and common spaces, individuals with hearing impairments or hearing-challenged individuals could fully participate in social and economic activities. Moreover, it will also ensure equal access to video

consultations for the deaf and hard of hearing especially in health consultations via telehealth and improve the lifestyle of the community.

1.3. Problem Statement

1. Dependency on trained interpreters, which may not always be available or accessible to the deaf and hard of hearing community.

Individuals with hearing impairments or hearing-challenged individuals often faces significant communication barriers due to the unavailability or inaccessibility of trained sign language interpreters. According to a survey conducted by the National Association of the Deaf (NAD), individuals have reported challenges in accessing communication services, including instances of unavailable or unqualified interpreters (National Association of the Deaf, 2023). The demand for a real-time interpreter is expected to rise significantly by 2050, as reported by the World Health Organization (WHO) (World Health Organization, n.d.).

2. Communication gap between individuals with and without hearing impairments in real-time.

Individuals with hearing impairments struggle to understand spoken language in real-time, leading to miscommunication and exclusion. This issue can result in misunderstandings and communication gaps, as evidenced by a study published in the Journal of Deaf Studies and Deaf Education (Zaidman-Zait and Dotan, 2017).

3. Inadequacy of the existing systems to work with variability in sign language

Existing sign language recognition systems often struggle to accurately recognize hand gestures, especially under varying lighting and background conditions. Studies have shown that recognition accuracy can significantly decrease in low-light environments (Mohammed et al., 2019). Additionally, these systems may have difficulty interpreting sign language gestures due to the high variability in individual signing styles (Quer and Steinbach, 2019).

1.4. Aim

The aim of this project is to develop a Hand Gesture Recognition System To Interpret Sign Language that addresses the interrelated problems mentioned above.

1.5. Objective

1. To design and develop an efficient algorithm for real-time hand gesture recognition that can detect and interpret sign language.

This is to reduce the reliance on trained interpreters as he or she may not always be available or accessible. This enables individuals with hearing impairments to communicate more effectively and independently.

2. To evaluate the efficiency of the hand gesture recognition system in real-time communication scenarios.

This is essential to ensure that the system meets the communication needs of people with hearing impairments, reducing the potential of social barrier between individuals with and without hearing impairments.

3. To identify potential challenges in the adoption of the system and develop solutions to address them.

This is crucial to ensure the successful adoption of the system and its effectiveness. Solutions will be figured out to tackle these challenges.

1.6. Research Questions

As part of the development of this system, the research questions that will guide the study are:

- 1. How can an algorithm be designed and developed to improve the efficiency of real-time hand gesture recognition in sign language interpretation?**

- 2. How efficient is the hand gesture recognition system in real-time communication scenarios?**

- 3. What are the potential challenges in the adoption of the system and how can they be addressed?**

1.7. Proposed Solution

Workflow:

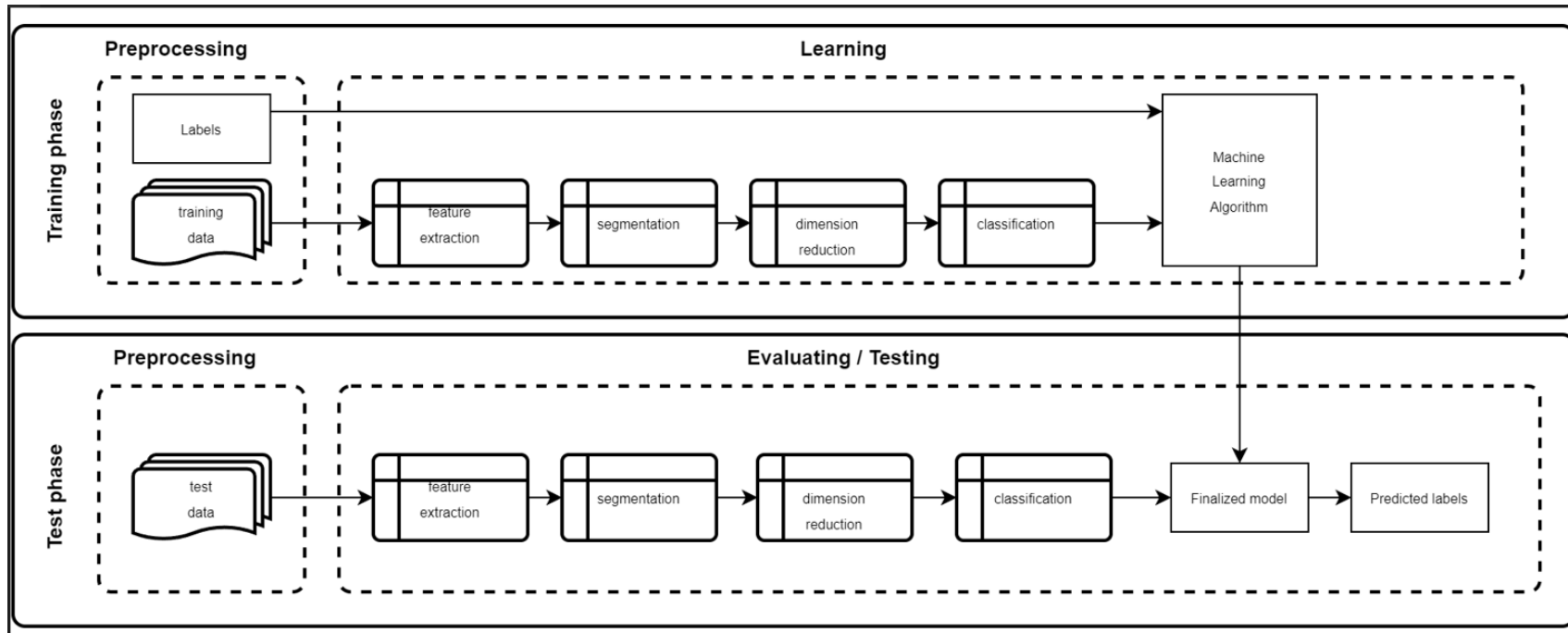


Figure 1.6.1: Workflow Diagram for the System for recognizing ASL hand gestures

The workflow of the proposed solution for the system for recognizing hand gestures in real-time to interpret sign language shown in Figure 1.6.1 involves the splitting into two parts: a training part and a test part.

During the **training phase**, the goal is to train a machine learning model on the training dataset. This phase typically involves several steps, starting with pre-processing the data. This **pre-processing step** involves cleaning and organizing the data and labelling it if necessary. This labelling is important to help the machine learning algorithm to understand what the data represents.

Once the data is pre-processed, the **learning phase** begins. This phase involves several sub-steps such as segmentation, feature extraction, dimension reduction, classification, and choosing a suitable machine learning algorithm. Feature extraction is conducted when the most relevant features are selected from the data to be used to train the machine learning model. Segmentation involves breaking down the data into smaller, more manageable parts. Dimension reduction involves reducing the number of features to avoid overfitting the model. Classification involves categorizing the data into classes based on its features. Finally, a suitable machine learning algorithm is chosen to perform training on the model based on the data.

After that, it will move to the **test phase**. During this phase, the goal is to make predictions on the test data with the existing trained model. This phase also starts with a **pre-processing step**, which involves cleaning and organizing the data. The data is then passed through the trained model, where it goes through the same sub-steps as the **learning phase**, including segmentation, feature extraction, dimension reduction, and classification. The model then generates predicted labels for the test data.

Finally, the predicted labels are checked if they matched to the actual labels of the test data. Then, evaluation takes place to calculate accuracy, precision, recall, and F1 score. Satisfactory model's performance can be used to make future predictions. If not, the model can be refined and retrained using different techniques or datasets.

In summary, the training phase includes pre-processing the data and training the machine learning model, while the test phase involves pre-processing the test data, using the trained model to make predictions, and evaluating the model's performance.

Architecture Design:

To implement this workflow, the proposed solution will require an architecture design that integrates the different stages seamlessly.

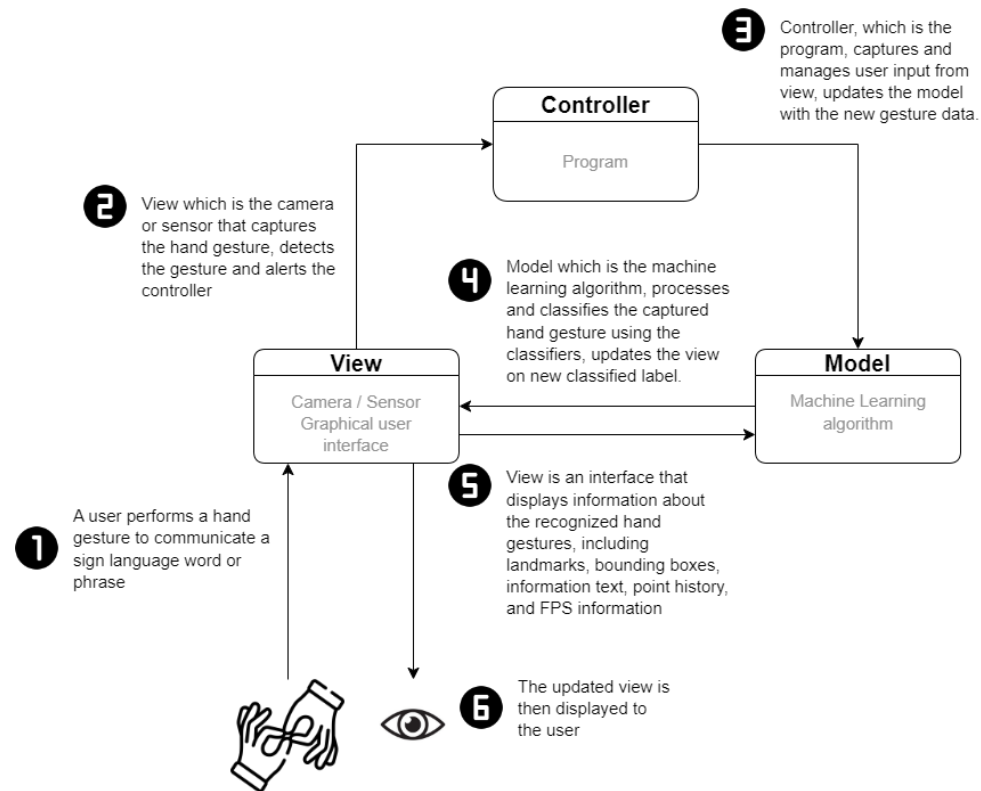


Figure 1.6.2: Model-View-Controller Architecture Pattern in context of system

The proposed solution will be developed using the **Model-View-Controller (MVC) pattern** as shown in Figure 1.6.2, which is a popular architecture pattern that divides the system into three components: the model, the view, and the controller (Pop and Altar, 2014). The **model** component of the system will handle the data and the application's logic. The **view** component will display the output of the system to the user, while the **controller** component will handle the user's input and interact with the model and view components.

The **sequence of events** in the development of the system will involve a user performing a hand gesture that they want to communicate, which will be detected by the **view** component. The view component will alert the **controller** component, which will capture and manages user input. The **model** component, which is the machine learning algorithm, will process and classify the captured hand gestures using classifiers and alert the view component that it has new classified result. The **view** component will display the recognized hand gestures, including landmarks, bounding boxes, information text, point history, and FPS information to the user, who can continue to perform hand gestures to communicate with the system. This process will repeat in real-time, allowing the user to communicate using hand gestures until the user presses the ESC key, which enable sign language words or phrases translated in real time.

The architecture design will also consider the flow of data between the different stages, and the **hardware and software resources** required to execute the system. A modular approach will be used to develop the different components separately and integrate them to form the complete system. This approach will allow for easy maintenance, scalability, and flexibility of the system.

Furthermore, the proposed solution's architecture design will take into account the **computational resources** required to execute the system efficiently. This consideration will ensure that the system runs smoothly and can handle recognition tasks.

In summary, the development of this system to interpret sign language will require a well-planned architecture design, which is the MVC pattern to ensure the modularity and maintainability of the system.

1.8. Scope and Limitations

The scope of this study is to **develop a system for recognizing sign language's hand gestures in real-time**. Cameras or sensors are used to capture hand movements, and algorithms would interpret the gestures. The development would take place over 8 months, and the system's variables and factors would include camera or sensor specifications, lighting and background conditions, hand shape and movement variability, and machine learning algorithms. The project aims to achieve the objectives as stated in Section 1.4. Objectives:

To fulfil **Objective 1** as stated in Section 1.5, this system would involve the **use of technologies such as deep learning**. For instance, machine learning algorithms would be used to classify hand gestures and translate them into captions. The pre-processing techniques such as noise reduction, segmentation, and normalization would be used to remove any irrelevant information. After that, the input data would be extracted, including the shape, position, and movement of the hand to accurately identify specific hand gestures.

To fulfil **Objective 2** as stated in Section 1.5, the system would be **trained on a variety of sign language gesture datasets** to ensure that it can interpret more gestures. The system would be tested in various environments, including different lighting and background conditions, to determine its effectiveness in real-life situations. Evaluations would be conducted by comparing its results in aspects of speed and response time with those of human interpreters. Limitations of the system would be identified, and solutions proposed to overcome them. This is followed by Optimization for real-time processing which would be conducted to ensure that hand gestures are detected and interpreted in real-time without any significant delays.

To fulfil **Objective 3** as stated in Section 1.5, an investigation would be conducted to **identify the challenges posed by the variability in sign language gestures**, including differences in hand shape, movement, and orientation, as well as the effects of lighting and background conditions, hardware and software limitations, and usability factors, such as ease of use and accessibility for individuals with hearing impairments. On hardware and software limitations, the system's performance may be affected by the capabilities of the hardware components used. Therefore, the study will be limited to using specific sign language (which is American Sign Language, or ASL), with the below hardware setup in Table 1.7.1 and software setup in Table 1.7.2.

Table 1.7.1: Hardware Setup

Hardware Setup	Description
Camera	A high-quality camera is required to capture the hand gestures in real-time.
Processor	Fast graphics processing unit (GPU) that is capable of handling real-time video processing and gesture recognition algorithms.
Memory	Enough memory to store the trained machine learning models and the input data
Display	Show the recognized gestures or translated text.

Table 1.7.2. Software Setup

Software Setup	Description
Operating system	Windows, Linux, or MacOS
Gesture recognition algorithm	Neural Network
Machine learning framework	TensorFlow Keras
Image processing library	OpenCV (used for preprocessing input images and feature extraction)
Testing and evaluation	Scikit-learn TensorFlow

Limitations of the development of a hand-gesture recognition system may include the environmental limitations, algorithmic limitations, hardware limitations, dataset limitations, and sign language-specific limitations.

The first limitations are **environmental limitations** including occlusion and lighting conditions. Occlusion occurs when part of the hand is hidden from the camera, which would result in recognition errors as the system is unable to recognize the complete hand gesture (Starner et al., 1996). Lighting conditions also interfere with recognition of hand gestures, in which poor lighting would result in shadows or reflections affecting the interpretation.

The second limitation is that **algorithmic limitations** include computational complexity. To develop the hand gesture recognition would involve processes, such as image processing which require computational resources and can be time consuming (O' Mahony et al., 2019).

The third limitation is **hardware limitations** is the limited availability of hardware. The recognition system may require specialized hardware, such as cameras which may not be available on older laptops.

The fourth limitations are **dataset limitations**, in which collecting and labelling such a dataset can be time-consuming as the system requires a large dataset of sign language gestures to train the model (Hou et al., 2019).

The next limitations are **sign language-specific limitations**, which include variability and complexity of sign language, context-dependent sign, real-time

recognition and user variability. In the context of variability and complexity of sign language, sign languages across different regions and countries make it difficult to create a universal recognition system, allowing only certain languages to be chosen (Woll et al., 2001). In addition, sign language involves a combination of not only hand gestures, also facial expressions, body posture, and movement, requiring advanced technology and algorithms. Next, sign language is context-dependent, which means it can have different meanings. Moreover, sign language interpretation to be built would be real-time which is expected to be challenging due to the complexity of the task and the need for fast processing.

Lastly, the **sign language-specific limitation** to be considered is user variability. The users may vary in terms of signing speed, style, and accuracy (Bellugi and Fischer, 1972). Therefore, the system must adapt to these variations to accurately interpret sign language.

1.9. Summary

Chapter 1 of the study introduces the **importance of developing a sign language recognition system** to facilitate communication between the deaf and hard of hearing communities. It explores various methods for recognition and proposes a method that combines their strengths to enhance the accuracy of sign language interpretation. The study aims to provide independence for individuals with hearing impairments or hearing-challenged individuals, promoting social inclusion. The **problem statement** outlines the communication challenges faced by individuals with hearing impairments or hearing-challenged individuals, while the **aim and objectives** of the study are presented. The **proposed solution** involves using machine learning algorithms, with the use of the Model-View-Controller (MVC) pattern. The **scope and limitations of the study** are also presented, which include the development of a recognition system for interpreting American Sign Language, or ASL, with solutions proposed for identified limitations.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The aftermath of the pandemic has left us to rely on virtual platforms for daily communication, including online learning, telehealth, and remote work. This has accelerated the adoption of virtual platforms for communication. However, currently there is no widely available recognition for sign language interpretation available in the market, leaving the individuals with hearing impairments or hearing-challenged individuals to amplify their need for accessible and inclusive communication.

While there are systems that can recognize hand gestures for sign language interpretation, it is prevalent that they are not yet ready to release. This is most likely due to the system is still in progress to accurately interpret all variations of sign language in real-time, producing minimal errors in interpretation, tested and validated by the individuals with hearing impairments or hearing-challenged individuals. Achieving this level of accuracy and reliability in real-time is expected to involve a significant amount of time and resources, with technical expertise.

With the belief that real-time interpretation technology would continue to advance as research in computer vision and machine learning progresses, it is expected to see more sophisticated systems become available in the future. This literature review would identify existing research in the field of hand gesture recognition systems for sign language interpretation. This approach is to identify gaps in knowledge, areas for improvement, and potential research directions, appropriate methodologies, to achieve the aim of increasing the reliability of the system.

2.2 Literature Review

Hand gesture recognition to interpret sign language encompasses the stages of pre-processing, segmentation, feature extraction, dimension reduction (if used), and classification.

Pre-processing is a stage when the input image is prepared for further processing. Pre-processing techniques are used to improve the quality of the image. In the literature review, we shall discuss presence or absence of pre-processing, along with techniques covering image processing which includes image resizing, smoothing, thresholding, as well as computer vision which involves depth images and skeleton data.

Segmentation is a stage when the separation of the hand gesture from the background takes place. Segmentation techniques are used to identify the hand region. Various techniques in segmentation, which include thresholding, contour detection, foreground, or background removal to track the hand or skeleton would be discussed. Segmentation may be assisted with **hand tracking API**, such as Mediapipe, hardware devices, such as Leap Motion Controller and sensors or software, such as skeleton tracking to improve the accuracy of the recognition system.

Feature extraction is a stage when features such as colour, texture, shape, and motion may be extracted from the image once the hand region is identified. Feature extraction techniques, which further categorised into geometric feature extraction such as hand shape and orientation, hand landmarks, hand shape descriptors, finger tracking, curvature, and surface normals, image processing feature extraction such as edge detection and contour detection, computer vision feature extraction such as depth estimation, 3D point cloud, motion features and depth-based features, mathematical feature extraction such as statistical features, sensor data feature extraction which includes sensor readings, and audio signal processing feature extraction which includes audio

features will be discussed. In addition to that, machine learning models or devices would be used such as CNN, ResNet, and SVM to provide additional input data for improved accuracy and robustness of the system.

Dimension reduction would be used to reduce the number of features in cases when the extracted features may be high-dimensional and redundant which results in classification being difficult. Therefore, it is not compulsory to be used in the development of the hand gesture recognition system. Techniques of **dimension reduction** used in existing studies include Principal component analysis (PCA) which is used to reduce the dimensionality of a dataset, linear discriminant analysis (LDA) which is used for classification tasks, independent component analysis (ICA) which is used to separate signals that are mixed.

Classification is a stage when the hand gesture is classified into one of several predefined categories. **Classification technique** in existing studies can be categorised as Supervised Learning Techniques which is to predict the class or category of new, unseen data, Unsupervised Learning Technique which is to identify patterns or relationships within the data, such as Template matching.

The literature review will also investigate if any **deep learning framework**, such as TensorFlow Lite or TensorFlow is used which leads to improved accuracy, and better performance with a set of tools and features provided for building, training, and deploying complex neural network architectures. Besides, the literature review shall identify whether a **traditional machine learning** which involves hand-engineered features **or deep learning model** which involves the use of artificial neural networks is being used in an existing study. The summary of the literature review would compare all existing studies in terms of its stages of development, deep learning framework, **accuracy, and complexity**.

2.3 OpenCV

Ahmad Puad Ismail et al. demonstrated in their 2020 research, published in the IOP Conference Series: Materials Science and Engineering, that Python and OpenCV can be utilised to create a Real-Time Hand Gesture Recognition System for Interpreting Sign Language. This was also supported in an article entitled '*A Hand Gesture Sign Language to Text Real-time Interpreter using Google Audiopipe Artificial Intelligence*' by Riaz Sulaimi who initially used OpenCV alone but encountered inconsistent results, the findings and outcomes would be mentioned in this segment.

2.3.1 Pre-processing

The proposed approach by Ahmad Puad Ismail et al. (Ismail et al., 2021) involved several stages, namely pre-processing, segmentation, feature extraction, and classification. In the **pre-processing stage**, the Haar-cascade classifier was used to detect the hand in the input image. The segmentation stage involved detecting hand gestures by calculating the space consumption within the area between the convex hull and contour of the hand as visualised in Figure 2.3.1 below.



Figure 2.3.1: The convex hull (in green) and the hand contour (in white)
(Ismail et al., 2021)

2.3.2 Feature Extraction

The **feature extraction** technique was based on the theory of Region of Interest (ROI), in which the region of overlapping ROIs, as visualised in Figure 2.3.2, are used to detect the appearance and gesture of the hand.

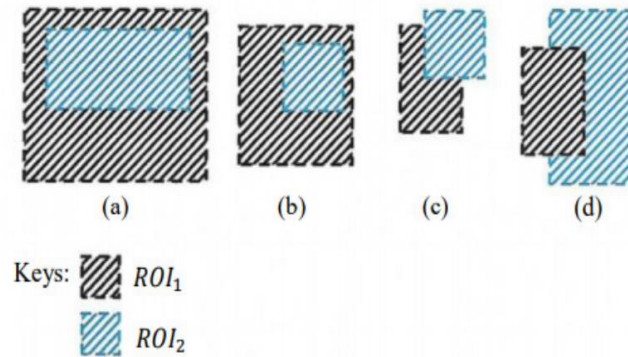


Figure 2.3.2: Region of overlapping ROIs (Ismail et al., 2021)

It was found that the region of interest (expressed in pixel value), for each gesture varies from one another.

Table 2.3.2: Analysis table for Region of Interest (Ismail et al., 2021)

Events	Area in Region of Interest, ROI		
	Area of Hull, H	Area of Contour, C	Area = H - C
0	62879	62879	0
1	13569	10876	2693
Good	13045	10821	2224
2	20646	13744	6902
Gun	20216	13804	6412
3	24362	17113	7249
OK	25468	16922	8546
4	25781	15395	10386
Rawr	20395	14148	6246
5	28383	16822	11561
Stop	33220	19047	14173

The differences of ROI between gestures are visualised in Table 2.3.2. Calculation is carried out to obtain Area, also known as area of recognition, where:

$$\text{Area} = H - C \quad (2.3.2.1)$$

H represents Area of Hull and *C* represents Area of Contour

Based on the analysis table 2.3.2, the numerical gestures of 0-5 consume area between 0 to 11561 pixels, while the phrases' gestures consume area between 2693 to 14173 pixels.

2.3.3 Classification

Lastly, **classification** was done to recognize the type of gesture, utilising a Haar-cascade classifier to determine whether the box in the frame could detect the hand gesture or otherwise.

The system was evaluated using a simulation. The simulation is conducted with a camera from the laptop. The screenshots of the simulation are displayed in Figure 2.3.3.1, Figure 2.3.3.2, and Figure 2.3.3.3.

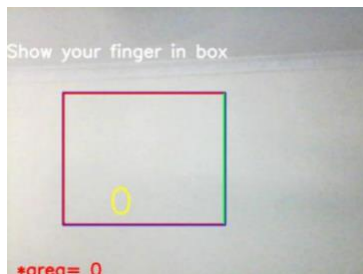


Figure 2.3.3.1 Image captured by camera (Ismail et al., 2021)

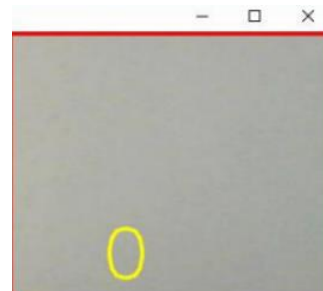


Figure 2.3.3.2 Bounding web box without any gesture shown. (Ismail et al., 2021)

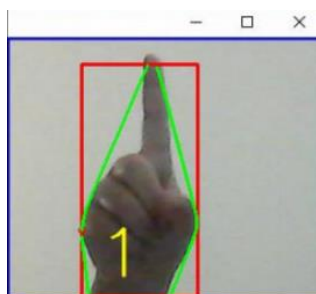


Figure 2.3.3.3. Recognition of hand showing number 1 (Ismail et al., 2021)

2.3.4 Accuracy

In the simulation part, Ahmad Puad Ismail et al. evaluated the proposed approach on a dataset of ten hand gestures, which are the numerical values from 1-5, as well as phrases of ‘Good’, ‘Gun’, ‘Ok’, ‘Rawr’ and ‘Stop’ which are posed from different scales for 10 times.

Table 2.3.4: Analysis table for gesture recognition (Ismail et al., 2021)

Events	Performance of Haar-cascade Classifier		
	Hits	Missed	False
1	10	0	0
Good	10	0	0
2	10	0	0
Gun	10	0	0
3	10	0	0
OK	10	0	0
4	10	0	0
Rawr	10	0	0
5	10	0	0
Stop	10	0	0

An analysis table in Table 2.3.4 then is produced to state the **hits, misses and falses** of Haar-cascade Classifier so as to determine if the box in the frame could detect the hand gesture. Based on all of the results in the table 2.3.4, the box in the frame can recognize all of the signs with 10 hits out of 10 trials.

2.3.5 Advantages

One of the notable **advantages** of this method is that it is computationally efficient and easy to implement, without the need for complex deep learning models. However, traditional computer vision techniques **may not** perform well in complex and noisy environments. This is proven in the article "*A Hand Gesture Sign Language to Text Real-time Interpreter using Google Mediapipe Artificial Intelligence*" by Riaz Sulaimi. It reports that the initial prototype of their system, which was developed using OpenCV, produced inconsistent results when detecting and interpreting hand gestures as visualised in Figure 2.3.5.1 and Figure 2.3.5.2.

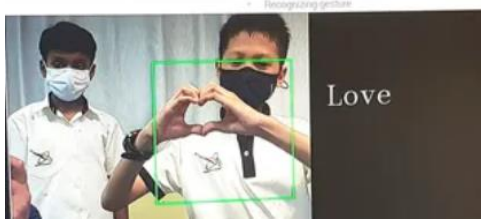


Figure 2.3.5.1: Right interpretation of sign language, 'Love' (Sulaimi, 2022)

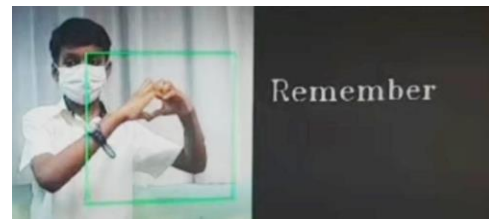


Figure 2.3.5.2: Wrong interpretation of sign language, 'Remember', supposed 'Love' (Sulaimi, 2022)

2.3.6 Disadvantages

The hand gesture detection was too dependent on many factors like a clean background and proper lighting for accurate hand detection (blob detection). The inconsistency in detecting and interpreting hand gestures led the team to research other possible ways to make detection more reliable and accurate, leading them to Google's Mediapipe. Therefore, as an enhancement, deep learning-based approaches using frameworks such as TensorFlow, Keras, and PyTorch, which involve a convolutional neural network (CNN) for feature extraction and classification, or a high-level framework for building media processing pipelines, such as Mediapipe are suggested along with OpenCV.

2.4 OpenCV with Mediapipe

Combining **OpenCV with Mediapipe** provides a powerful framework for developing recognition systems that can be used to interpret sign language. The open-source tool Mediapipe, created by Google, allows for the extraction of hand landmark position data from images using the Mediapipe Hands module. In this review, we discuss three studies that have utilised this approach and highlight the similarities and differences in their methodologies and findings.

The 2022 IEEE International Symposium on Smart Electronic Systems (iSES) paper titled “*Hand Gesture Recognition System in the Complex Background for Edge Computing Devices*” by C. M. Suryateja et al. (Suryateja et al., 2022), uses OpenCV with MediaPipe Hands framework to extract the landmarks' position data, which are then recorded in a CSV format, in the development of the system for recognizing hand gestures using palm and finger positions. The system comprises four clear components, as illustrated in the flowchart presented in Figure 2.4: a palm detection module, a hand landmark (HL) position extractor, a data scrubber, and a gesture recognizer.

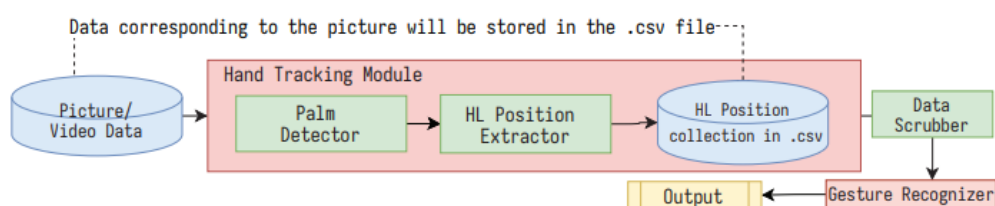


Figure 2.4: The structure of the Sign Language Recognition system (Suryateja et al., 2022)

The finger and palm position data were obtained utilizing the MediaPipe Hands framework, showcasing an average accuracy of 95.7% in palm detection. The training of the system involved leveraging the American Sign Language Dataset, as depicted in Figure 2.4.1. This dataset comprises hand gestures categorized into 10 classes, representing the letters "A" to "J". Each gesture image has dimensions of 160×120 , and the dataset consists of 200 photos for each gesture type, featuring variations in background and individuals.

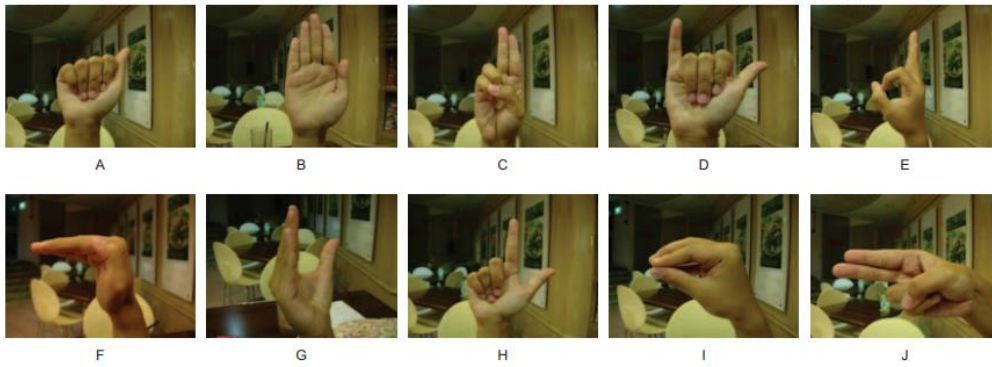


Figure 2.4.1: 10 gestures in American Sign Language (ASL) corresponding to the alphabet's "A" to "J." (Suryateja et al., 2022)

The proposed approach by C. M. Suryateja et al. involved several stages, namely pre-processing, segmentation, feature extraction, and classification. The study did not mention the use of any dimensionality reduction techniques.

2.4.1 Pre-processing

The **pre-processing** involves transforming the image dataset into a hand landmark (HL) position dataset using MediaPipe Hands, where the dataset is normalised to the wrist as visualised in Figure 2.4.1.1, and the output is stored in a dataset structured as a list for each gesture.



Figure 2.4.1.1: Wrist and upwards as result of normalisation (Suryateja et al., 2022)

The dataset is divided into training and testing, where 80% of the data is used for training and 20% for testing. Pre-processing also involves the removal of invalid entries in the dataset using Pandas.

2.4.2 Feature Extraction

Feature extraction involves utilizing a Hand Tracking Module (HTM), comprising a Palm Detector and an HL Position Extractor, as illustrated in Figure 2.4.2. The Palm Detector functions on the entire input image to detect the palm in the data. On the other hand, the HL Position Extractor operates on the cropped hand bounding box provided by the palm detector, extracting position data for the landmarks.

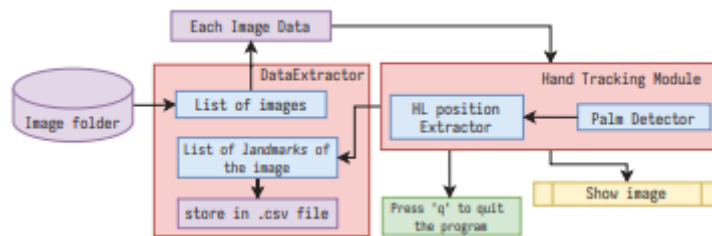


Figure 2.4.2: The working of HTM (Suryateja et al., 2022)

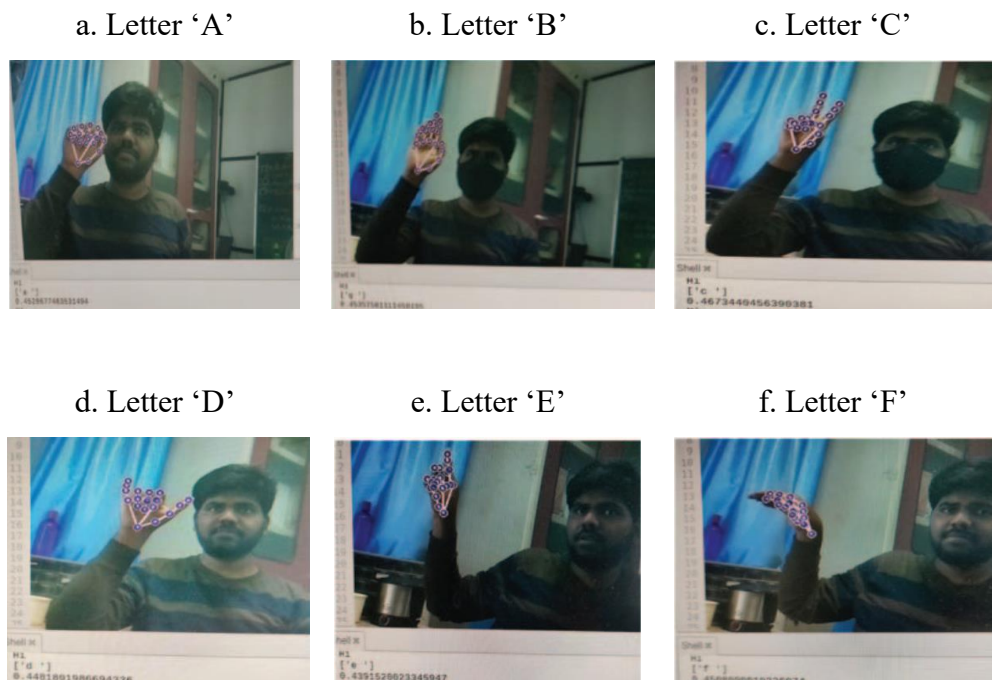
2.4.3 Segmentation

The **segmentation** process is fundamental in the extraction of hand landmarks, encompassing both palm and finger positions. This process follows the feature extraction stage, leveraging the Hand Tracking Module (HTM) and DataExtractor files. Following the identification of the palm in the input image by the Palm Detector during feature extraction, this information is utilized to train the system to estimate bounding boxes for rigid objects such as palms and fists. As palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Meanwhile, the HL Position Extractor in HTM continues to operate on the cropped hand bounding box provided by the Palm Detector and returns landmarks with the goal that this extracted data will be used to build a ML model for gesture recognition. Thus, it can be concluded that the HTM may have contributed to both Feature extraction and Segmentation by identifying the relevant features (palm and finger positions) necessary for hand gesture recognition.

2.4.4 Classification

The process of **classification** is carried out during the development and training of the ML model using machine learning classification algorithms like Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Support Vector Machines (SVM), and K-Nearest Neighbours (KNN), the ML model is trained. The training data is derived from the extracted HL (Hand Landmark) position dataset obtained through the segmentation process. It was trained using an 80-20 split of the dataset into training and testing sets. The hyperparameters of the model were tuned to achieve the best possible accuracy.

After training the model, testing the model took place when the test dataset is used to predict the hand gestures in the test data. Evaluation of performance is done by comparing the predicted gesture, as visualised in Figure 2.4.4 with the actual gesture.



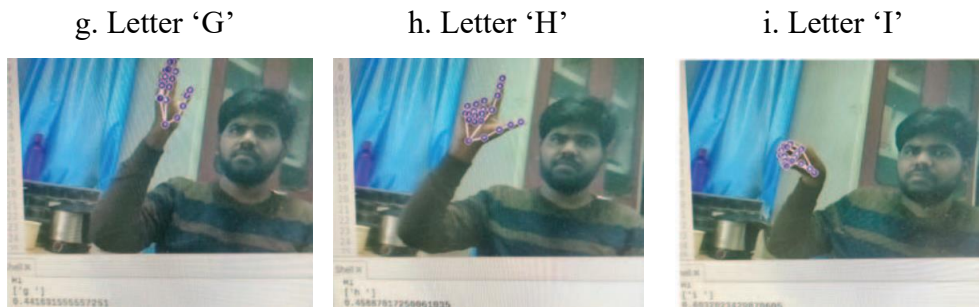


Figure 2.4.4 (a-i): Real-time evaluation outcomes of the gesture recognition system (Suryateja et al., 2022)

2.4.5 Accuracy

The **accuracy** of the model specific classification algorithm, such as KNN was calculated using the confusion matrix as visualised in Figure 2.4.5, which gives the number of true positive, true negative, false positive, and false negative predictions.

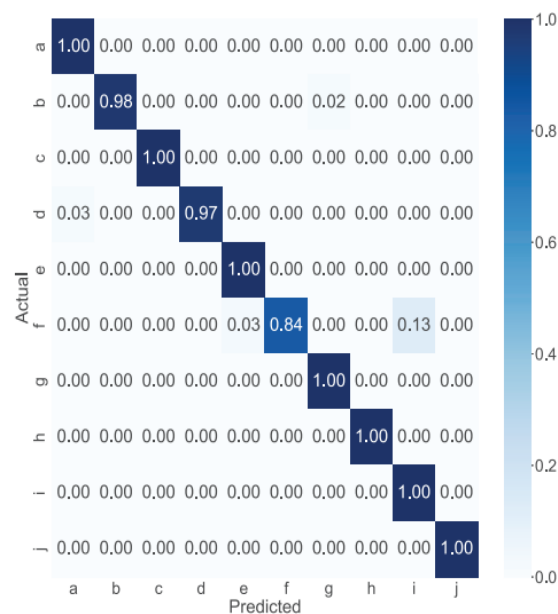


Figure 2.4.5: Confusion matrix of the model utilizing K-Nearest Neighbours (KNN) (Suryateja et al., 2022)

The **Decision Tree Classifier (DTC)** demonstrated perfect training data accuracy at 100% and achieved an accuracy of 96.4% on the test data. On the other hand, the model employing the **Random Forest Classifier (RFC) algorithm** achieved a training data accuracy of 100% and an accuracy of 97.2% on the test data. Utilizing a radial base kernel, the **Support Vector Machines (SVM)** achieved an accuracy of 97.9% on the test data. Additionally, the **K-Nearest Neighbours (KNN)** algorithm-based model achieved a training accuracy of 97.4% and an accuracy of 98.1% on the test data.

Table 2.4.5.1: Precision, F1-score, and Recall were computed for all models during the ML model development and testing (Suryateja et al., 2022)

Model	DTC			RFC			SVM			KNN		
Class	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score
A	91.8	94.4	93.1	98.6	97.3	98.6	97.2	97.2	97.2	98.6	97.3	98.6
B	95.5	97.2	96.6	97.7	97.7	98.8	97.7	97.7	97.7	98.8	100	98.8
C	100	92.5	96.1	100	100	100	97.2	100	98.6	100	100	100
D	97.3	92.5	94.8	97.3	97.3	98.6	97.3	97.3	97.3	98.6	100	98.6
E	88.8	91.4	90.1	97.2	97.2	98.6	100	97.2	98.6	98.6	97.2	98.6
F	83.8	96.2	89.6	94.9	100	91.2	90.3	100	94.9	91.2	100	91.2
G	97.2	97.2	97.2	98.6	100	98.6	97.2	100	98.6	98.6	97.3	98.6
H	93	100	96.3	100	100	100	100	97.2	98.8	100	100	100
I	89.6	86.6	88.1	93.5	87	93.5	100	90.6	95	91.8	88.7	93.5
J	100	97.1	98.5	100	100	100	100	100	100	100	100	100

The precision, F1-score, and recall values for the models based on Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Support Vector Machines (SVM), and K-Nearest Neighbours (KNN) are presented in Table 2.4.5.1 above.

Table 2.4.5.2: Model and respective accuracies for the 10-class NUS dataset, including Training Accuracy (Tr. A), Testing Accuracy (Te. A), Overall Accuracy (O.A), and Average Inference Time (A.I.T) (Suryateja et al., 2022)

Model	Tr.A	Te.A	O.A	A.I.T
KNN	97.4%	98.1%	98.1%	48ms
SVM	96%	97.9%	97.6%	45ms
DTC	100%	96.4%	99.0%	73ms
RFC	100%	97.2%	99.5%	83ms

In Table 2.4.5.2, the Training Accuracy (Tr. A), Testing Accuracy (Te. A), Overall Accuracy (O.A), and Average Inference Time (A.I.T) for the DTC,

RFC, SVM, and KNN-based models are presented. The inference time for the trained ML model falls within the range of 45-83 ms, while the overall accuracy ranges from 97.6% to 99.5%. Following this, the model was deployed on edge computing devices such as Raspberry Pi 4 and NVIDIA AGX Xavier to create a portable ASL recognition system.

2.4.6 Advantages

As an overall, the system utilised a Deep Learning Framework - the Mediapipe Hands framework, which is an open-source framework for hand tracking and gesture recognition. The **advantages** of using the framework include its high accuracy in palm detection and hand landmark extraction.

2.4.7 Disadvantages

However, the **disadvantages** include the complexity of the system built is high, and the dataset might contain invalid entries, which require removal.

2.5 LeapMotion Controller (LMC)

The Leap Motion Controller (LMC) as visualized in Figure 2.5 is a small device that can accurately track hand movements and capture fine-grained details, making it a popular choice among researchers for the development of gesture recognition systems for sign language interpretation.

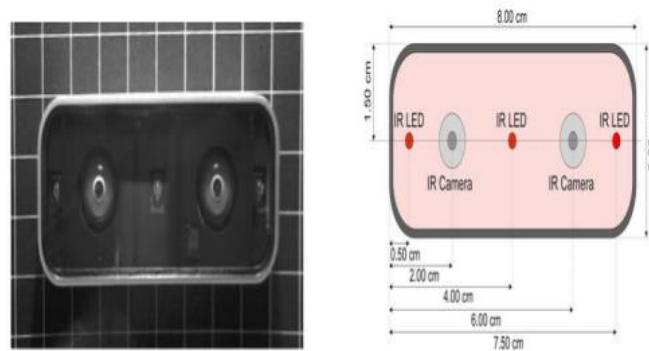


Figure 2.5: Leap Motion Controller (LMC)

Its ability to detect hand gestures and track hand movements in real-time has been utilized in many research papers, including *"Real Time Sign Language Recognition using the Leap Motion Controller"* by Naglot and Kulkarni (2016), *"American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach"* by Chong and Lee (2018), and *"British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language"* by Bird, Ekárt, and Faria (2020).

The first study is a paper by Naglot and Kulkarni (2016) that focused on developing a real-time sign language recognition system using the Leap Motion Controller (LMC) (Naglot et al., n.d.). The system aimed to track hand movements and recognize hand gestures in real-time. The overall process involved pre-processing, segmentation, feature extraction, classification, and final prediction. The authors used deep learning frameworks such as MLP, DEvoMLP, CNN, Dense Interpretation Network, Image Classification

Network, and Bone Data Classification Network for the classification task as illustrated in Figure 2.5.1.

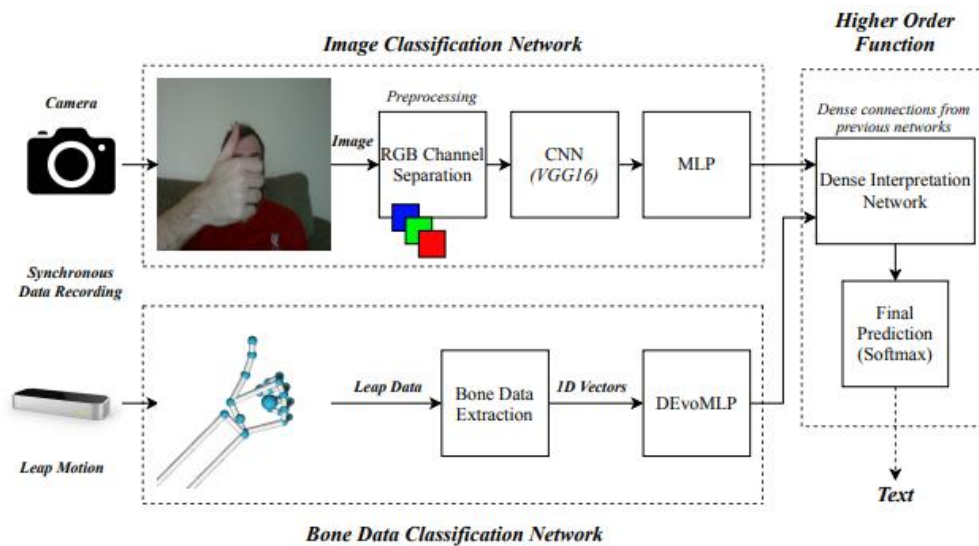


Figure 2.5.1: Overall networks and function involved in building the British Sign Language Recognition System (Naglot et al., n.d.)

The second paper titled "*American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach*" by Chong and Lee (2018) proposes a system for recognizing American Sign Language (ASL) using a Leap Motion Controller (LMC) with a machine learning approach (Chong and Lee, 2018). The system uses the LMC to detect hand gestures and track hand movements. In the last study to be investigated is by Bird, Ekárt, and Faria (2020), they developed a British Sign Language recognition system by utilizing a Late Fusion of Computer Vision and Leap Motion Controller (LMC) with transfer learning from American Sign Language (ASL). The proposed system flow of the study is shown in Figure 2.5.2 below:

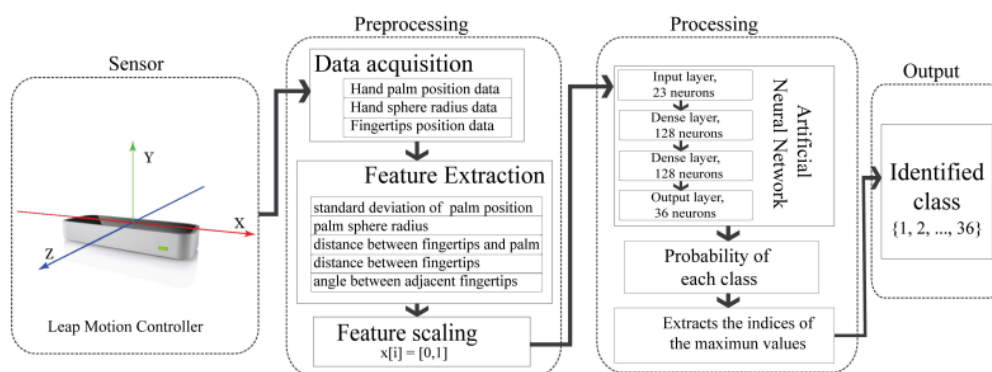


Figure 2.5.2: Proposed flow for development of Sign Language Recognition
Using Leap Motion Controller(Chong and Lee, 2018)

2.5.1 Pre-processing

The **pre-processing** step by Naglot and Kulkarni (2016) involved collecting data from the sensors, camera, and the LMC. The authors recorded each gesture for 30 seconds, 15 seconds per dominant hand, at a frequency of 0.2 seconds as shown in Figure 2.5.1.1. (Naglot et al., n.d.)

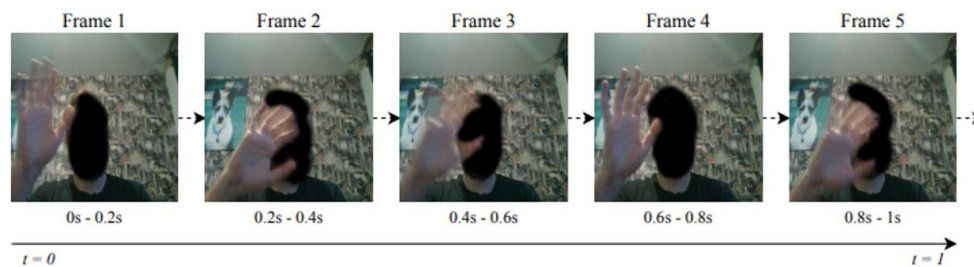


Figure 2.5.1.1: RGB image data of 1s which collected at frequency 0.2s/frame
(5 Hertz)

The data was captured using a device positioned above the camera, facing the subject. The collected data were then inserted into the dataset as numerical vectors to be classified. The preprocessing stage also encompassed addressing empty frames in cases where the sensor failed to detect either hand. The pre-processing step by Chong and Lee (2018) involved normalizing the dataset before providing it to the Artificial Neural Network (ANN) for training and testing (Chong and Lee, 2018). During the pre-processing stage by Bird, Ekárt, and Faria (2020), the LMC device was connected to a desktop PC and placed on the table to detect and track the subject's hand and finger gestures (Bird et al., 2020). The setup is shown in Figure 2.5.1.2 below.

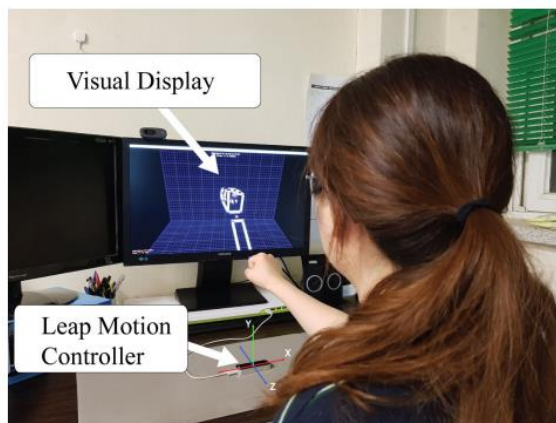


Figure 2.5.1.2: Setup of LMC with 3D graphic display (Bird et al., 2020)

The data collected from the LMC device was pre-processed to extract meaningful features.

2.5.2 Segmentation

In the **segmentation** step by Naglot and Kulkarni (2016), the authors collected a dataset of British Sign Language (BSL) comprising 18 distinct gestures from five participants (Bird et al., 2020). Each gesture was recorded for 30 seconds, with 15 seconds allocated per dominant hand. This resulted in a dataset of numerical vectors for each gesture. Participants were instructed to perform the gesture at a comfortable pace during the recording. To maintain data quality and prevent fatigue, a recording duration of 15 seconds was chosen. In the segmentation stage by Chong and Lee (2018), the Palm and Finger dataset were selected as features for the feature extraction process (Chong and Lee, 2018). The Leap motion API provides various features for hand, fingers, bones, and gestures, including finger direction, position, and length as shown in Figure 2.5.2 below.



Figure 2.5.2: Features provided by LMC API

The ANN was trained using two features: the Euclidean distances between the consecutive fingertip position to palm position and the Euclidean distances between the fingertip position of each consecutive finger.

2.5.3 Feature Extraction

In the **feature extraction** step by Naglot and Kulkarni (2016), various features were extracted from the recorded data (Naglot et al., n.d.). These features encompassed the initiation and termination coordinates of the arm within 3D space (X, Y, and Z coordinates), along with the 3D angle between these points. Furthermore, the velocity of the arm in all three dimensions (X, Y, and Z) was calculated. Additionally, the 3D spatial coordinates (X, Y, and Z) of the elbow and wrist positions were recorded. The features also included the pitch, yaw, and roll of the palm. Finally, the 3D angle of each finger and each bone in the hand as visualized in Figure 2.5.3 were also extracted from the recorded data.

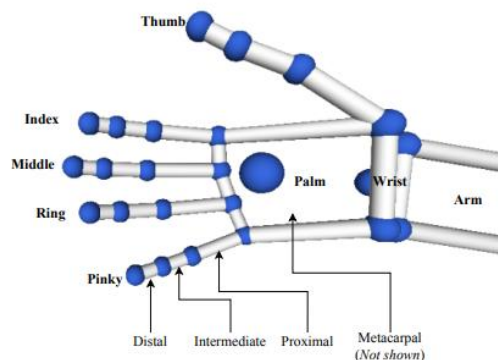


Figure 2.5.3: Bone data of each finger detected by LeapMotion sensor

The use of multiple features provided a comprehensive representation of hand and arm movements, enabling the system to recognize a wide range of sign language gestures. In the feature extraction stage by Bird, Ekárt, and Faria (2020), features such as hand palm sphere radius, hand palm position, and fingertip position were extracted from the collected data (Suryateja et al., 2022). The authors organized these features into five distinct groups and extracted a total of 23 features to use as input parameters to the classifiers.

2.5.4 Classification

For the **classification** task by Naglot and Kulkarni (2016), two main classification tasks were performed: image classification and gesture recognition using the LMC. For image classification, the authors-initiated feature extraction from image data by employing a Convolutional Neural Network (CNN) with the VGG16 architecture as a starting point as illustrated in Figure 2.5.4 below.

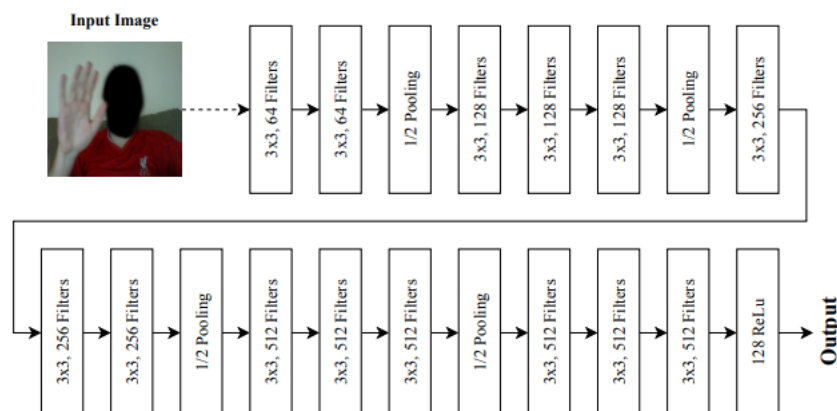


Figure 2.5.4: Input image is fed into a fine-tuned VGG16 CNN, where a layer comprising 128 ReLU neurons generates the output, which is then utilized in late fusion with the Leap Motion network (Naglot et al., n.d.)

The authors used the first three hidden layers of the CNN, containing 4096 neurons each, as feature extractors. For concatenation, the Softmax output layer was eliminated. For classification by Chong and Lee (2018), the paper used a

Multilayer Perceptron (MLP) neural network, which has three types of layers - input, output, and hidden layers as shown in Figure 2.5.4.1 below on its architecture.

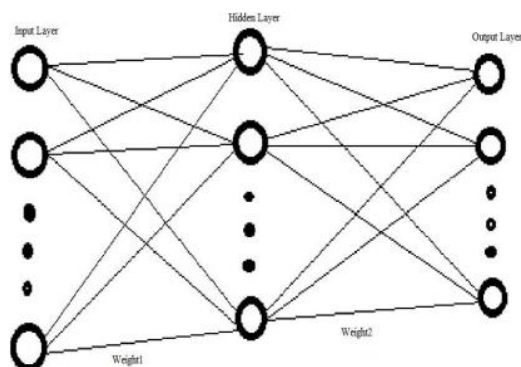


Figure 2.5.4.1: Architecture of MLP neural network (Chong and Lee, 2018)

The network was trained using the **Back Propagation (BP) algorithm**, which involved applying input to the network, initializing weights and bias, forward pass, and reverse pass. During the forward pass, the net input and output to each hidden layer unit and output layer unit were calculated. During the reverse pass, the error was calculated, and the weights of interconnections from the hidden layer unit to the output layer unit were updated. The error of the hidden layer was also calculated, and the weights of interconnections from input layer neurons to hidden layer neurons were updated. The processing module by Bird, Ekárt, and Faria (2020) provided classification results using two classifiers: **Support Vector Machine (SVM) and Deep Neural Network (DNN)**. The SVM was used with a linear kernel and "one-vs-rest" (OVR) decision method to handle the 36 classes in the dataset, while the DNN model consisted of two hidden layers with 128 neurons each, using the ReLU activation function, and a final Softmax layer with 36 neurons. The models were trained and tested using the leave-one-subject-out (LOO) approach to avoid subject bias.

2.5.5 Deep Learning Model / Traditional Machine Learning Model

For gesture recognition using the LMC, a **Dense Interpretation Network and a Bone Data Classification Network** were used by Naglot and Kulkarni

(2016). The authors used DEvoMLP for the classification of Leap Motion data. The most effective layer, composed of 16 neurons, received input from both the Image and Leap Motion classification networks. This layer was linked to a final SoftMax output. Furthermore, a higher-order function network was employed for late fusion, combining the two modalities into a multimodal solution. Pre-processing of the data was conducted, resulting in the acquisition of 1D vectors for classification. The authors also used a traditional machine learning model, SVM, for classification. The paper by Chong and Lee (2018) did not use any deep learning framework and instead used the MLP neural network as a **traditional machine learning model** for classification. The study by Bird, Ekárt, and Faria (2020) utilized a **DNN classifier** as well as **traditional machine learning techniques** such as SVM for comparison. The authors demonstrated that the proposed system is capable of accurately recognizing ASL and British Sign Language gestures.

2.5.6 Accuracy

The system by Naglot and Kulkarni (2016) attained an average mean classification accuracy of 94.44% on a 10-fold cross-validation dataset, outperforming previous models in the literature as shown in Table 2.5.6 below.

Table 2.5.6: Mean classification accuracy with RGB, LeapMotion and Multi-modality model (Naglot et al., n.d.)

Model	Sign Language Recognition Ability
<i>RGB</i>	88.14%
<i>Leap Motion</i>	72.73%
<i>Multi-modality</i>	94.44%

According to the Confusion Matrix plotted by the authors, Chong, and Lee (2018) shown in Figure 2.5.6, the proposed system achieved an accuracy of 96.15% in recognizing letters/alphabet using the trained ANN.

		Predicted Class																																	
		Sign	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y					Z	Total	TP	FN
Actual Class	A	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	B	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%
	C	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%
	D	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%
	E	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	2	83.33%	100%	
	F	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	G	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	1	90.90%	100%	
	H	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	I	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	J	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	K	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	L	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	M	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	2	0	0	0	0	0	0	10	8	2	2	80%	80%	
	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	10	0	0	100%	100%	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	1	0	0	0	0	0	0	10	9	1	0	100%	90%	
	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	10	10	0	2	83.33%	100%	
	T	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	6	0	0	0	0	0	0	0	10	6	4	0	100%	60%	
	U	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	8	0	0	0	0	0	0	10	8	2	3	72.72%	80%	
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	10	10	0	0	100%	100%	
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	10	10	0	0	100%	100%	
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	10	10	0	0	100%	100%	
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	10	10	0	0	100%	100%	
	Z	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	10	9	1	0	100%	90%
Total	10	10	10	10	12	10	11	10	10	10	10	10	10	10	10	10	10	10	10	10	12	6	11	10	10	10	10	9	260	250	10	10	96.54%	96.15%	

Figure 2.5.6: Confusion Matrix of Predicted class vs Actual class of system trained using the Back Propagation (BP) algorithm (Chong and Lee, 2018)

The DNN model by Bird, Ekárt, and Faria (2020) outperformed the SVM, with a mean accuracy of 90.58% and 85.65% for the 26-class and 36-class ASL recognition, respectively. The C6 feature group performed the best, with an accuracy rate of 93.81% and 88.79% for the 26-class and 36-class recognition using DNN. The sensitivity and specificity of each class were calculated, with the letters 'B', 'C', 'F', 'I', 'W', and 'Y' having the highest sensitivity rates, and 'H', 'S', and 'U' having the lowest confidence levels. The results indicated that DNN with the C6 feature group is an effective method for ASL recognition.

2.5.7 Complexity

The proposed system by Naglot and Kulkarni (2016) mentioned that LMC has a low computational **complexity** and can run in real-time on a standard laptop computer.

2.5.8 Advantages

After the evaluation of 3 papers which utilised the Leap Motion Controller, it is found that LMC has several **advantages** as a sensor for hand gesture recognition, such as high accuracy, non-invasiveness, and ease of use. It is also

relatively low-cost compared to other devices, making it more accessible for research and practical applications. Additionally, the ability to capture hand movements in 3D space allows for more natural and intuitive interactions with the system, making it an ideal choice for sign language recognition.

2.5.9 Disadvantages

However, there are also some **limitations** to consider when using the Leap Motion Controller for hand gesture recognition. Its limited field of view and range of detection can pose a challenge for sign language recognition. The device may also have difficulty distinguishing between movements that occur close to each other in space and may require a clear line of sight, which can be obstructed by clothing or other objects. Finally, the device's compatibility with certain operating systems and software platforms may also be limited.

The Leap Motion Controller (LMC) has proved to be an effective tool for developing real-time hand gesture recognition systems for interpreting sign language. Its ability to accurately track hand movements and capture fine-grained details has made it a popular choice among researchers.

2.6 Gloves with sensors

Hand gesture recognition using sensor gloves has gained significant attention in recent years due to their ability to capture the intricate movements of the hand. These gloves are equipped with sensors that capture hand movements and facilitating the transformation of sign language into text or speech. This section of the literature review will examine two papers, Shukor et al.'s "*A new data glove approach for Malaysian sign language detection*" (Shukor et al., 2015) and Lee and Lee's "*Smart wearable hand device for sign language interpretation system with sensors fusion*" (Lee and Lee, 2018) which propose novel approaches for gesture recognition using sensor gloves.

Shukor et al. developed a continuous sign language detection system using a data-glove-based approach using a tilt sensor and an accelerometer. The system can recognize both fingerspelling and sign gestures, making it suitable for deciphering Malaysian Sign Language. The system consists of a microcontroller, a Bluetooth module, a tilt sensor, and an accelerometer as shown in Figure 2.6.1.

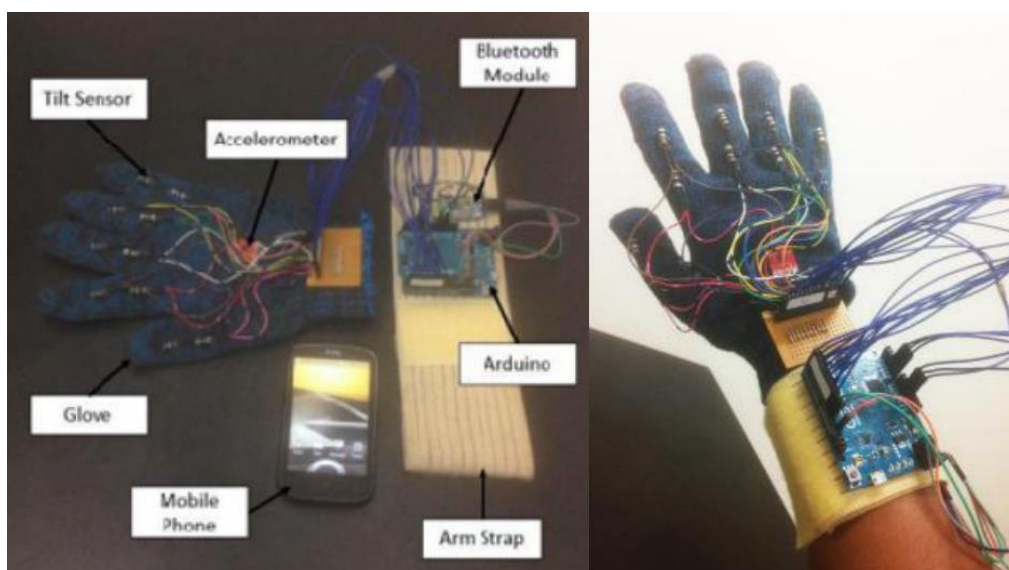


Figure 2.6.1: Assembly of data-glove (Shukor et al., 2015)

The flow chart of how it works is shown in Figure 2.6.2.

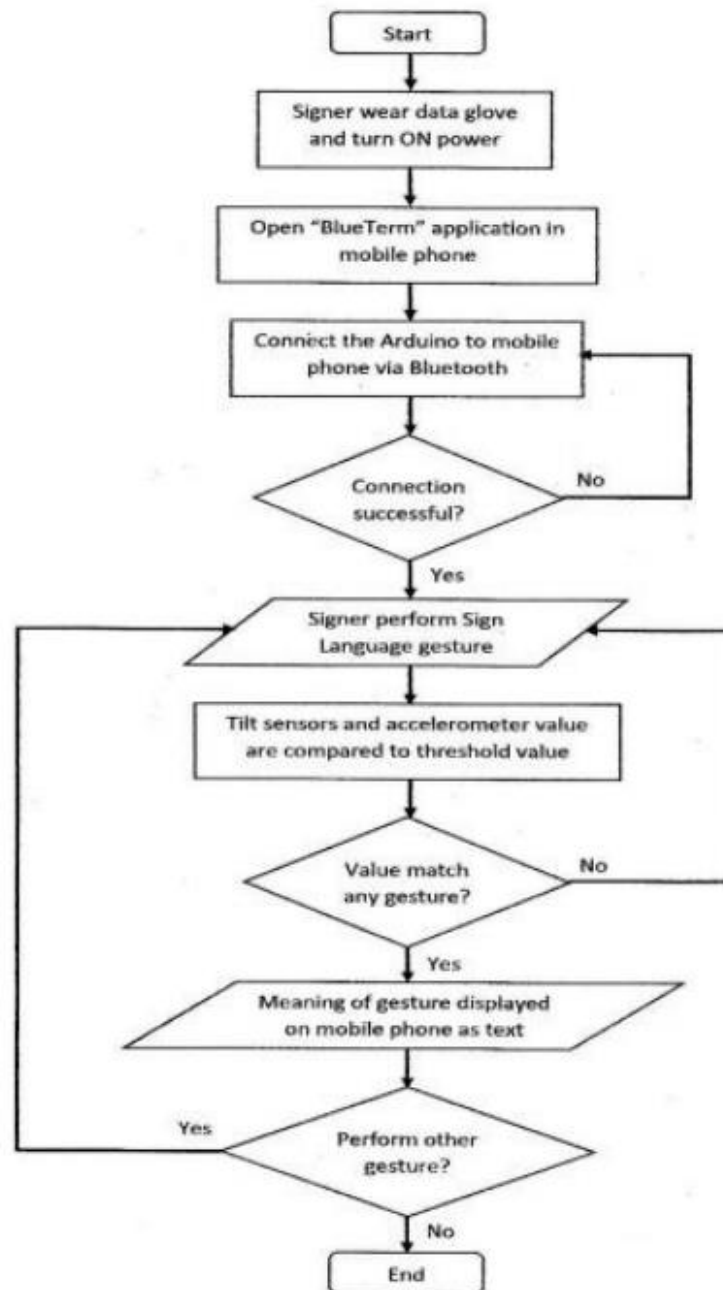


Figure 2.6.2: Flowchart of Sign Language Detection Algorithm (Shukor et al., 2015)

On the other hand, Lee and Lee's sign interpretation system involved the usage of a custom-made wearable device that utilizes flex sensors and an inertial motion unit, and an improved system with fusion of pressure sensor added to the middle finger as shown in Figure 2.6.3.

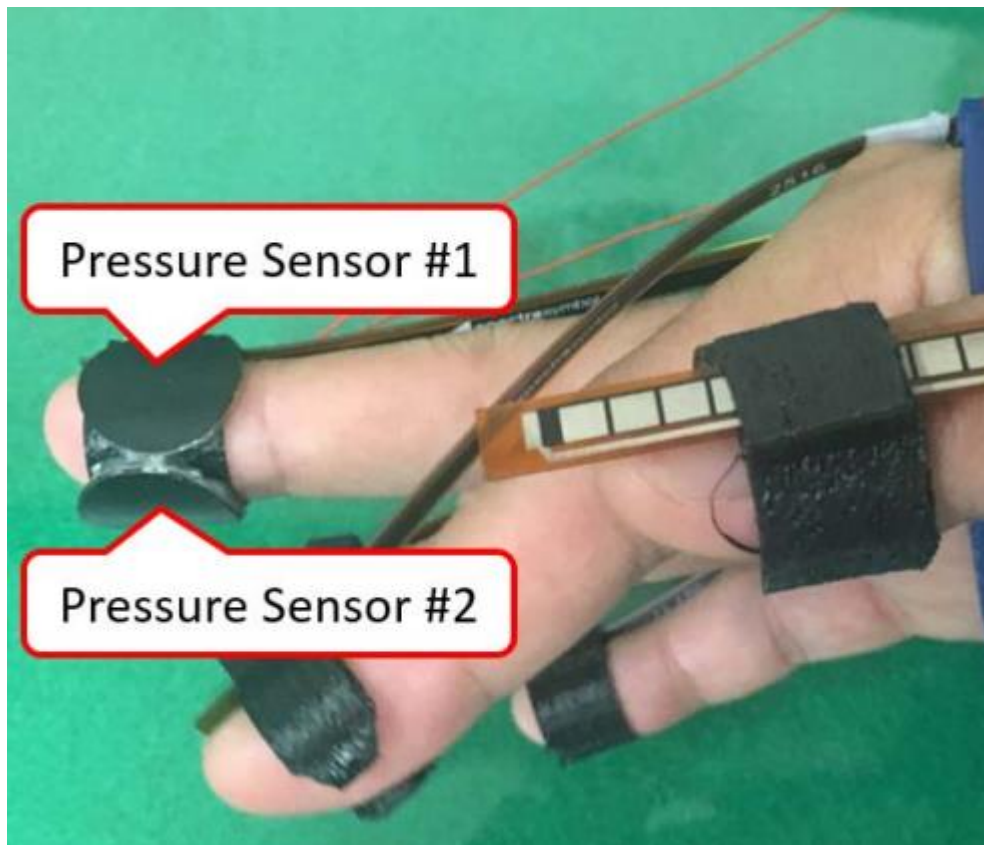


Figure 2.6.3: Improved system with Pressure Sensors (Lee and Lee, 2018)

Their system is divided into three modules: a sensor module, a processing module, and an application module as illustrated in Figure 2.6.4.

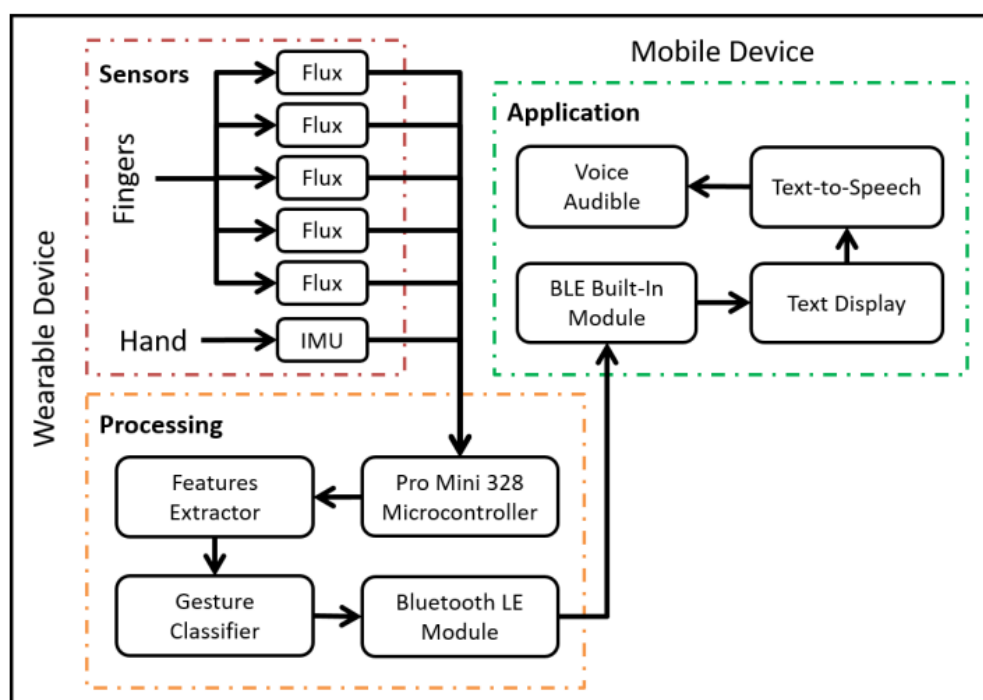


Figure 2.6.4: Overview of modules involved in sign interpretation system (Lee and Lee, 2018)

2.6.1 Pre-processing

In this study by Shukor et al., the **pre-processing** phase involves reading the tilt sensor (open or closed-circuit connection, which is read as digital inputs) and accelerometer inputs from the data glove and comparing them with stored memory in the microcontroller (for the few set data) when signer performs sign language. The pre-processing stage by Lee and Lee involves the processing of raw data, which includes filtering, normalization, and segmentation. The flex sensor and IMU data collected using an Arduino Pro Mini 328 are pre-processed. Throughout the experiments, the flexion values of the sensors varied among the different subjects due to variations in the hand sizes and the sensor placements. Therefore, a normalization step was performed to scale the data to a common range. Below is a normalization formula in Figure 2.6.5 involving where fs_i , \bar{fs} , and σ_{fs} , which are the i -th sensor reading, mean and SD of flex sensor value respectively.

$$\widehat{fs}_i = \frac{(fs_i - \overline{fs})}{\sigma_{fs}}$$

Figure 2.6.5: Normalization formula (Lee and Lee, 2018)

2.6.2 Segmentation

For **segmentation**, the algorithm used by Shukor et al. for continuous sign language detection is a fusion of tilt sensors and accelerometers. The accelerometer is required to detect gesture motion by setting a threshold in the axis of motion (x, y or z) since it is placed in the palm of the glove. In the study by Lee and Lee, the features extracted from the sensor data are the mean and standard deviation of each sensor signal. The extracted features are used to train the SVM classifier, which recognizes the hand gestures from the data.

2.6.3 Feature Extraction

During **feature extraction**, flex sensor and tilt sensors is used by Shukor et al. to detect the flexion of fingers, and the accelerometer captures the movements of the wrist and hand for each sign gesture. Flex sensor which is also known as bend sensor is a sensor that changes its resistance according to the amount of bend on the sensor. It is a passive resistance device fabricated by laying strips of carbon resistive elements within a thin flexible substrate. However, constructing the data glove with flex sensors can be quite costly because the price for flex sensor is quite expensive. Therefore, a tilt sensor or inclinometer is used to detect the bending of a finger, which is cheaper than flex sensor.

2.6.4 Dimension Reduction

Dimension reduction is used by Lee and Lee to reduce the number of features without losing important information. Principal component analysis (PCA) is used to reduce the dimensionality of the feature space while retaining most of the relevant information, such as the variance in the data. principal component analysis (PCA) algorithm for speeding up the processing time.

2.6.5 Classification

For the **classification** task, the gesture data is classified by Shukor et al. based on the features extracted from the sensors to recognize the sign language being performed. the classification process is done using a sign language detection algorithm, simpleSigner. The algorithm is a fusion between tilt sensors and accelerometers.



Figure 2.6.5: A tilt sensor at upright (vertical) position and a tilt sensor bent at 50 degrees(Shukor et al., 2015)

2.6.6 Deep Learning Model / Traditional Machine Learning Model

A built-in SVM classifier was used by Lee and Lee to classify the gestures into the 26 alphabet letters of ASL, a “neutral” state, and an invalid sign. With that, the SVM classifier classifies the input features into different categories and outputs the corresponding letter. The system by Lee and Lee uses a traditional machine learning approach that involves a built-in SVM classifier which is used to train the model using the extracted features.

2.6.7 Accuracy

The experiment by Shukor et al. involving a sign language detection system that uses tilt sensors and an accelerometer, was successful in detecting flexor motion using tilt sensors, and after fitting ten tilt sensors on a data glove, the system was tested on candidates performing sign language and gestures for alphabets, numbers, and words. The results showed that the system had a reasonably high **accuracy** ranging from 78.33% to 95% for all the tests, with higher accuracy for alphabets and numbers, and lower accuracy for words due to the involvement of motion that needed to be detected by the accelerometer. The accuracy of Lee and Lee’s system is measured by comparing the predicted output of the classifier with the actual output of the test data. The accuracy of the proposed system is discussed in detail. Table 2.6.7 summarizes the classification results for the first and second versions of the system.

Table 2.6.7: Classification accuracy 1st and 2nd version of sign recognition system (Lee and Lee, 2018)

Subject	Sample Size ((TP + TN) / Total)	AC (%)
1 st version	425,736 / 648,000	65.7
2 nd version	636,336 / 648,000	98.2

AC: Accuracy

It is observed that the accuracy of the first version was low, 65.7%, and there were negative classifications due to similar patterns occurring among several of the signs. The accuracy of the system improved significantly to 98.2% after the pressure sensors were added to the system. However, there were still minor misclassifications in the second version of the system. An analysis indicated that the incorrect pattern recognitions for all subjects occurred more commonly between the letter's "E" and "S." A similar issue appeared for letters "M" and "N" as well. Thus, the system misinterpreted the thumb region as a PR instead of a CR, or vice versa.

2.6.8 Complexity

In terms of complexity, the Lee and Lee's system revealed that classification would become more complex when there is a high accuracy of recognition for different signs. The complexity of the system is also discussed in the context of the pressure sensors. The accuracy rate of signs recognition for alphabet "U" increased significantly when two pressure sensors data are included for the classification. Likewise, the mean accuracy for alphabet "R" and "V" increased dramatically. The inclusion of the first pressure sensor surface showed significant differences for the signs between the letter's "U" and "V."

2.6.9 Advantages

Gloves with sensors provide significant **advantages** in the field of sign language interpretation. One of the most significant advantages is their ability to recognize both fingerspelling and sign gestures, making them an effective tool for deciphering Malaysian Sign Language. Additionally, sensor gloves are not affected by environmental factors such as lighting and offer higher accuracy compared to visual-based approaches. Moreover, these gloves are non-invasive and intuitive, making them easy to use and transportable. Their ability to provide

real-time recognition further increases their potential in applications that require real-time communication.

2.6.10 Disadvantages

However, sensor gloves also have several **disadvantages**. One of the major drawbacks is the high cost of constructing the data glove with flex sensors. Furthermore, reading flex sensors is not very stable and is sensitive to noise, which may affect the accuracy of the system. The gloves may not be suitable for all users due to variations in hand size and sensor placement. The sensors' accuracy may also be affected by the user's hand movements, making regular calibration necessary to maintain accuracy.

In conclusion, gloves with sensors have shown great potential for recognizing sign language, and significant research efforts have been directed towards improving their performance.

2.7 Skeleton Aware Multimodal SLR framework (SAM-SLR)

SAM-SLR is a machine learning algorithm that can help computer programs better understand complex data by identifying patterns and relationships between different parts of the data. In the development of the recognition system to interpret sign language, the use of the SAM-SLR framework has been identified as an effective approach. In this literature review, we will take a closer look at two papers, namely Jiang et al.'s "*Skeleton aware multi-modal sign language recognition*" and De Coster et al.'s "*Isolated sign recognition from RGB video using pose flow and self-attention*" on their processes of pre-processing, segmentation, feature extraction, dimension reduction, classification, and the use of deep learning frameworks involved.

The first paper by Jiang et al. (2021) introduces the Skeleton Aware Multimodal Sign Language Recognition (SAM-SLR) framework, which is a multimodal approach for sign language recognition. The SAM-SLR framework utilizes two models called SL-GCN and SSTCN for skeleton keypoints and features, respectively, in addition to a 3D CNNs model for other modalities. The defined SAM-SLR framework is as shown in Figure 2.7.

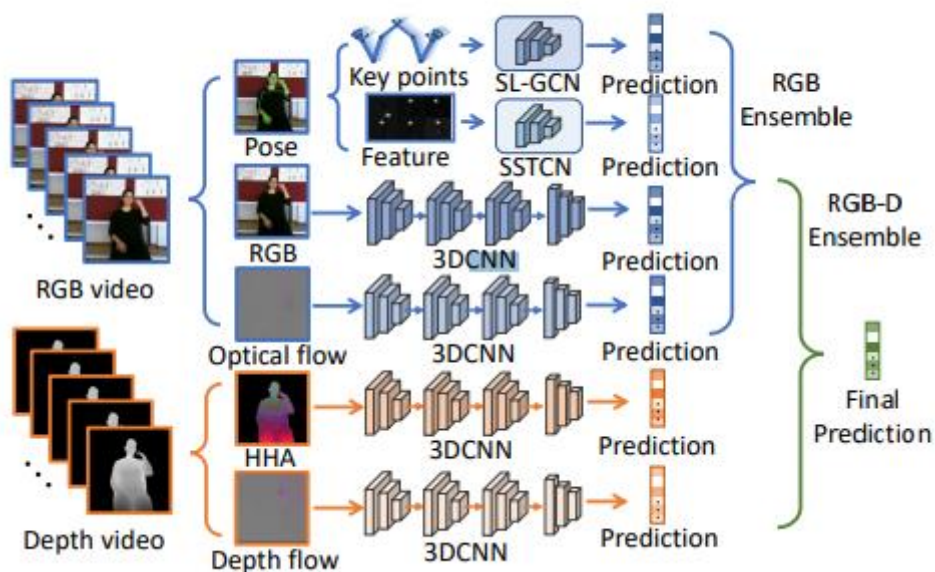


Figure 2.7: Concept of Skeleton Aware Multi-modal Sign Language Recognition Framework (SAM-SLR)

As visualized above, the framework involves using various inputs such as pose, optical flow, HHA, and depth flow data to capture different aspects of the hand movements, with inputs to train models such as the SL-GCN, SSTCN, 3DCNN, and RGD Ensemble. The second paper, De Coster et al. (2021) proposed an isolated sign recognition system using the Skeleton Aware Multimodal SLR (SAM-SLR) framework on the AUTSL dataset. The framework utilizes pose flow and self-attention mechanisms for the recognition of isolated signs from RGB videos.

2.7.1 Pre-processing

The **pre-processing** step by Jiang et al. involves using a pretrained whole-body pose estimation network to provide 133 keypoints estimated from the detected person in videos (Jiang et al., n.d.). A spatio-temporal graph can then be constructed by connecting the adjacent keypoints in the spatial dimension according to the natural connections of the human body and connecting all keypoints to themselves in the temporal dimension as shown in Figure 2.7.1 below.



Figure 2.7.1: RGB with whole-body keypoints overlay

On the second paper, the dataset De Coster et al. used in the experiment is the balanced AUTSL dataset which consists of 36,302 samples (De Coster et al., n.d.). The data comprises 226 distinct signs, each captured by one of 43 individuals. The dataset has been divided into independent sets for training, validation, and testing, considering variations in signers, filming locations, and viewpoints. The video samples are recorded at a resolution of 512 by 512 pixels

and a frame rate of 30 frames per second (FPS), encompassing both RGB and depth data. However, this study exclusively focuses on utilizing the RGB data for experimentation purposes.

Hand cropping is performed to extract hand images as main inputs to the model. The OpenPose BODY-135 model is used to estimate keypoints (VTN-PF) for the body, hands, face, and feet as shown in Figure 2.7.2.

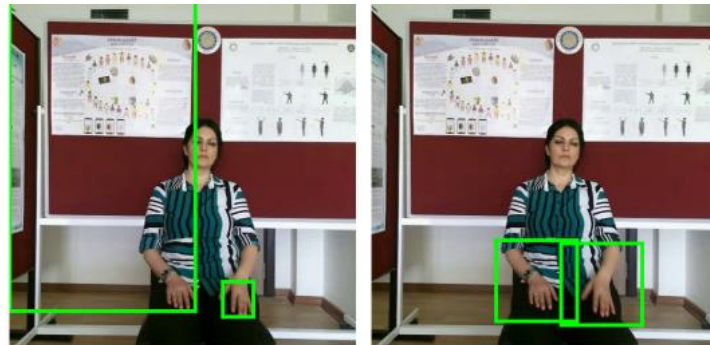


Figure 2.7.2: Cropping based on hand keypoints by OpenPose may result in distorted crops in cases where certain keypoints are not detected (De Coster et al., n.d.)

Hand cropping involves identifying an appropriate position for the hand crop extension from the forearm. This determination relies on the positioning of elbow and wrist keypoints. The crop size is selected to maintain relative consistency, accounting for variations in camera distance and individual physical attributes.

2.7.2 Segmentation

The **segmentation** process by Jiang et al. involves reducing the graph size from 133 nodes to 27 nodes, which contain the essential information required for SLR. This step also results in faster model convergence and significantly higher recognition rates. During the segmentation process, the samples utilized in the work by De Coster et al. (Figure 2.7.2.1) have diverse lengths, with a median duration of 61 frames.

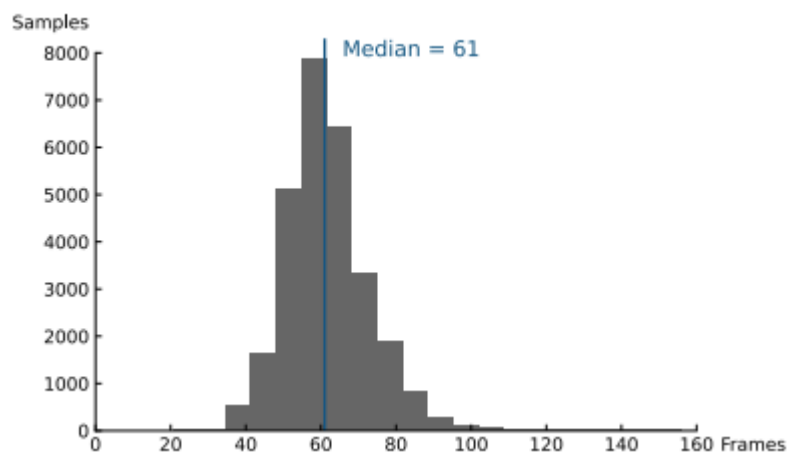


Figure 2.7.2.1. The depicted samples exhibit differing lengths, with a median duration of approximately 61 frames, equivalent to about 2 seconds, within the training set

Each video sample contains initial wind-up and final wind-down segments. However, for the purpose of isolated sign recognition within this dataset, these wind-up and wind-down segments are excluded from consideration. Instead, a segment is chosen from the middle of the video, and a selection of 16 frames is made with a stride of 2 frames, resulting in an effective temporal receptive field of 32 frames. It's important to note that the wind-up and wind-down segments are not taken into account during the segment selection process.

2.7.3 Feature Extraction

The **feature extraction** process by Jiang et al. involves using spatio-temporal GCN with spatial partitioning strategy to model the dynamic skeletons(Jiang et

al., n.d.). This is done by adopting a decoupling graph convolution to boost the capacity of GCN. The spatial and temporal GCN are implemented as performing standard 2D convolution and then multiplying the results by a trainable weight matrix of the convolution. In isolated sign recognition, De Coster et al. identified significant parameters for recognition, including hand shape, orientation, movement, and place of articulation. They also acknowledged the importance of non-manual components like mouthings, eye gaze, and eyebrow movements in sign languages. However, for the specific task of isolated sign recognition, non-manual components are considered less crucial. In the AUTSL dataset, videos have a spatial resolution of 512 by 512 pixels. Considering that the model is trained using inputs of 224 by 224 pixels, there's a notable spatial down-scaling of the inputs. As a result, a pre-processing step involves cropping out hand images (VTN-HC) and utilizing them as the primary inputs to the model, thus preserving more spatial information pertaining to the hand areas.

2.7.4 Classification

The **classification** process is done by Jiang et al. using a multi-modal ensemble process (Jiang et al., n.d.), which combines the outputs of the SL-GCN and SSTCN models for skeleton keypoints and features, respectively, and the 3D CNNs model for other modalities. The ensemble process involves a weighted voting mechanism to obtain the final prediction. For classification by De Coster et al., a deep learning framework is used. A VTN, or Vision Transformer Network, is applied to model both spatial and temporal information through the utilization of deep Convolutional Neural Networks (CNNs) for spatial data and self-attention mechanisms for temporal data. Several enhancements have been introduced to improve the performance and capabilities of the VTN.

2.7.5 Deep Learning Model / Traditional Machine Learning Model

The **deep learning framework** used in the SAM-SLR framework by Jiang et al. includes the use of artificial neural networks in the SL-GCN and SSTCN models. The traditional machine learning approach involves hand-engineered features that are used in the 3D CNNs model for other modalities. Deep learning framework is also used by Coster et al., specifically a VTN. The model uses artificial neural networks. The VTN (Vision Transformer Network) is employed to model spatial information through deep Convolutional Neural Networks (CNNs) and temporal information through self-attention mechanisms. The VTN is improved with several modifications. The deep learning model used in the classification is a traditional machine learning method which involves hand-engineered features.

2.7.6 Accuracy

The evaluation of the proposed approach is done by Jiang et al. using different modalities, and the performance of the models is measured using the top-1 and top-5 accuracy. The results of the experiments show that the proposed SAM-SLR framework outperforms the baseline methods in terms of **accuracy**. Specifically, the multi-stream SL-GCN model achieved the highest top-1 accuracy of 95.45% on the validation set, while the top-1 accuracy of the baseline RGB and RGB-D models were 49.23% and 62.03%, respectively.

Table 2.7.6: Performance of multi-stream SL-GCN(De Coster et al., n.d.)

Streams	Top-1	Top-5
Joint	95.02	99.21
Bone	94.70	99.14
Joint Motion	93.01	98.85
Bone Motion	92.49	98.78
Multi-stream	95.45	99.25

Table 2.7.7: Performance baseline results RGB and RGB-D(De Coster et al., n.d.)

	Finetune	Track	Top-1
Baseline	-	RGB	49.23
Baseline	-	RGB-D	62.03

The results demonstrated that the proposed approach using different modalities can improve the overall recognition rate of sign language recognition. In the case of Jiang et al., they utilized the categorical cross-entropy loss function for all three experiments. The final model was chosen based on the lowest loss observed. The authors conducted a comparative analysis of the three models, and according to Figure 2.7.6, the VTN-PF demonstrated the highest accuracy at 91.51% on the validation set.

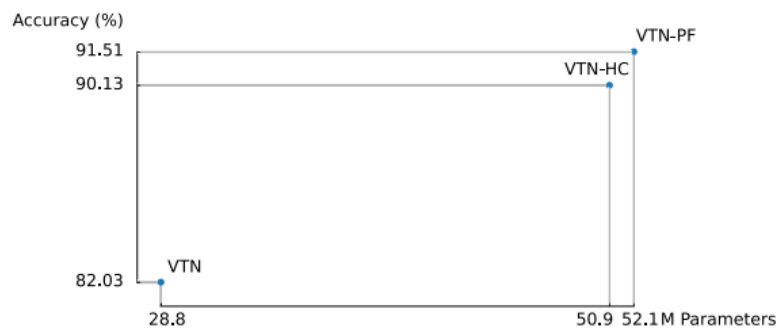


Figure 2.7.6: Graph Accuracy against Parameters for 3 experiments: VTN, VTN-HC, VTN-PF

They also compared the number of trainable parameters between the models, and the VTN-PF had slightly more trainable parameters than the VTN-HC model.

2.7.7 Complexity

According to Jiang et al., the proposed approach is relatively complex as it involves the use of multiple modalities, such as whole-body pose keypoints and features, RGB frames, depth, masked HHA, optical flow and depth flow as shown in Figure 2.7.7 below.

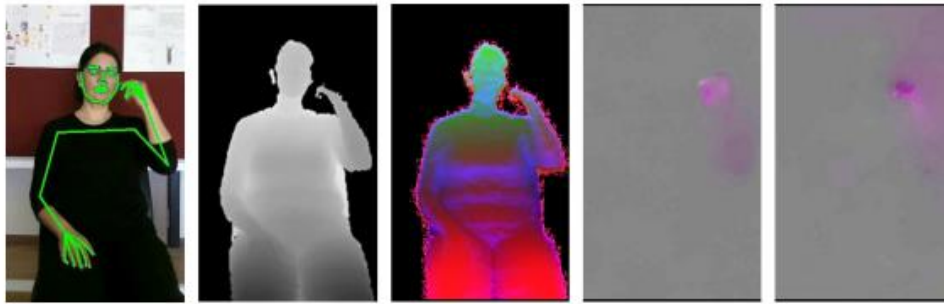


Figure 2.7.7: Visualization of modalities: RGB frames, depth, masked HHA, optical flow and depth flow (left-right) (Jiang et al., n.d.)

Each modality requires specific processing steps and data augmentation techniques to extract the necessary features. For instance, the whole-body pose keypoints are extracted using a pretrained HRNet whole-body pose estimator, and then processed into four streams for joint, bone, joint motion, and bone motion. Similarly, RGB frames and optical flow are extracted from the videos using the TVL1 algorithm and then cropped and resized to 256x256 based on the keypoints. The depth HHA features are also extracted from depth videos using a mask to fill out the missing regions. Moreover, the proposed approach involves the use of complex models, such as the multi-stream SL-GCN model as shown in Figure 2.7.7.1 below, which requires careful design and optimization to achieve high accuracy.

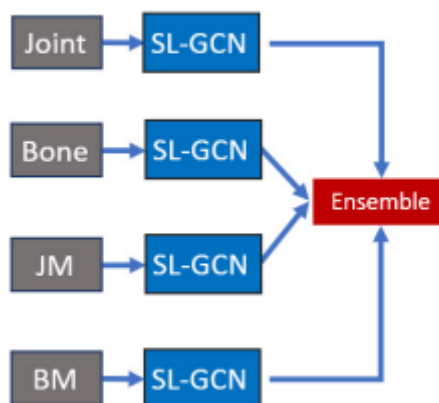


Figure 2.7.7.1: Multi-stream SL-GCN model (Jiang et al., n.d.)

According to Coster et al., the complexity of the model depends on different approaches that were taken to improve the accuracy of the models. In the VTN-HC model, the hands were cropped based on wrist positions identified using OpenPose. This cropping process led to an augmentation in the embedding size within the self-attention decoder model, reaching 1024. Consequently, this adjustment rendered the VTN-HC model more intricate compared to the original VTN model.

2.7.8 Advantages

The use of the SAM-SLR framework in hand gesture recognition **offers** high accuracy, making it ideal for applications that require precision. The framework uses deep learning techniques that allow for the recognition of complex gestures and the ability to distinguish between similar gestures. Additionally, the use of multiple models for classification increases the model's robustness and accuracy, making it suitable for a wide range of applications.

2.7.9 Disadvantages

One major **disadvantage** of using the SAM-SLR framework is the high computational cost associated with training and inference. Due to the use of deep learning and the combination of multiple models, the model requires a significant amount of computational power and resources. This can make it challenging to implement the framework on low-power devices such as smartphones and embedded systems. Additionally, the complexity of the framework can make it challenging to understand and modify for researchers who are not familiar with deep learning techniques.

By examining their research on the use of SAM-SLR, one can gain valuable insights into the current state of the art in hand gesture recognition systems. Further research in this area will potentially lead to more accurate and efficient sign language recognition systems, with potential applications in various fields, including assistive technologies and communication devices for the hearing-impaired.

2.8 Summary

This table 2.8 below provides an overview of different approaches to hand gesture recognition. Each approach is evaluated based on the methods used for pre-processing, segmentation, feature extraction, dimension reduction, and classification. The accuracy of each approach is also provided.

Approaches evaluated include OpenCV only, OpenCV with Mediapipe, LeapMotion Controller with CNN, Gloves with sensors, and Skeleton Aware Multimodal SLR framework.

Table 2.8: Overview of different approaches to hand gesture recognition

Approach	OpenCV only	OpenCV with Mediapipe	LeapMotion Controller with CNN	Gloves with sensors	Skeleton Aware Multimodal SLR framework
Process					
Pre-processing Methods	Haar-cascade classifier for hand detection	Transformation of image dataset into a hand landmark (HL) position dataset using MediaPipe Hands	1. Addressing empty frames in instances where the sensor failed to detect either hand.	1. Read tilt sensor and accelerometer inputs, compare with stored memory	1. Not specified 2. Not specified

			<p>2. Normalizing the dataset before training and testing.</p> <p>3. The data collected from was pre-processed to extract features.</p>	<p>2.Filtering, normalization, and segmentation</p>	
--	--	--	---	---	--

<p>Segmentation Methods</p>	<p>Calculation of space consumption between the convex hull and contour of the hand</p>	<p>Hand Tracking Module (HTM) consisting of a Palm Detector and HL Position Extractor</p>	<p>1. Not specified 2. Palm and Finger dataset were selected as features for the feature extraction process. 3. Not specified</p>	<p>1. Not specified 2. Mean and standard deviation of each sensor signal</p>	<p>1. Not specified 2. Choosing a segment from the centre of the video and extracting 16 frames with a step size of 2 frames, achieves an effective temporal receptive field spanning 32 frames.</p>
------------------------------------	---	---	---	--	--

<p>Feature Extraction Methods</p>	<p>Region of Interest (ROI) calculation for appearance and gesture detection</p>	<p>Extraction of position data of the landmarks</p>	<p>1. Extracted Note: 3D space (X, Y, Z coordinates)</p> <ul style="list-style-type: none"> - The start and end positions of the arm in 3D space - The angle in 3D space between the initial and final arm positions. - The speed or velocity of the arm in all three dimensions. 	<p>1. Not specified</p> <p>2. Mean and standard deviation of each sensor signal</p>	<p>1. Not specified</p> <p>2. Classification of significant parameters such as hand shape, orientation, movement, and place of articulation</p>
--	--	---	--	---	---

			<ul style="list-style-type: none">- The position of the elbow and wrist in 3D space.- The pitch, yaw, and roll of the palm.- 3D angle of each finger and each bone in the hand. <p>2. Extracted the Euclidean distances between the consecutive fingertip position to palm</p>		
--	--	--	--	--	--

			position and the Euclidean distances between the fingertip position of each consecutive finger. 3. Not specified		
Dimension Reduction Methods (if used)	Not specified	Not specified	Not specified	1. Not specified 2. Principal Component Analysis (PCA)	Not specified

<p>Classification Methods</p>	<p>Haar-cascade classifier for gesture recognition</p>	<p>The models utilized in this study include the Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Support Vector Machines (SVM), and K-Nearest Neighbours (KNN)</p>	<p>1. MLP, DEvoMLP, CNN, Dense Interpretation Network, Image Classification Network, and Bone Data Classification Network. 2. Artificial Neural Network (ANN). 3. Leap Motion Controller (LMC) through late fusion using transfer learning</p>	<p>1. simpleSigner, Sign Language Detection Algorithm (fusion of tilt sensors and accelerometers) 2. SVM Classifier</p>	<p>1. SL-GCN, SSTCN, 3D CNN, and RGD Ensemble 2. VTN for sign classification</p>
--------------------------------------	--	--	--	--	---

Deep Learning Framework	Not specified	Not specified	1. TensorFlow, Keras 2. TensorFlow 3. TensorFlow	Not specified	Not specified
Traditional Machine Learning/ Deep Learning Model	Traditional Machine Learning	Traditional Machine Learning, Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Support Vector Machines (SVM), K-Nearest Neighbours (KNN)	1. Artificial Neural Network (ANN) using the Euclidean distances between the consecutive fingertip position to palm position and the Euclidean distances between the fingertip position of each consecutive finger. 2. Artificial Neural Network (ANN)	1. Traditional Machine Learning 2. Built-in SVM classifier	Deep Learning Model

			3. Convolutional Neural Network (CNN)		
Accuracy	Recognizes all 10 hand gestures with 10 hits out of 10 trials	<p>DTC: Training data accuracy 100%, Test data accuracy 96.4%</p> <p>RFC: Training data accuracy 100%, Test data accuracy 97.2%</p> <p>SVM: Achieving an accuracy of 97.9% with the test data using a radial base kernel.</p> <p>KNN: Training accuracy 97.4%, Test accuracy 98.1%</p>	<p>1. 95%</p> <p>2. 90.3%</p> <p>3. 96%</p>	<p>1. 78.33% to 95%</p> <p>2. 65.7% to 98.2%</p>	<p>1. Multi-stream SL-GCN model (top-1 accuracy) 95.45% on the validation set, baseline RGB and RGB-D (top-1 accuracy) models were 49.23% and 62.03%</p> <p>2. VTN-PF: 91.51% on the validation set</p>

Complexity	Low. Computationally efficient and easy to implement	High, requires technical expertise in parameter tuning, and algorithms for accurately identifying hand landmarks from images.	High	Low	High
Advantages	<ul style="list-style-type: none"> - Computationally efficient - Easy to implement without complex deep learning models 	<ul style="list-style-type: none"> - High accuracy in palm detection and hand landmark extraction 	<ul style="list-style-type: none"> - High accuracy - Non-invasiveness - Ease of use - Relatively low-cost - Captures hand movements in 3D space for 	<ul style="list-style-type: none"> - Recognizes fingerspelling and sign gestures - Unaffected by environmental factors such as lighting - Higher accuracy 	<ul style="list-style-type: none"> - High accuracy - Recognition of complex gestures - Ability to distinguish between similar gestures

			natural interactions	compared to visual-based approaches - Non-invasive and intuitive	- Increased robustness and accuracy
Disadvantages	<ul style="list-style-type: none"> - May not perform well in complex and noisy environments - Dependent on factors like clean background and proper lighting - Inconsistent detection and interpretation of 	<ul style="list-style-type: none"> - High complexity of the system - Dataset may contain invalid entries that need to be removed. 	<ul style="list-style-type: none"> - Limited field of view and range of detection - Difficulty distinguishing between movements close in space - Requires clear line of sight 	<ul style="list-style-type: none"> - High cost of construction - Reading flex sensors is not very stable and sensitive to noise - May not be suitable for all users due to variations in hand size and 	<ul style="list-style-type: none"> - High computational cost for training and inference - Challenging to implement on low-power devices - Complexity can make it challenging to

	<p>hand gestures with OpenCV</p> <ul style="list-style-type: none"> - Frameworks like Mediapipe are suggested for more reliable and accurate detection. 		<p>which can be obstructed</p> <ul style="list-style-type: none"> - Compatibility with certain operating systems may be limited. 	<p>sensor placement</p>	<p>understand and modify for non-experts in deep learning.</p>
Authors	<ol style="list-style-type: none"> 1. Ahmad Puad Ismail et al. 2. Riaz Sulaimi 	<p>C. M. Suryateja et al.</p>	<ol style="list-style-type: none"> 1. Naglot and Kulkarni (2016) 2. Chong and Lee (2018) 3. Bird, Ekárt, and Faria (2020) 	<ol style="list-style-type: none"> 1. Shukor et al. 2. Lee and Lee 	<ol style="list-style-type: none"> 1. Jiang et al. 2. De Coster et al.

The table 2.8 above has shown that there are several approaches available for gesture recognition using hand landmarks. Among these approaches, using **OpenCV with Mediapipe** has shown promising results for the real-time sign language.

Firstly, **this approach** provides a real-time and efficient solution for hand landmark detection, which is crucial for interpreting sign language in real-time. The combination of these tools offers a robust and accurate hand landmark extraction method, which can be used to create a large dataset for training and testing machine learning models.

Secondly, the OpenCV with Mediapipe approach has shown better results compared to other methods, such as Haar-cascade classifiers, which are not robust to variations in hand size, orientation, and lighting conditions. This means that the approach can accurately detect hand gestures, even in challenging environments.

In summary, using OpenCV with Mediapipe provides a practical and efficient solution for developing a recognition system to interpret sign language. This approach will be integrated into the workflow for developing a recognition system for sign language interpretation in the next chapter.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 The Proposed Model Workflow

In Chapter 1, a proposed workflow was presented for developing a hand gesture recognition model using deep learning techniques. In this chapter, we will delve **deeper** into the training and testing phases of the proposed workflow and provide detailed diagrams for each process.

The workflow includes data pre-processing, segmentation, feature extraction, dimension reduction, and classification for both the **Training and Testing phases**. This section will introduce the proposed model workflow and explain the steps involved in developing a recognition system. The workflow diagrams are categorized into three parts: **Training Phase, Test Phase, and Common Phases**, with **individual diagrams** for Pre-processing in Training and Test Phase, and Common Phases containing processes for Feature Extraction, Segmentation, Dimension Reduction, and Classification.

The pre-processing, segmentation, feature extraction, dimension reduction, and classification processes are derived from this main workflow diagram.

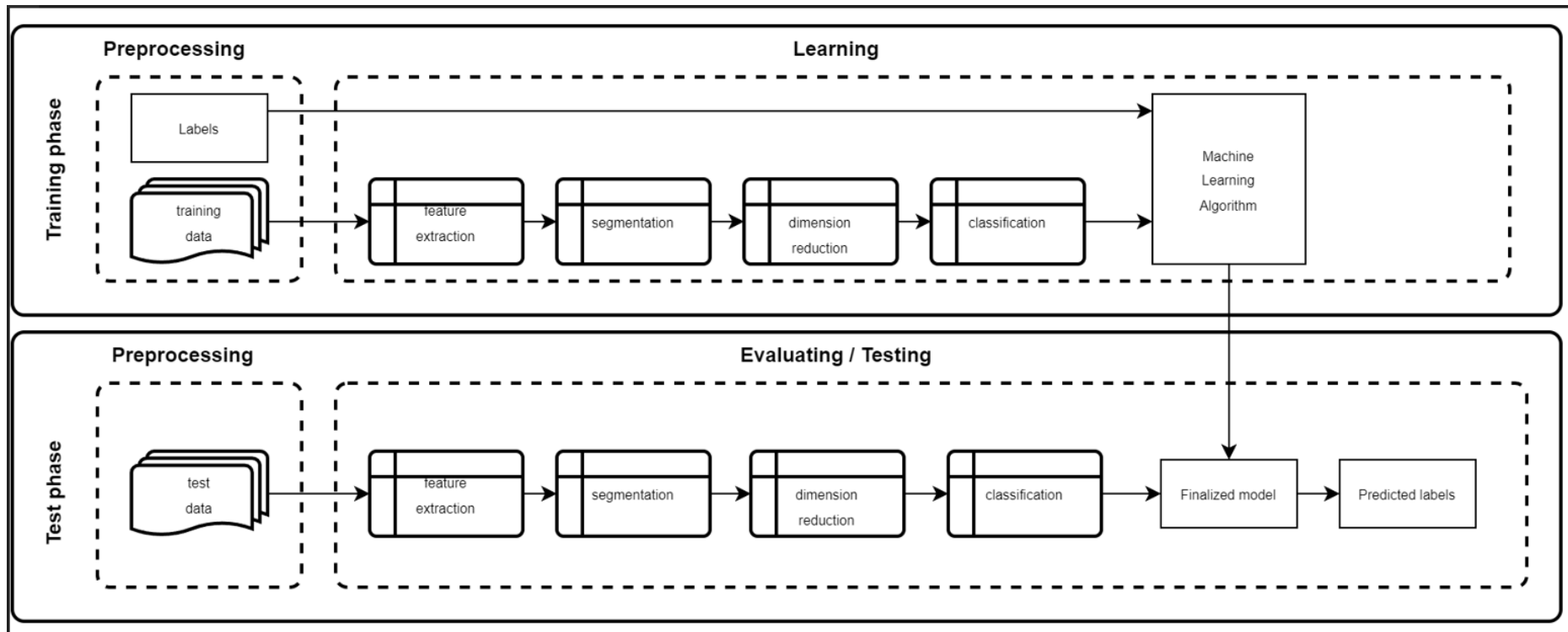


Figure 1.6.1: Workflow Diagram for the Real-Time Hand Gesture Recognition System To Interpret Sign Language from Chapter 1

3.2 Pre-processing

3.2.1 Training Phase

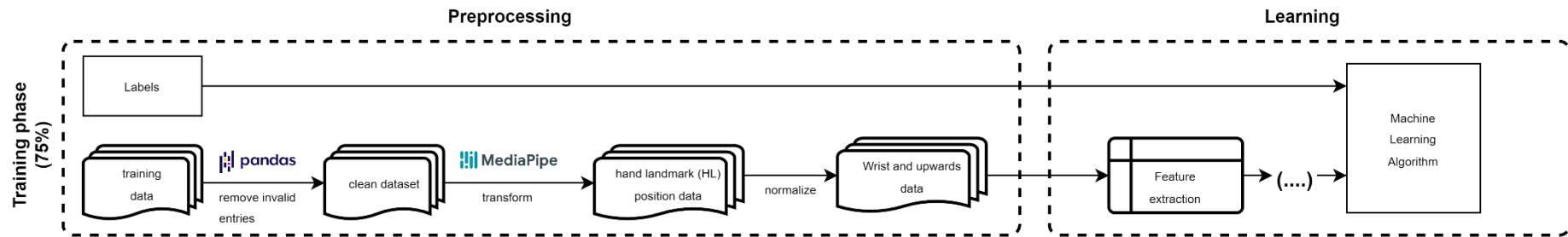


Figure 3.2.1: Workflow Diagram for Pre-processing (Training)

3.2.2 Test Phase

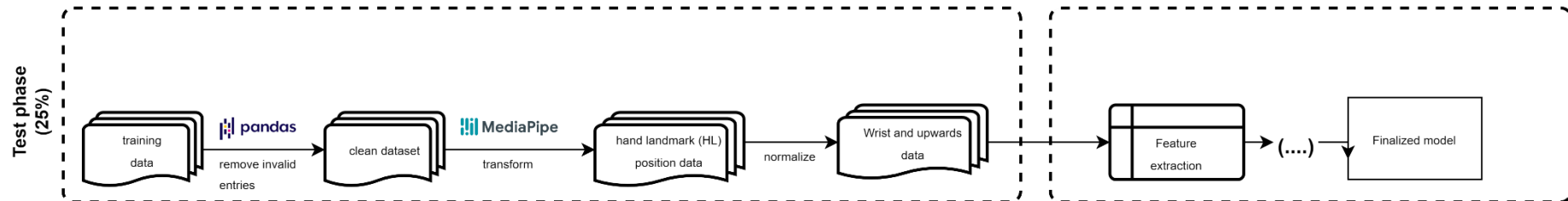


Figure 3.2.2: Workflow Diagram for Pre-processing (Test)

Pre-processing is a crucial step in the proposed model workflow for developing a recognition system to interpret sign language. The goal of pre-processing is to prepare the raw data (images) for the subsequent stages of the system, such as feature extraction and classification.

The first step in the pre-processing stage is to acquire a labelled dataset consisting of static hand gestures, 10 letters (i) A, (ii) E, (iii) H, (iv) I, (v) L, (vi) N, (vii) O, (viii) S, (ix) T, (x) U and moving hand gestures, 7 vocabularies in American Sign Language, namely (i) Best, (ii) Birthday, (iii) Please, (iv) Happy, (v) Hearing, (vi) Like and (vii) Feel. These datasets are collected whenever a user train the system by pressing 'k' for keypoint classifier (static hand gesture) followed by the data label number (0-9). As for moving hand gestures are collected when user train the system by pressing 'h' for point history classifier (moving hand gestures) followed by the data label (0-6). This process involved conversion of images to appropriate input format in the form of RGB matrices.

In the notebook, the paths are specified for the dataset and the model files where data will be loaded, which later loaded as features (`X_dataset`) and labels (`y_dataset`) by NumPy.

The dataset includes 21511 instances of static hand gestures (keypoints) and 44953 instances of moving hand gestures (point history), which are split into **75% for training and 25% for testing**. Meanwhile, the number of classes are defined, with 10 for static hand gestures, and 7 for moving hand gestures in the classification task. Necessary libraries and modules such as `csv`, `numpy`, `TensorFlow` and more are imported.

Once the dataset is acquired, the next step is to remove any invalid data entries. This is done using the Python library **Pandas**, which provides functions for data cleaning and pre-processing. Invalid data entries, such as missing or corrupted data are removed, as they can adversely affect the performance of the system.

The next step is to transform the clean dataset into Hand Landmark Position data (HL data) using **Google's Mediapipe framework**. Mediapipe is an open-source framework that provides a comprehensive set of tools for building real-time multimodal applications, including hand tracking and gesture recognition. HL data consists of the **3D coordinates of 21 hand landmarks**, which are used to represent the hand gestures.

Upon obtaining the HL data, the next step is to **normalize the data** to extract only the wrist and upwards of the signer's hand. This is done to reduce the complexity of the data and to focus only on the relevant features for gesture recognition. This step involves scaling and shifting the data to a common reference frame, such as the origin and orientation of the camera.

Once the pre-processing stage is complete, the resulting pre-processed data is ready for the **feature extraction** stage. Feature extraction involves identifying relevant features from the data that can be used for classification, such as the angles and distances between hand landmarks. The extracted features are then used to train a machine learning model for gesture recognition.

3.2.3 Common Phases

3.3 Feature Extraction

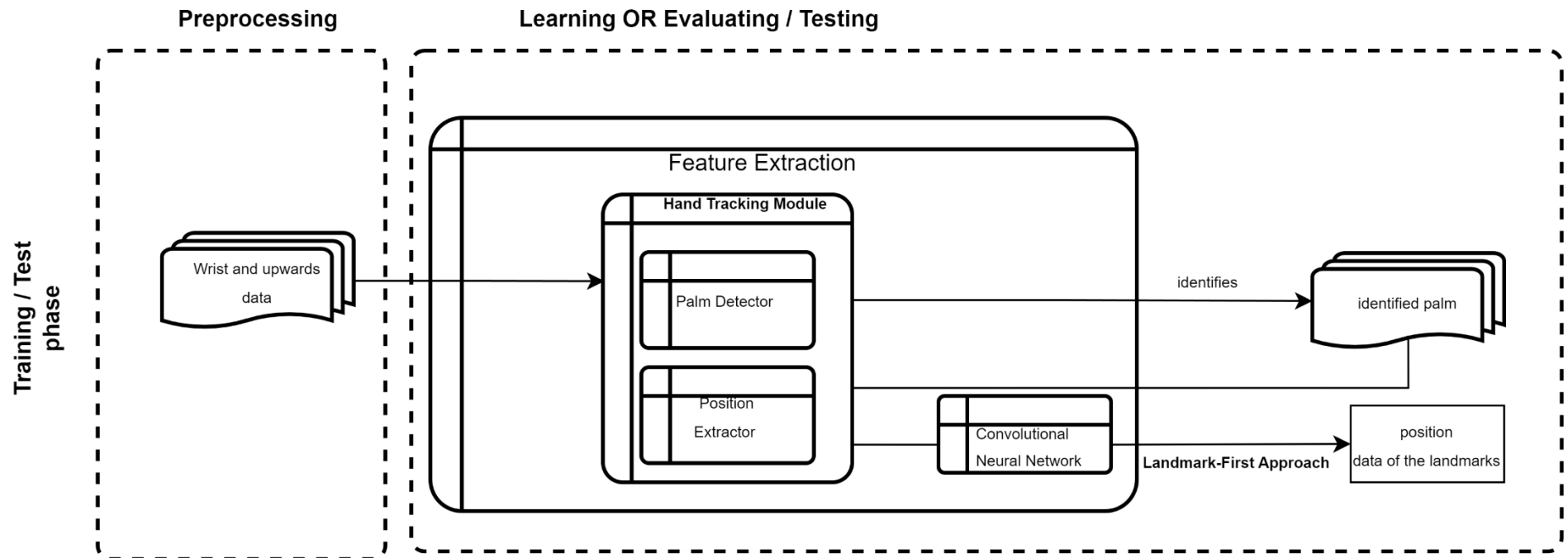


Figure 3.3.1: Workflow Diagram for Feature Extraction

The next process is the feature extraction stage, which involves the extraction of relevant features from the pre-processed data to be used for classification. In the context of our system, the facial landmarks, hand poses, and body poses are extracted using a **MediaPipe holistic model**.

In this proposed model workflow, the Feature Extraction stage involves the wrist and upwards data (as RGB matrices) passing through the hand tracking module, followed by the Convolutional Neural Network (CNN) to extract hand landmarks.

The hand tracking module contains two main components: the Palm Detector and the Position Extractor. The first component, the **Palm Detector**, is responsible for identifying the palm from the pre-processed wrist and upwards data. This aspect holds significance as the palm serves as a crucial reference point for determining the position of hand landmarks. The utilization of machine learning algorithms are to detect the palm in the input image, and then crops the image to focus only on the area around the palm.

The second component, the **Position Extractor**, uses the identified palm to obtain the position data of the landmarks through *Landmark-First Approach*. This involves identifying the hand landmark with machine learning technique known as **Convolutional Neural Network (CNN)**. Below describes how CNN works in identifying the hand landmark after the palm is cropped.

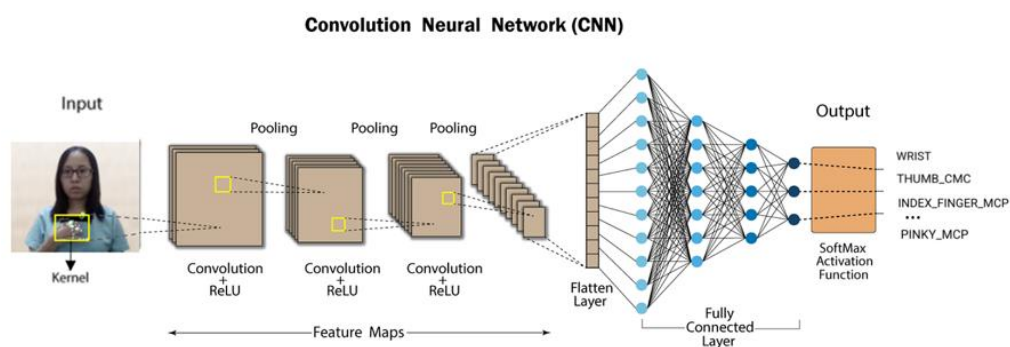


Figure 3.3.2: Convolutional Neural Network (CNN) in hand landmark identification

Based on the above, the **input** to the CNN is a region of interest containing the hand or palm. CNNs consist of multiple **convolutional layers** that apply filters (small grids) to the input image, detecting patterns and features associated to the hand landmarks. Along each convolutional operation, an activation function (**ReLU - Rectified Linear Unit**) is applied to introduce non-linearity and help the network learn complex patterns. **Pooling layers** reduce the spatial dimensions of the feature maps produced by the convolutional layers, reducing computation and makes the network more manageable. After pooling, the feature maps are **flattened** into a vector. This vector is then fed into **fully connected layers**, where each neuron is connected to every neuron in the previous layer. The **output layer** of the CNN predicts the positions of hand landmarks with Softmax activation function, as shown below.

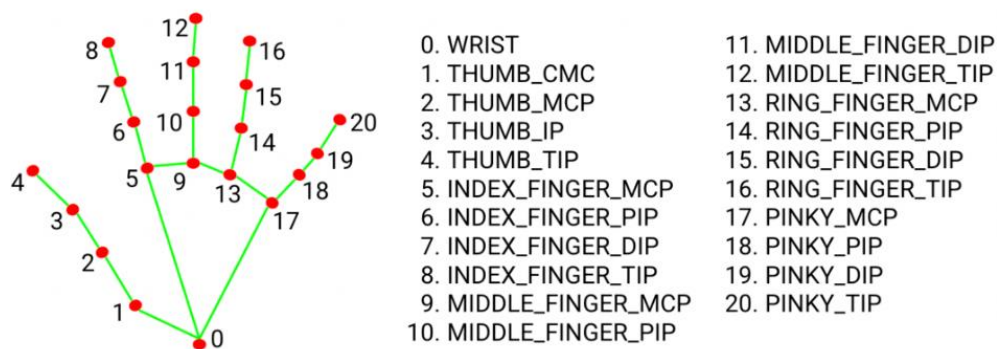


Figure 3.3.3: Finalized extracted hand landmarks

Once the landmarks are identified, their positions are extracted and represented as a set of features. These features may include the angles and distances between the landmarks, which can be used for classification.

Upon completion of the Feature Extraction stage, the identified palm and position data of landmarks are brought to the **Segmentation phase**. The Segmentation phase involves dividing the hand gestures into individual signs and separating the signs from the background and any other objects in the image. This is important for accurate recognition of individual signs, and for improving the overall performance of the system.

3.4 Segmentation and Dimension Reduction

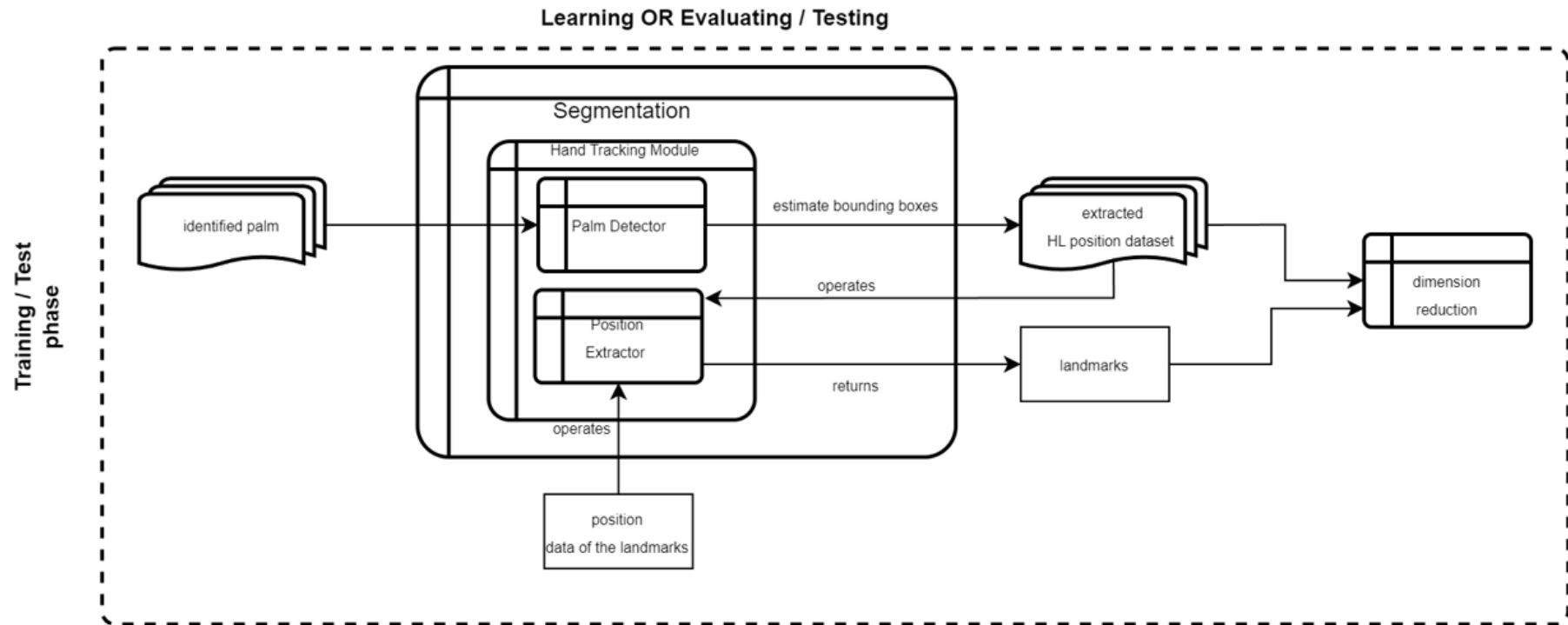


Figure 3.4.1: Workflow Diagram for Segmentation and Dimension Reduction

The **Segmentation stage** involves dividing the hand gestures into individual signs and separating the signs from the background and any other objects in the image.

In this proposed model workflow, the Segmentation stage involves the Hand Tracking module, which contains two main components: the **Palm Detector and the Position Extractor**, similar to the Feature Extraction stage.

The first component, the **Palm Detector**, estimates the bounding boxes around the identified palm from the feature extraction stage. This is important because it provides a reference for the position of the hand and the boundaries within which the individual signs can be recognized. The Palm Detector uses machine learning algorithms to estimate the bounding boxes, and then crops the image to focus only on the area around the palm.

The second component, the **Position Extractor**, operates on the extracted HL position dataset to return the landmarks from 3.3 Feature Extraction to be carried forward to the next stage, which is the Dimension Reduction process. This involves identifying the hand landmarks using computer vision techniques, such as template matching and feature detection, and separating the hand signs from the background.

Once the hand signs have been separated from the background, the **Dimension Reduction** process can be performed. This involves reducing the dimensionality of the feature vector for each sign, using techniques such as Principal Component Analysis (PCA). This reduces the computational complexity of the system and improves the accuracy of the classification model.

After that, the reduced feature vectors are used to **train a classification model**. The classification model is responsible for recognizing the individual hand signs and interpreting them as their corresponding American Sign Language gestures.

3.5 Classification

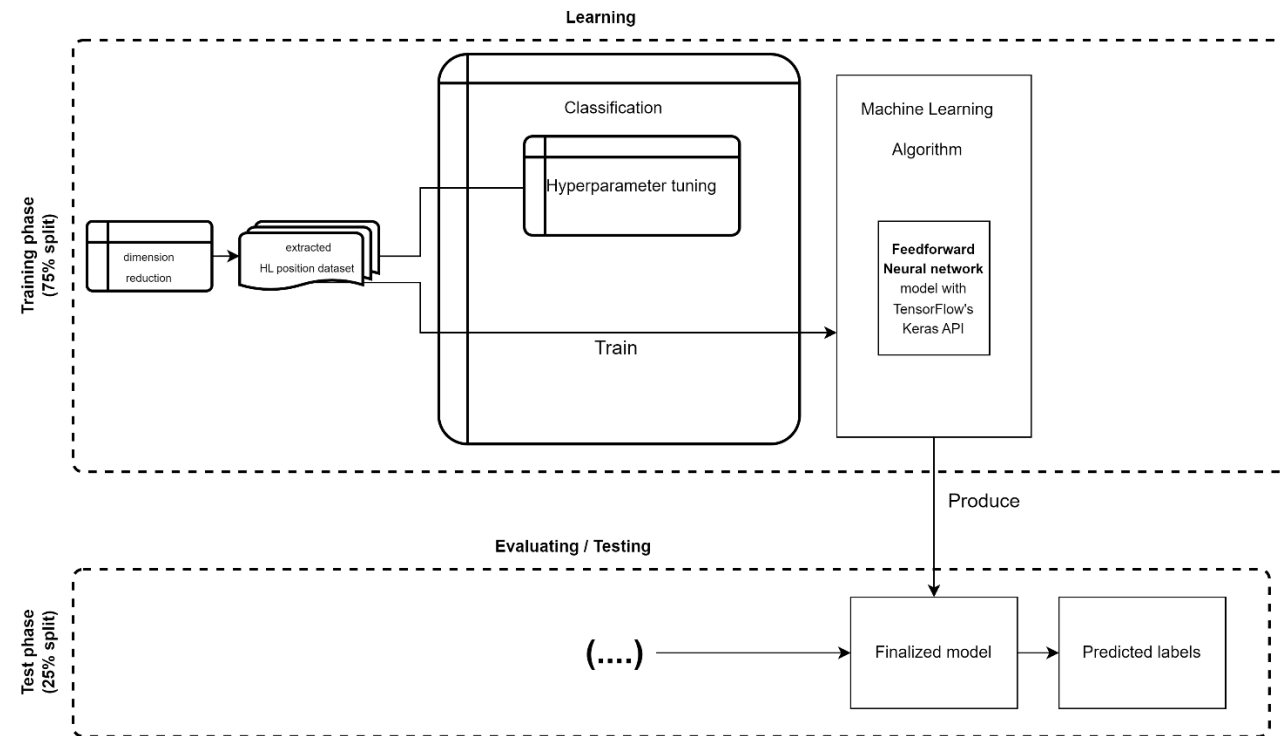


Figure 3.5.1: Workflow Diagram for Classification

The Classification stage is the final step in the proposed model workflow for developing a recognition system to interpret sign language. This stage involves building and training a machine learning model to recognize and classify the individual hand signs.

As mentioned previously, dimension reduction is determined and conducted if required. This is important because high-dimensional datasets can be computationally expensive and difficult to train machine learning models on.

Once the dimensionality of the dataset has been reduced, the next step is to build the machine learning model, which is the **Feedforward neural network (FNN)**. Below describes how FNN works in static and moving Sign Language classification.

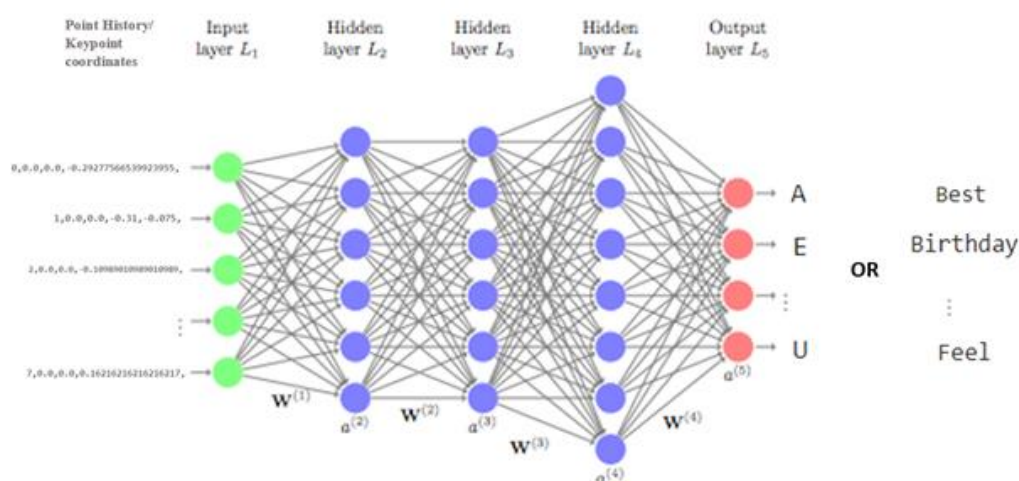


Figure 3.5.2: Feedforward neural network (FNN) in Sign Language classification

Based on above, the FNN follows a feedforward approach, where the Point History and Keypoint coordinates flows from the input layer (L_1) through the hidden layers (L_2 , L_3 , L_4) to the output layer (L_5).

Necessary libraries including CSV for handling data, NumPy for numerical operations, TensorFlow (tf) for machine learning, and train_test_split from scikit-learn for splitting the dataset are imported. A random seed of 42 is set for

reproducibility. This sequential model is defined using the Keras API from TensorFlow.

The **input layer (L1)** has several features determined by the expression $21 * 2$, assuming each data point has $21 * 2$ features. The **hidden layers** which are **L2, L3, L4**, consist of 2 dropout layers; one of them has a dropout rate of 0.2 is added to prevent overfitting, while the other has dropout rate of 0.4 for further regularization. They also have 2 dense layers with ReLU activation functions. The **output layer (L5)** contains softmax activation for classification.

This Feedforward neural network model is trained on the training dataset (X_{train}, y_{train}) for 1000 epochs with a batch size of 128. The validation data (X_{test}, y_{test}) is provided to evaluate the model's performance during training. Callbacks for model checkpoint and early stopping are utilized to save the model's weights during training and stop training early if validation loss does not improve, respectively. After the model is trained, it can predict the labels of the static/moving sign language gestures based on the learned patterns during training.

Overall, the Classification stage is a critical component of the proposed model workflow, as it involves building and training of the machine learning models to recognize and classify the individual hand signs. This stage is essential for producing a recognition system that can accurately interpret American Sign Language.

3.6 Model Evaluation

The model's performance is assessed using the test dataset, and the **validation loss and accuracy** are computed. To evaluate the **performance** of the real-time hand gesture recognition system for interpreting sign language, **a confusion matrix** is generated and visualized using seaborn and matplotlib. Since the system is designed to recognize gestures for interpreting sign language of 10 letters and 7 vocabularies, the confusion matrix would be of a 10x10, and 7x7 dimension, as shown in Table 3.6.1 and Table 3.6.2.

Confusion Matrix

Table 3.6.1: Confusion Matrix for 10 classes (letters)

		Predicted class										FN
		A	E	H	I	L	N	O	S	T	U	
Actual Class	A	TP ₁	I	II	III	IV	V	VI	VII	VIII	IX	I+II+...+I X
	E	X	TP ₂	XI	XII	XIII	XIV	XV	XVI	XVII	XVIII	X+XI+...+ XIV
	H	XIX	XX	TP ₃	XXI	XXII	XXIII	XXIV	XXV	XXVI	XXVII	XIX+ XX+...+ XXVII
	I	XXVIII	XXIX	XXX	TP ₄	XXXI	XXXII	XXXIII	XXXIV	XXX V	XXXVI	XXVIII+ XXIX+ + XXXVI

L	XXXVII	XXXVIII	XXXIX	XL	TP₅	XLI	XLII	XLIII	XLIV	XLV	XXXVII+ XXXVIII+ ... + XLV
N	XLVI	XLVII	XLVIII	XLIX	L	TP₆	LI	LII	LIII	LIV	XLVI+ XLVII+ ... + LIV
O	LV	LVI	LVII	LVIII	LIX	LX	TP₇	LXI	LXII	LXIII	LV+ LVI+... + LXIII
S	LXIV	LXV	LXVI	LXVII	LXVIII	LXIX	LXX	TP₈	LXXI	LXXII	LXIV+ LXV+ ...+ LXXII
T	LXXIII	LXXIV	LXXV	LXXVI	LXXVII	LXXVIII	LXXIX	LXXX	TP₉	LXXXI	LXXIII+ LXXIV+ ...+ LXXXI

	U	LXXXII	LXXXIII	LXXXIV	LXXXV	LXXXVI	LXXXV II	LXXX VIII	LXXXI X	XC	TP₁₀	LXXXII+ LXXXIII+ ...+ XC
	FP	X+XIX+ ...+ LXXXII	I+XX+ ... + LXXXIII	II+XI +... +LXXXIV	III+XII +... +LXXX V	IV+XIII +... +LXXXVI	V+XIV +... +LXXX VII	VI+XV +... +LXX XVIII	VII+X VI +... +LXX XIX	VIII+ XVII +... +LXX I+XC	IX +XVIII +...+L XXXI	

TP: True Positives

FP: False Positives

FN: False Negative

Table 3.6.2: Confusion Matrix for 7 classes (vocabularies)

		Predicted class							FN
		Best	Birthday	Please	Happy	Hearing	Like	Feel	
Actual Class	Best	TP₁	I	II	III	IV	V	VI	I+II+...+VI
	Birthday	VII	TP₂	VIII	IX	X	XI	XII	VII+VIII+...+XII
	Please	XIII	XIV	TP₃	XV	XVI	XVII	XVIII	XIII+XIV+...+XVIII
	Happy	XIX	XX	XXI	TP₄	XXII	XXIII	XXIV	XIX+XX+...+XXIV
	Hearing	XXV	XXVI	XXVII	XXVIII	TP₅	XXIX	XXX	XXV+XXVI+...+XXX
	Like	XXXI	XXXII	XXXIII	XXXIV	XXXV	TP₆	XXXVI	XXXI+XXXII+...+XXXVI
	Feel	XXXVII	XXXVIII	XXXIX	XL	XLI	XLII	TP₇	XXXVII+XXXVIII+...+XLII
	FP	VII+ XIII+ ...+ XXXVII	I+XIV+ ...+ XXXVIII	II+VIII +... +XXXIX	III+ IX +... +XL	IV+ X+... +XLI	V+XI +... +XLII	VI+XII +... +XXXVI	

TP: True Positives

FP: False Positives

FN: False Negatives

Classification Report

The **classification report** is also generated, which includes metrics as below in the test dataset.

Precision measures the proportion of true positive predictions among all positive predictions

$$Precision = \frac{TP}{TP+FP} \quad (3.61)$$

Recall measures the proportion of true positive predictions among all actual positives

$$Recall = \frac{TP}{TP+FN} \quad (3.62)$$

F1-Score is the harmonic mean of precision and recall.

$$F1 - Score = 2 \frac{Precision * Recall}{Precision + Recall} \quad (3.63)$$

Support is the number of occurrences of each class in the dataset.

The **accuracy** of the system is determined using equation 3.64, which calculates the ratio of the total number of correct predictions to the total number of predictions made by the system.

$$Accuracy = \frac{\sum TP}{Total} \quad (3.64)$$

3.7 Requirement Specifications

The requirement specifications provided in Table 3.7 outline the essential criteria for building the sign language interpretation using standard laptop webcams. Here's a breakdown of the requirements:

Table 3.7: Requirement Specifications to build system

Requirements	Tools / Suggestions
Hardware requirements	Standard laptop webcams
Input Video Requirements	Process video input in real-time (30 fps or higher)
Gesture Recognition Requirements	<p>The system should be able to detect hand gestures in different lighting conditions, backgrounds, and environments.</p> <p>The system should be able to recognize different hand gestures used in sign language, static or moving.</p>
Performance Requirements	<p>The system should be able to process video input and recognize gestures in real-time with minimal latency (less than 15 seconds). The system should be able to handle multiple users.</p>

3.8 Tools to use

Programming languages: Python

Libraries: OpenCV, Mediapipe, NumPy, TensorFlow

Pre-trained models: Hand tracking and sign language recognition models

3.9 Pseudocode for this project

Start

Define function `get_args()`:

- a. Define and parse command-line arguments

Define function `main()`:

- a. Parse command-line arguments using `get_args`
- b. Initialize camera capture
- c. Initialize MediaPipe Hand Model
- d. Read label information from CSV
- e. Initialize FPS calculation
- f. Create data structures for hand gesture history
- g. Initialize mode
- h. Specify Paths and Number of Classes (Keypoint Classification)
- i. Specify Paths and Parameters (Point history Classification)

- j. Main processing loop
 - while True:
 - k. Calculate FPS
 - l. Process key presses (e.g., change mode or exit)
 - m. Read a frame from the camera
 - n. Perform hand detection and tracking using MediaPipe

 - o. If `hands_detected(results)`:
 - p. Calculate bounding box and landmarks
 - q. Pre-process landmarks and point history
 - r. Log data to CSV (if required)
 - s. Hand sign classification
 - t. Finger gesture classification
 - u. Update finger gesture history
 - v. Drawing part
 - w. Display the processed frame

- x. Check for ESC key press to exit the loop
- y. Release camera and close OpenCV windows

z. Specify Paths and Number of Classes

- aa. Dataset Reading [For keypoint classifier, point history classifier]
- bb. Train-Test Split
- cc. Model Building
- dd. Model Training
- ee. Model Evaluation
- ff. Convert to TensorFlow Lite Model
- gg. Inference Test with TensorFlow Lite Model
- hh. Confusion Matrix and Classification Report
- ii. Save the Model for Inference

End

3.10 Work Breakdown Structure of the Project

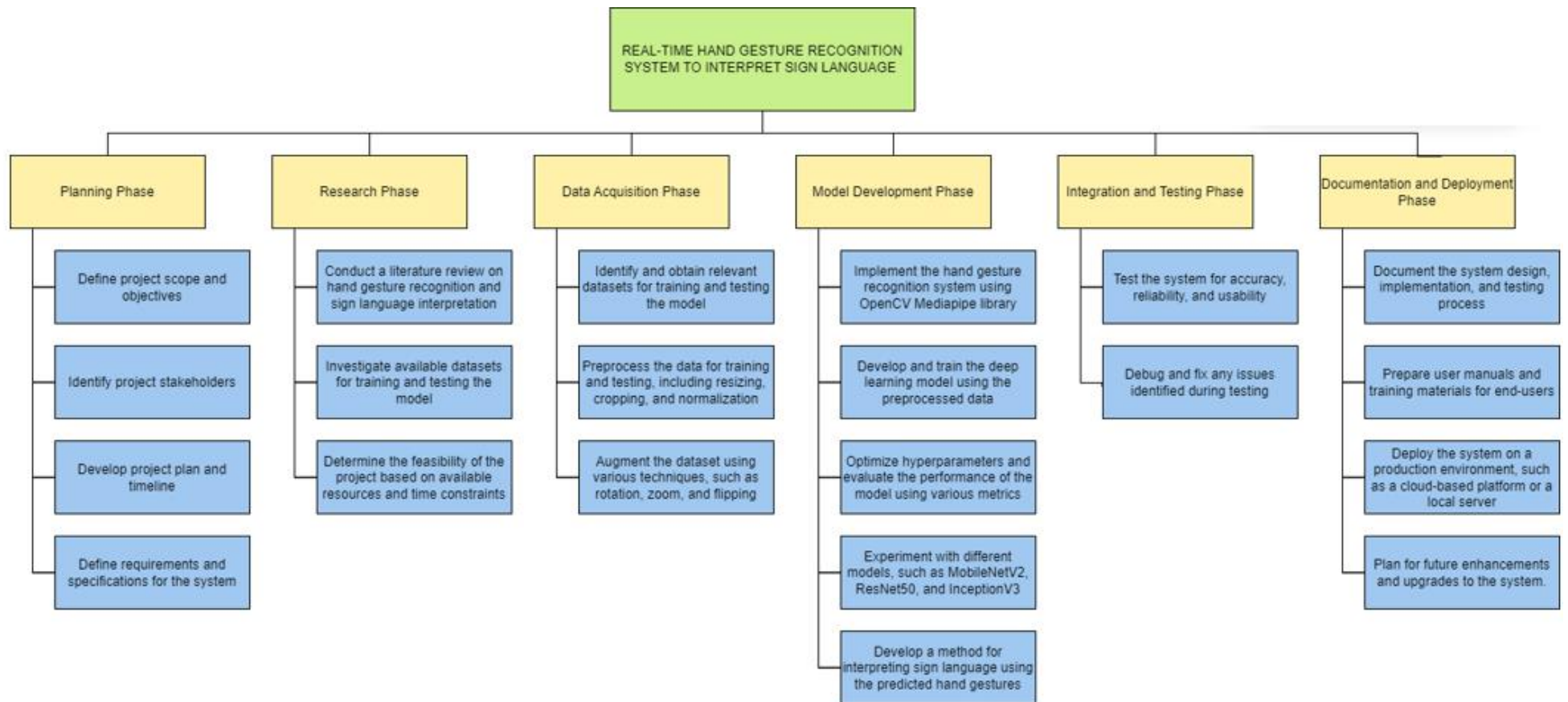


Figure 3.10: Work Breakdown Structure of the Project

3.11 Gantt Chart of Project

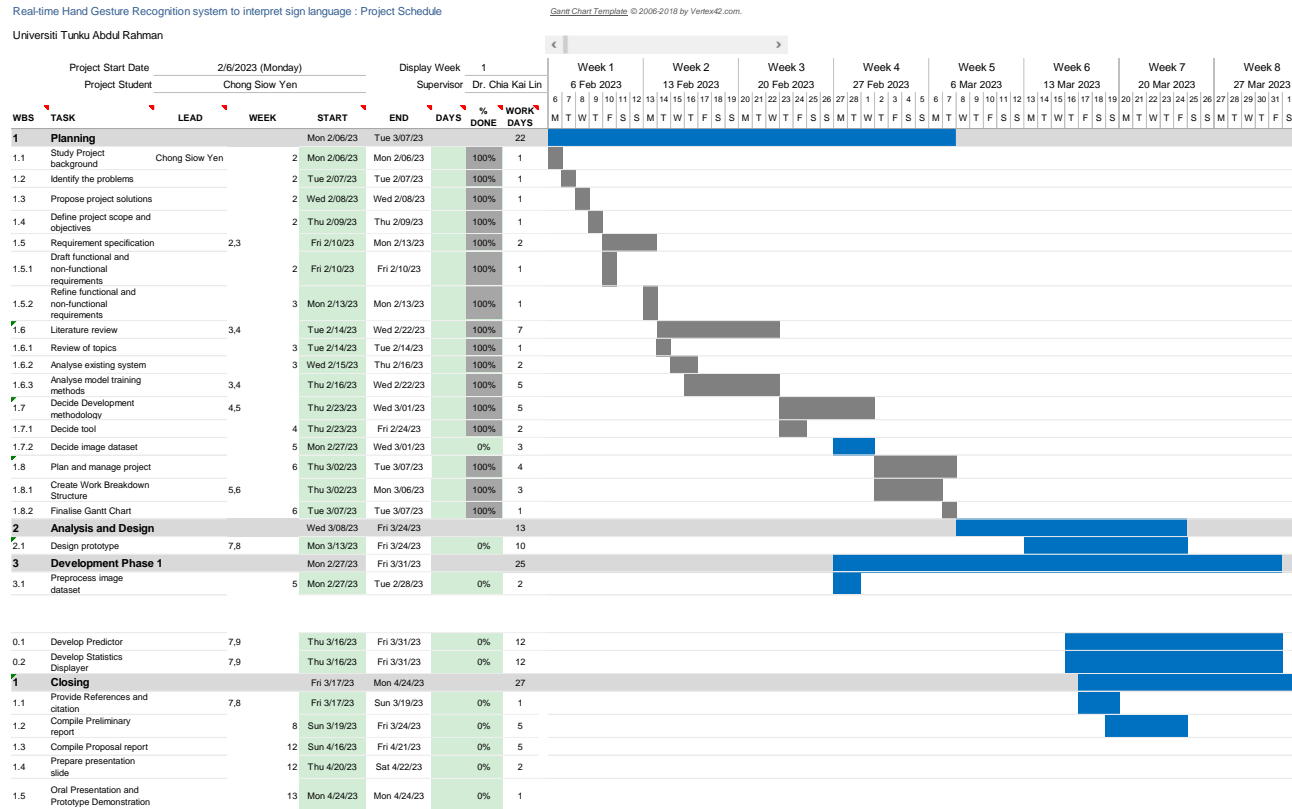


Figure 3.11a: Gantt Chart of Project I

Real-time Hand Gesture Recognition system to interpret sign language : Project Schedule

Gantt Chart Template © 2006-2018 by Vertex42.com.

Project II

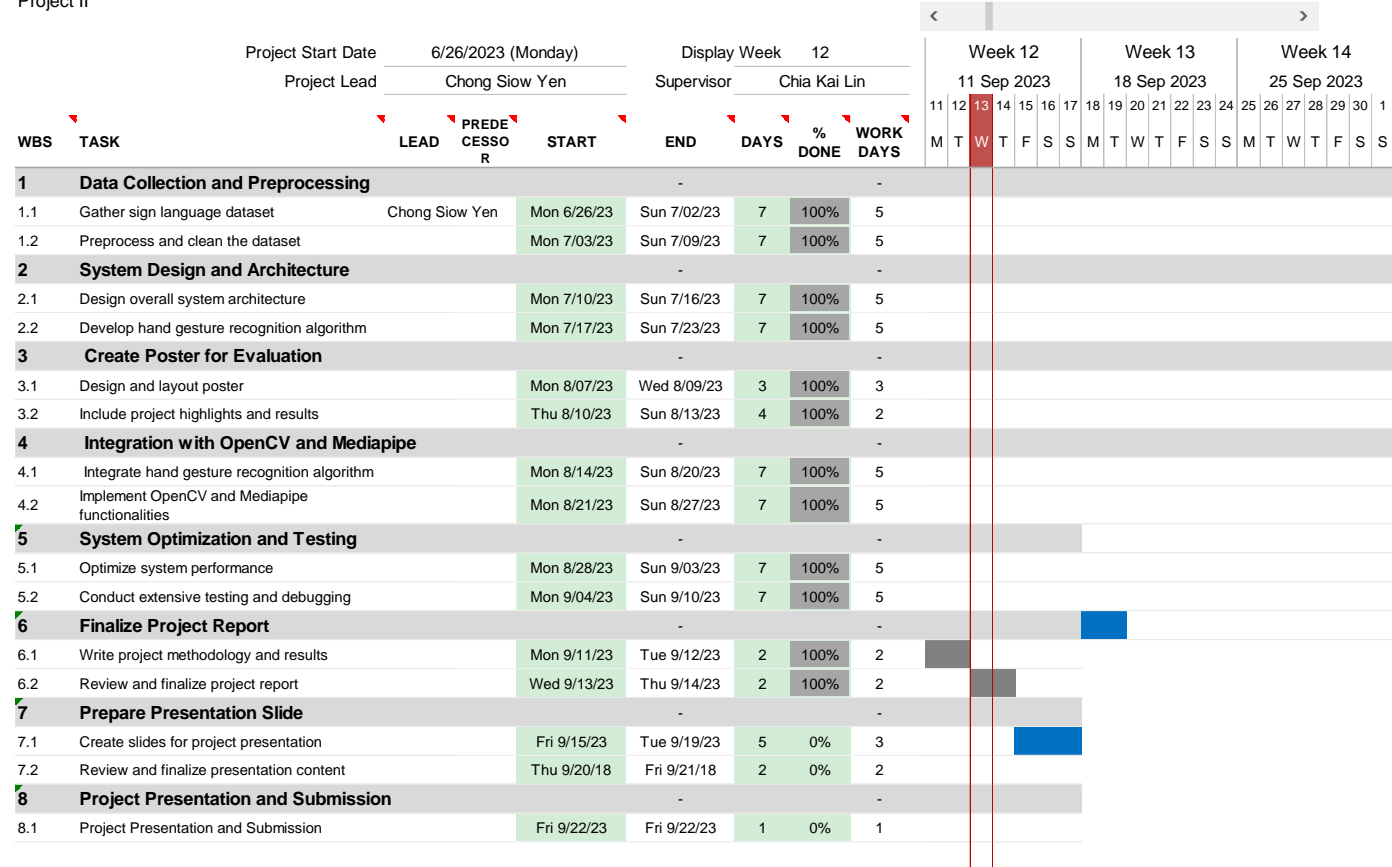


Figure 3.11b: Gantt Chart of Project II

The provided **WBS and Gantt Chart** pertain to a project aimed at developing a Mediapipe gesture recognition system for interpreting sign language. The project is divided into several phases, including planning, research, data acquisition, model development, interface design, integration and testing, documentation, and deployment. The planning phase focuses on defining the project scope and objectives, identifying stakeholders and team members, developing a project plan and timeline, defining requirements and specifications for the system.

The Gantt Chart of Project I and Project II in Figure 3.11a and 3.11b show the timeline for the gesture recognition system project. The planning phase spans several weeks and involves tasks such as defining the project scope and developing a project plan. Each phase is assigned a specific timeframe, and tasks are scheduled to ensure timely completion of the project.

CHAPTER 4

SYSTEM PERFORMANCE

This section dives into the performance evaluation of the neural network model, specifically designed for Keypoint (static hand gestures) and Point History (moving hand gestures) detection within the system.

4.1 Keypoint classifier detection

4.1.1 Training and Validation Metrics

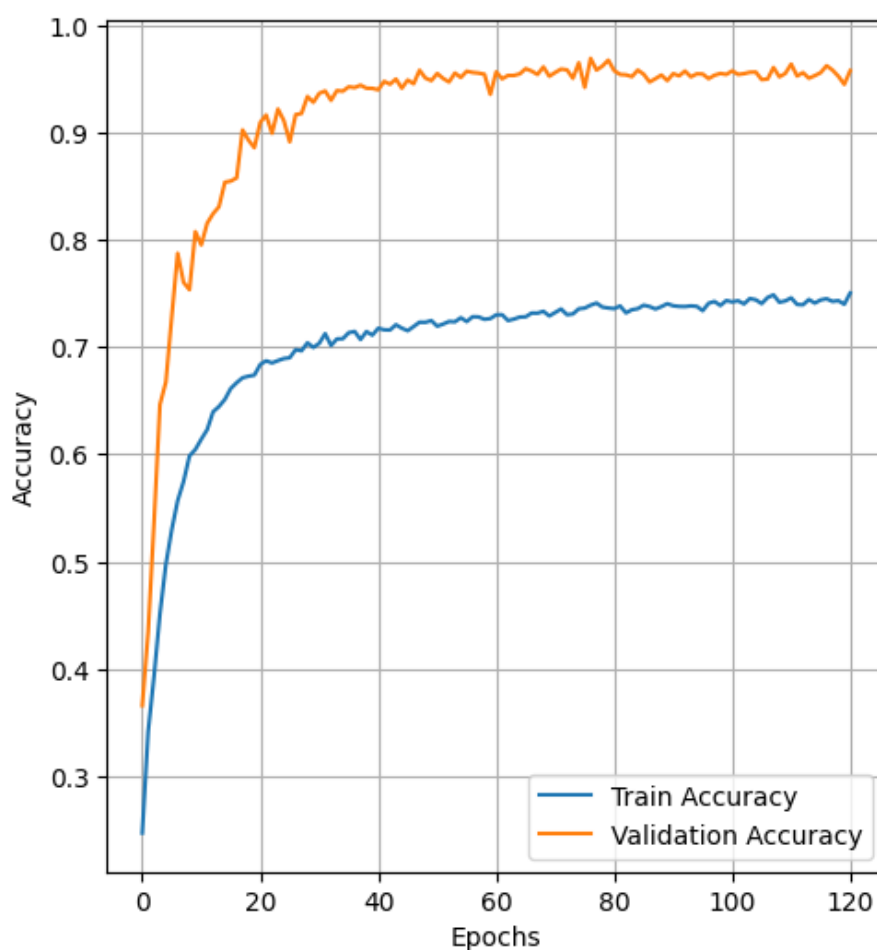


Figure 4.1.1a: Training history of Keypoint classifier (Accuracy over epochs)

The accuracy of the Keypoint classifier as shown in Figure 4.1.1a steadily increases over epochs during training, indicating that the model is learning and improving its ability to recognize keypoints in sign language gestures.

The training accuracy starts at a relatively low value (between 0-0.3) but gradually improves as the model learns from the training data. It reaches a high level of accuracy, achieving a value between 0.7-0.8, indicating that the model can correctly identify majority keypoints in the training dataset.

The validation accuracy follows a similar trend, initially starting at a lower value (between 0.3-0.4) and then improving over epochs. It also reaches a high accuracy level, close to 1.0, suggesting that the model generalizes well to unseen data.

Overall, the Keypoint classifier demonstrates excellent learning and generalization capabilities, achieving high accuracy on both the training and validation datasets.

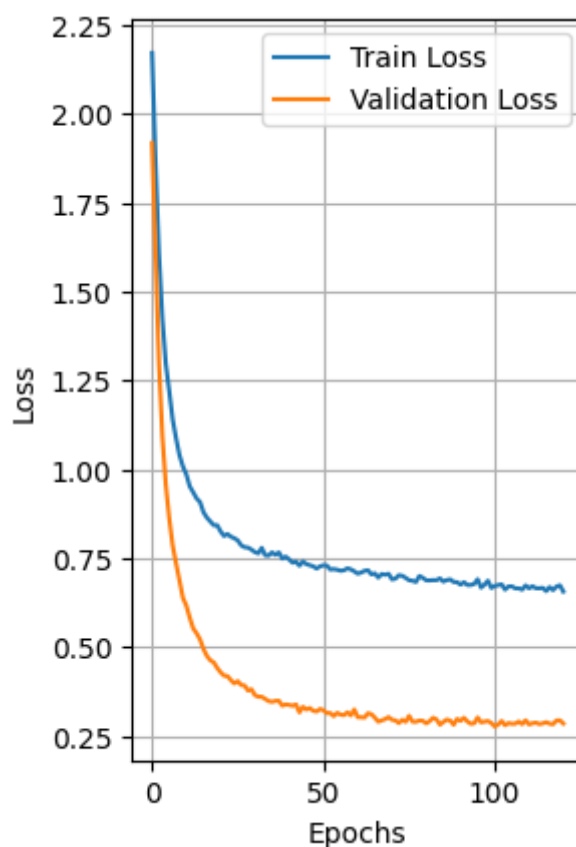


Figure 4.1.1b: Training history of Keypoint classifier (Loss over epochs)

The training loss, which represents the error during the training process as shown in Figure 4.1.1b, starts at a relatively high value of between 2.00-2.25 but consistently decreases over epochs. This indicates that the model is effectively reducing its training error and improving its ability to predict keypoints in sign language gestures.

The validation loss, which measures the error on unseen validation data, also follows a similar trend. It starts at a higher value of between 1.75-2.00 but steadily decreases over epochs. This demonstrates that the model is not overfitting the training data and is generalizing well to new, unseen data.

Both the training and validation losses converge to low values, indicating that the Keypoint classifier is learning effectively and producing accurate predictions for keypoints in sign language gestures.

	Metric	Value
0	Final Train Loss	0.656932
1	Final Train Accuracy	0.750541
2	Final Val Loss	0.285744
3	Final Val Accuracy	0.958055

Figure 4.1.1c: Key Training Metrics of Keypoint classifier

In terms of system performance, the Keypoint classifier demonstrates promising results based on Figure 4.1.1c. The final training loss, at 0.656932, indicates a good fit to the training data. Moreover, the final training accuracy of 0.750541 signifies successful pattern recognition within the training dataset. The final validation loss, which stands at 0.285744, implying the model's ability to generalize effectively to unseen data. The final validation accuracy, with value of 0.958055, underscores the model's proficiency in making accurate predictions on new and previously unseen data.

Based on above, the relatively low training and validation losses, along with reasonably high training and validation accuracies, suggest that the classifier has learned to recognize keypoints in sign language gestures effectively.

4.1.2 Confusion Matrix

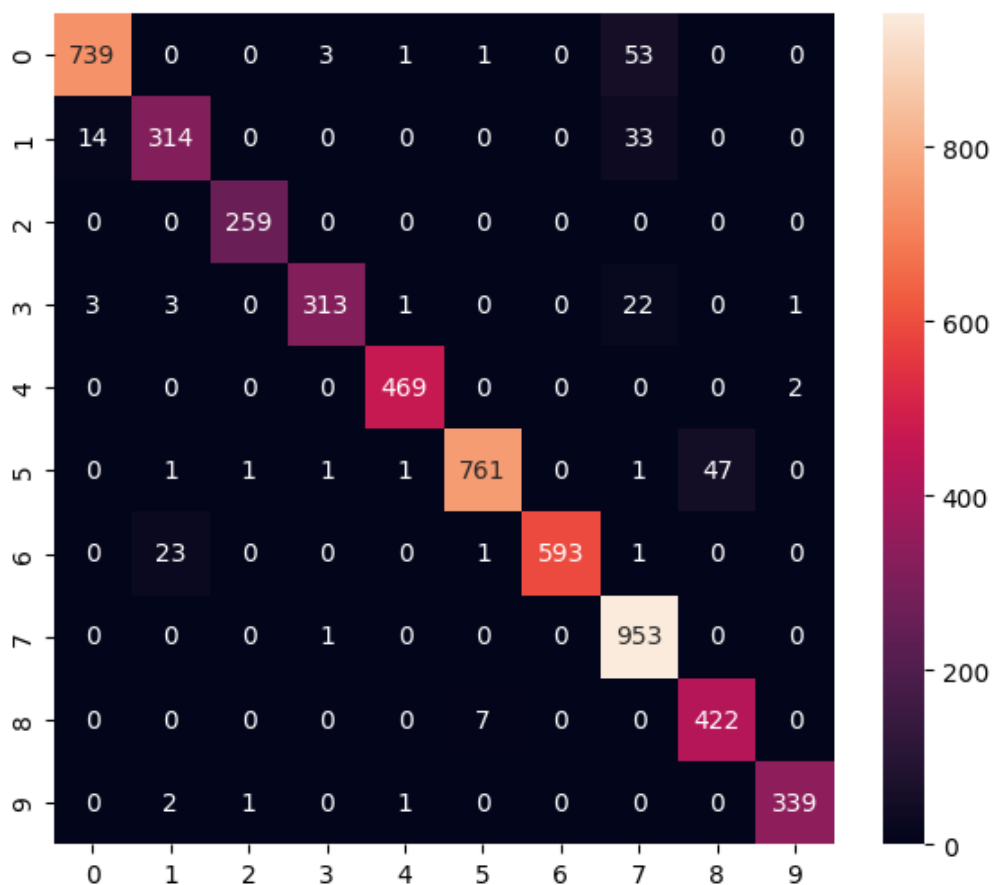


Figure 4.1.2: Confusion matrix of Keypoint classifier

Based on Figure 4.1.2, the high values (259-953) along the diagonal (True Positives for each class) indicate that the model correctly identifies most static hand gesture. Low off-diagonal values (0-53) (False Positives for each class) suggest that the model rarely misclassifies one static hand gesture as another.

The system exhibits excellent performance, achieving high accuracy and precision for all classes.

4.1.3 Classification Report

Classification Report				
	precision	recall	f1-score	support
0	0.98	0.93	0.95	797
1	0.92	0.87	0.89	361
2	0.99	1.00	1.00	259
3	0.98	0.91	0.95	343
4	0.99	1.00	0.99	471
5	0.99	0.94	0.96	813
6	1.00	0.96	0.98	618
7	0.90	1.00	0.94	954
8	0.90	0.98	0.94	429
9	0.99	0.99	0.99	343
accuracy			0.96	5388
macro avg	0.96	0.96	0.96	5388
weighted avg	0.96	0.96	0.96	5388

Figure 4.1.3: Classification Report of Keypoint classifier

The model as shown in Figure 4.1.3 performs well in correctly identifying instances of class "0" while maintaining a good balance between precision and recall. The precision of class "1" is reasonably high, the recall suggests that there may be room for improvement in correctly identifying instances of this class. Class "2" and Class "3" exhibits excellent performance with a precision, recall, and F1-score. Class "4" maintains high precision, recall, and F1-score values. Class "5" achieves an F1-score of 0.96, reflecting a good balance between precision and recall. Class "6" exhibits a perfect F1-score of 1.00, indicating that the model accurately identifies instances of class "6". Class "7" and "8" both have an F1-score of 0.94, suggesting solid performance. Class "9" achieves high precision and recall.

Keypoint classifier demonstrates strong performance overall, with some classes achieving near-perfect accuracy and F1-scores.

4.2 Point History Classifier Detection

4.2.1 Training and Validation Metrics

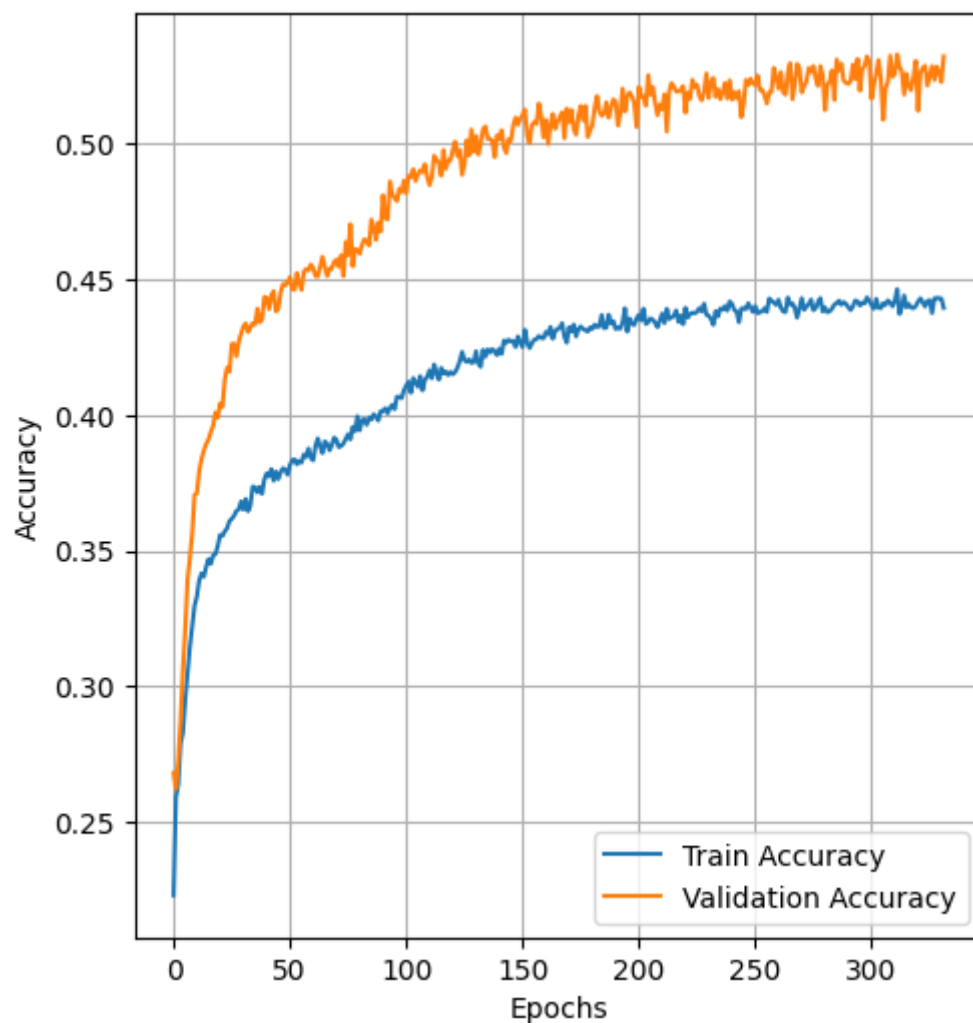


Figure 4.2.1a: Training history of Point History Classifier (Accuracy over epochs)

The training accuracy, represented by the blue curve in Figure 4.2.1a, starts at an initial value of between 0-0.25 and gradually increases over epochs. It reaches an accuracy of between 0.40-0.45 by the end of training, indicating that the model improves its ability to fit the training data over time.

The validation accuracy, in orange, also shows an upward trend throughout the training process. It starts between 0.25-0.30 and steadily increases, eventually reaching an accuracy between 0.50-0.60 by the end of training. This suggests

that the model generalizes well to unseen data, as evidenced by the improvement in validation accuracy.

Overall, the training history demonstrates that the Point History Classifier makes progress in learning from the training data and generalizing its knowledge to validation data. While the final accuracy values may not be extremely high, the upward trends in both training and validation accuracy indicate that the model is learning effectively.

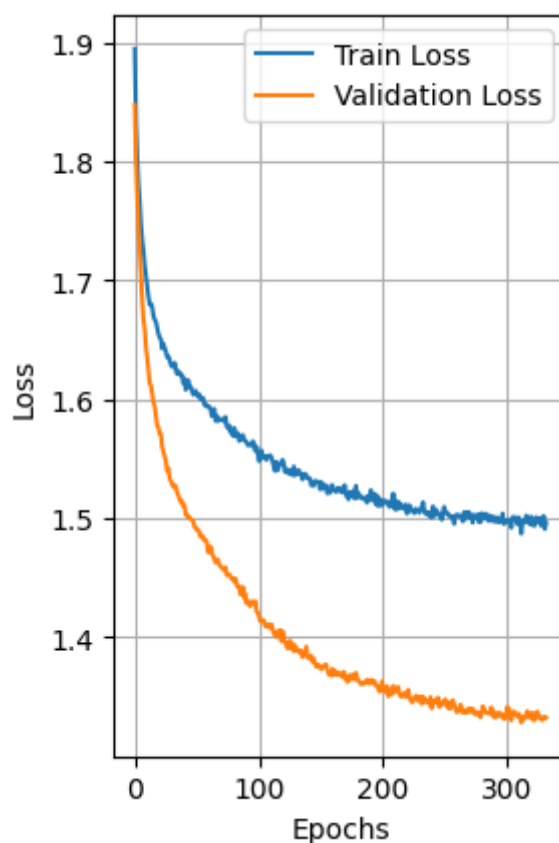


Figure 4.2.1b: Training history of Point History Classifier (Loss over epochs)

The training loss, in blue in Figure 4.2.1b, starts at an initial value of approximately 1.9 and gradually decreases as training progresses. It reaches a minimum training loss of 1.5 by the end of training, indicating that the model

is effectively minimizing the error between its predictions and the actual training data.

The validation loss, represented by the orange curve, also shows a decreasing trend throughout the training process. It starts around 1.8-1.9 and steadily decreases, eventually reaching a validation loss of near 1.3 by the end of training. This suggests that the model is not only fitting the training data well but also generalizing effectively to unseen validation data.

Overall, the training history demonstrates that the Point History Classifier successfully reduces its loss function, indicating improved model performance. These results indicate promising system performance, with the potential for further enhancements through fine-tuning or additional training epochs.

	Metric	Value
0	Final Train Loss	1.496126
1	Final Train Accuracy	0.439580
2	Final Val Loss	1.332529
3	Final Val Accuracy	0.532254

Figure 4.2.1c: Key Training Metrics of Point History Classifier

The Point History Classifier exhibits notable performance metrics, as shown in Figure 4.2.1c. The final training loss at 1.496126, indicates an optimal learning process with a close fit to the training data. The training accuracy of 0.439580, reflecting average pattern recognition within the training dataset. For generalization, the model's final validation loss is 1.332529, signalling its ability to extrapolate learning to new, unseen historical data. This followed by a validation accuracy of 0.532254, highlighting the model's proficiency in making accurate predictions on previously unobserved historical data points.

The model shows promise in capturing historical patterns, as indicated by the validation metrics. The training accuracy can be improved further. Further refinements and optimizations could enhance its performance in recognizing historical trends.

4.2.2 Confusion Matrix

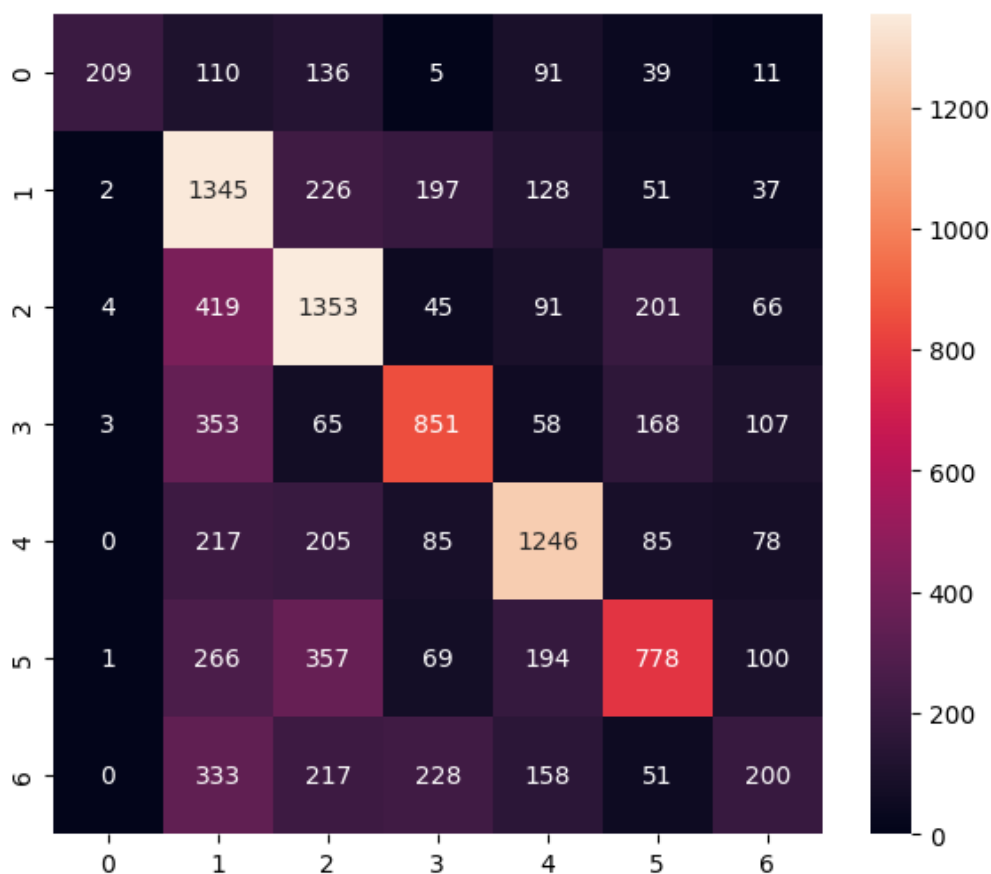


Figure 4.2.2: Confusion matrix of Point History classifier

High values are found along the diagonal (true positives for each class) in Figure 4.2.2 for class 1 - *Birthday*, 2 - *Please*, 4 - *Hearing* indicate that the model correctly identifies for these respective moving hand gestures. Lighter colours with value (1-419) off the diagonal suggest areas where the classifier may struggle to make accurate predictions. This indicates the model needs slight improvement correctly identifies all classes.

4.2.3 Classification Report

Classification Report				
	precision	recall	f1-score	support
0	0.95	0.35	0.51	601
1	0.44	0.68	0.53	1986
2	0.53	0.62	0.57	2179
3	0.57	0.53	0.55	1605
4	0.63	0.65	0.64	1916
5	0.57	0.44	0.50	1765
6	0.33	0.17	0.22	1187
accuracy			0.53	11239
macro avg	0.58	0.49	0.50	11239
weighted avg	0.55	0.53	0.52	11239

Figure 4.2.3: Classification Report of Point History classifier

For class 0 in Figure 4.2.3, while the model has a high precision for this class, it struggles with recall, meaning it correctly identifies instances of class 0 but misses many. The model has a lower precision but better recall for this class, indicating it can correctly classify more instances of class 1. The model performs reasonably well in terms of precision and recall for class 2. Class 3 has a balanced performance. Class 4 has slight strong performance in terms of precision and recall. Class 5 has a moderate performance. Class 6 exhibits the lowest performance metrics, suggesting challenges in correctly classifying this category.

Overall, the model's performance is moderate, with an accuracy of 0.53, indicating that it correctly classified approximately 53% of the data points.

4.3 Comparing OpenCV with Mediapipe and Neural Network with other methodologies in Literature Review (refer Table 2.8)

Table 4.3. Accuracy Comparison Literature Review and Current Development

Approach Process	OpenCV only	OpenCV with Mediapipe	LeapMotion Controller with CNN	Gloves with sensors	Skeleton Aware Multimodal SLR framework	OpenCV with Mediapipe and TensorFlow's Neural Network
Accuracy	Recognizes all 10 hand gestures with 10 hits out of 10 trials	DTC: Training data accuracy 100%, Test data accuracy 96.4% RFC: Training data accuracy 100%, Test data accuracy 97.2% SVM: 97.9% accuracy with the	1. 95% 2. 90.3% 3. 96%	1. 78.33% to 95% 2. 65.7% to 98.2%	1. Multi-stream SL-GCN model (top-1 accuracy) 95.45% on the validation set, baseline RGB and RGB-D (top-1 accuracy) models were 49.23% and 62.03%	Keypoint Classifier: 96% Point-History Classifier: 53%

		test data for a radial base kernel KNN: Training accuracy 97.4%, Test accuracy 98.1%			2. VTN-PF: 91.51% on the validation set	
Authors	1. Ahmad Puad Ismail et al. 2. Riaz Sulaimi	C. M. Suryateja et al.	1. Naglot and Kulkarni (2016) 2. Chong and Lee (2018) 3. Bird, Ekárt, and Faria (2020)	1. Shukor et al. 2. Lee and Lee	1. Jiang et al. 2. De Coster et al.	Chong Siow Yen

Based on Table 4.3, it is witnessed that the developed system, which combines **OpenCV with Mediapipe and TensorFlow's Neural Network model**, has achieved superior accuracy, especially in Keypoint Classification, where it reached an impressive 96%. This performance surpasses that of both Gloves with sensors and the LeapMotion Controller with CNN. Furthermore, the system's overall accuracy, including Point History Classification at 53%, outperforms the accuracy achieved by the Skeleton Aware Multimodal SLR framework.

TESTING RESULTS

4.1 Introduction

As outlined in Chapter 1, Section 1.5, the primary objective of this project is to develop an efficient algorithm for the system for recognizing hand gestures in real-time. This section delves into the testing results, providing an analysis of the system's performance and its ability to interpret and recognize sign language gestures.

To meet this objective, a comprehensive approach has been employed that includes:

4.1.1 Approach

a. Testing Audience

The target audience for this project consists of 4 testers from diverse demographics, including age, ethnicity, gender, hand dominance (right-handed/left-handed/ambidextrous), ASL proficiency (beginner/intermediate/advance), and experience with ASL technology (novice/experienced).

This diverse group of testers ensures a thorough evaluation of the system's performance across various user profiles. Table 4.2.4.1a below shows the testers' profile and feedback on system.

Table 4.1.1a: Testers' Profile and Feedback on System

Tester	Age	Ethnicity	Gender (Male/F emale)	Hand Dominance (Right- handed/Left- handed/ Ambidextrou s)	ASL Proficiency (Beginner/ Intermediate/ Advance)	Experience with ASL Technology (Novice/Exp erience)
Tester 1	25	Chinese	Male	Right-handed	Beginner	Novice
Tester 2	52	Chinese	Female	Right-handed	Beginner	Novice
Tester 3	22	Chinese	Female	Right-handed	Intermediate	Novice
Tester 4	22	Indian	Female	Left-handed	Beginner	Novice

Visualizations of Demographics of Testers as shown below would include Figure 4.1.1a: Age of Testers, Figure 4.1.1b: Ethnicity of Testers, Figure 4.1.1c: Gender of Testers, Figure 4.1.1d: Hand Dominance of Testers, Figure 4.1.1e: ASL Proficiency and Figure 4.1.1f: Experience of ASL Tech.

Demographics of Testers visualized as below:

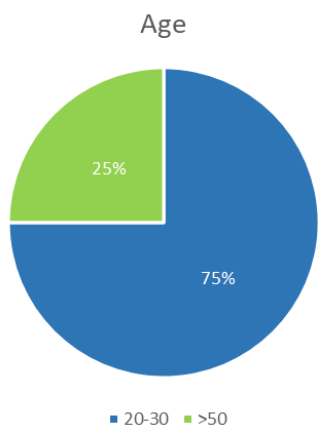


Figure 4.1.1a: Age of Testers

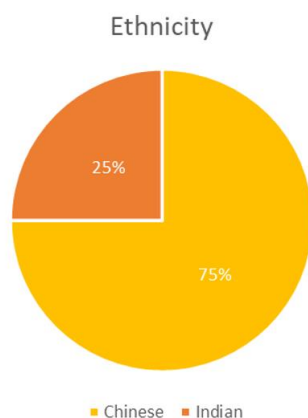


Figure 4.1.1b: Ethnicity of Testers

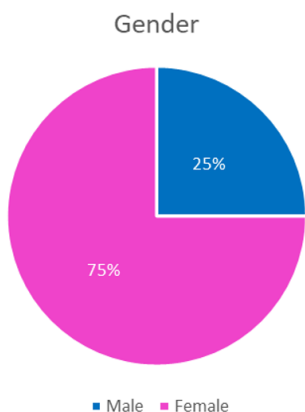


Figure 4.1.1c: Gender of Testers

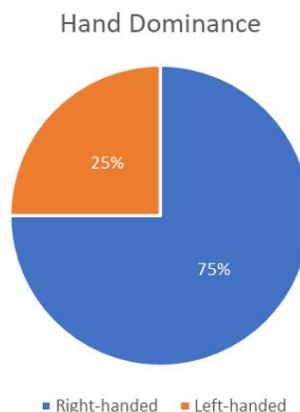


Figure 4.1.1d: Hand Dominance of Testers

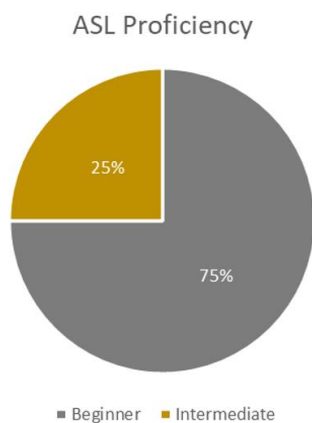


Figure 4.1.1e: ASL Proficiency

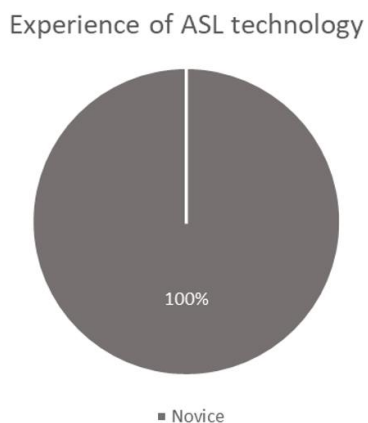


















Figure 4.1.1f: Experience of ASL Tech

b. Gesture Variability

Both static and moving gestures have been considered for testing, replicating real-world sign language communication scenarios. Static gestures represent stationary hand signs, while moving gestures involve dynamic hand movements commonly used in sign language. The gestures are trained with left and right hand as shown in Table 4.2.4.1b and Table 4.2.4.1c, to ensure versatility and accuracy in recognizing sign language expressions, regardless of the signer's dominant hand.

Table 4.1.1b: Static Gestures with contrasting hand signing

0. A	
Right	Left
	
1. E	
Right	Left
	
2. H	
Right	Left
	
3. I	
Right	Left

	
4. L	
Right	Left
	
5. N	
Right	Left
	
6. O	
Right	Left
	
7. S	
Right	Left
	
8. T	





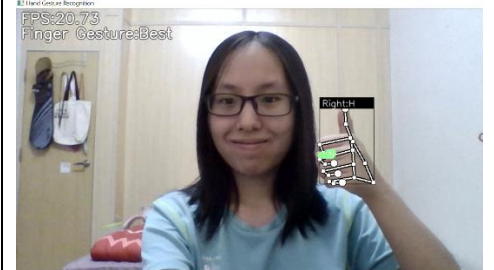



Right	Left
	
9. U	
Right	Left
	

Table 4.1.1b1: Moving Gestures with contrasting hand signing

0. Best	
Right	Left
	
1. Birthday	
Right	Left
	
2. Please	
Right	Left

<p>Hand Gesture Recognition FPS:16.83 Finger_Gesture:Please</p>		<p>Hand Gesture Recognition FPS:12.55 Finger_Gesture:Please</p>	
3. Happy			
Right		Left	
<p>Hand Gesture Recognition FPS:21.36 Finger_Gesture:Happy</p>		<p>Hand Gesture Recognition FPS:10.23 Finger_Gesture:Happy</p>	
4. Hearing			
Right		Left	
<p>Hand Gesture Recognition FPS:20.78 Finger_Gesture:Hearing</p>		<p>Hand Gesture Recognition FPS:14.17 Finger_Gesture:Hearing</p>	
5. Like			
<p>Hand Gesture Recognition FPS:21.33 Finger_Gesture:Like</p>		<p>Hand Gesture Recognition FPS:12.7 Finger_Gesture:Like</p>	
6. Feel (unidentified)			
Right		Left	



c. Evaluation Metrics

The evaluation metrics include **average duration, speed, and recognition counts for each tested gesture**. These metrics allow us to assess the system's accuracy, efficiency, and overall performance in interpreting sign language gestures.

d. Feedback and Comments

Tester feedback and comments play a crucial role in refining the system. Feedback, such as difficulties encountered or suggestions for improvement, helps in identifying areas where the system may require enhancements.

4.1.2 Expected Achievement

By following this approach, the project is set to achieve the following

a. Evaluate Accuracy and Efficiency

Assess the accuracy and efficiency of the hand gesture recognition system in real-time communication scenarios. This evaluation is essential to ensure that the system effectively meets the communication needs of individuals with hearing impairments, thereby reducing potential communication gaps.

b. Address Challenges

Identify potential challenges in the adoption of the system and develop solutions to address them. This proactive approach is vital to ensure the successful adoption of the system and enhance its overall accuracy and efficiency.

4.2 Test Setup and Data Collection

In this section, a detailed account of the test setup and data collection process is provided for evaluating the developed sign language recognition system. A systematic approach was followed to ensure accurate and comprehensive assessment.

4.2.1 Test Environment Setup

a. For testers

The testing environment is conducted at regular lighting conditions, at comfort-level of distance and height of laptop setup of testers, to mimic real-world conditions.

Here are images of testers (Figure 4.2.1a - 4.2.1d) trying out the system at the comfort of their homes or dining places.

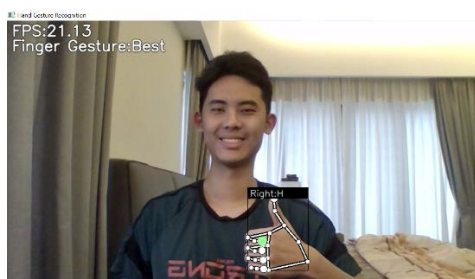


Figure 4.2.1a: Tester 1 doing 'Best' gesture being detected

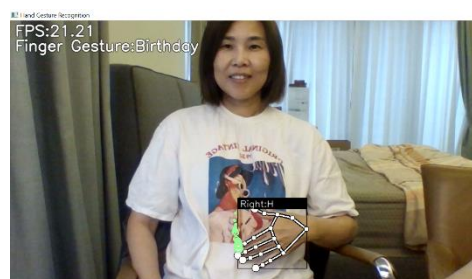


Figure 4.2.1b: Tester 2 doing 'Birthday' gesture being detected

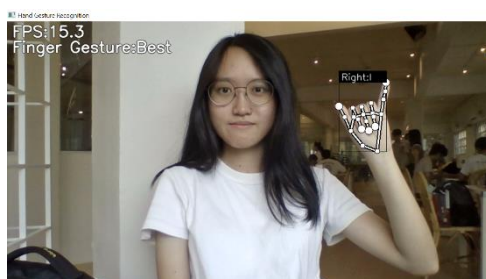


Figure 4.2.1c: Tester 3 doing 'I' gesture being detected

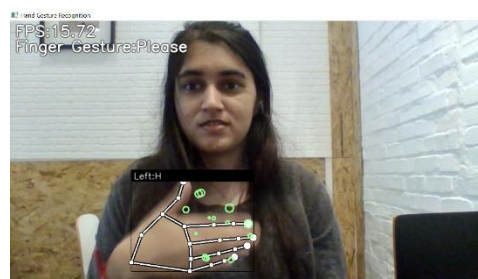


Figure 4.2.1d: Tester 4 doing 'Please' gesture being detected

b. For self

The testing environment was carefully configured. Two **lighting conditions** were considered: one at regular lighting, and another at dim lighting. Other than that, the **sitting positions** are set at 36cm from their respective table, and another at 53cm from the table. These conditions were chosen to assess the system's performance in different visual scenarios.

Example snapshots of system (4.2.1aa- 4.2.1dd) detecting user's 'A' gesture under various conditions.



Figure 4.2.1.1aa: 36cm – from table

Light condition



Figure 4.2.1.1bb: 53cm – from table

Light condition

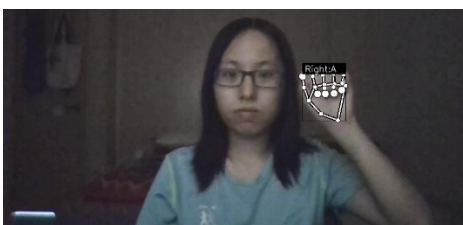


Figure 4.2.1.1cc: 36cm – from table

Dim condition

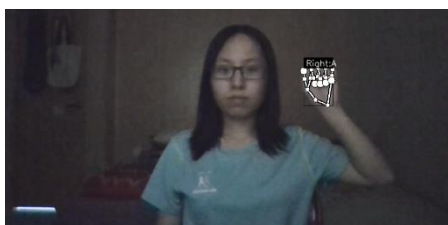


Figure 4.2.1.1dd: 53cm – from table

Dim condition

4.2.2 Gestures Used for Testing

Evaluation involved a diverse set of gestures, categorized into two main types:

Static Gestures:

These represent stationary hand signs used in sign language. The hand signs which are involved, are 10 letters (A, E, H, I, L, N, O, S, T, U).

Moving Gestures:

These involve dynamic hand movements typically used for expressive communication in sign language. The hand movements involved in this testing contain 7 vocabularies, namely (i) Best, (ii) Birthday, (iii) Please, (iv) Happy, (v) Hearing, (vi) Like and (vii) Feel.

4.2.3 Tester Recruitment and Data Collection

A diverse group of 4 testers was recruited to ensure a comprehensive evaluation of the sign language recognition system. Tester demographics, including age, ethnicity, sex, used hand (right-handed/left-handed/ambidextrous), ASL proficiency (beginner/intermediate/advance), and experience with ASL technology (novice/experienced), were recorded to account for variations in user profiles.

Data collection

Data collection involved the execution of various gestures by testers, and the following data points were collected for each gesture:

Recognition Result

Whether the system correctly recognized the gesture.

Recognized Counts

The number of times the gesture was correctly recognized.

Time Taken (s : cs)

The time taken by the system to recognize the gesture, recorded in seconds and centiseconds.

Average Time Taken

The average time taken for recognition across all testers.

Average Speed

The average speed of gesture execution, measured in gestures per second (s⁻¹).

4.3 Evaluation and Tester Feedback

a. For testers

Gesture Recognition results

The recognition results for each gesture, both static and moving, were documented for each tester. This includes whether the gesture was correctly recognized or not. Recognized counts provide insights into the system's consistency and reliability in recognizing specific gestures.

Formulas to calculate Average time taken and Average speed as follows:

$$\text{Average time taken} = \frac{\text{Time taken (s)}}{\text{Number of testers}} \quad (4.2.6.1)$$

$$\text{Average speed} = \frac{1}{\text{Average time taken (s)}} \quad (4.2.6.2)$$

The data collected for a subset of gestures and testers is presented in Table 4.2.6 below:

Table 4.3 Tabulation of Testers Attempt of Sign Language Recognition System

Static/Moving	Gesture	Tester	Recognition Result	Recognised counts	Time taken (s : cs)	Average time taken (s)	Average speed (s ⁻¹)
Static	A	1	Recognised	4	02:42	$\frac{2.42 + 2.67 + 1.52 + 1.49}{4} = 2.03s$	$\frac{1}{2.03} = 0.49s^{-1}$
		2	Recognised		02:67		
		3	Recognised		01:52		
		4	Recognised		01:49		
Static	E	1	Recognised	4	01:58	$\frac{1.58 + 1.59 + 0.98 + 1.48}{4} = 1.41s$	$\frac{1}{1.41} = 0.71s^{-1}$
		2	Recognised		01:59		
		3	Recognised		00:98		
		4	Recognised		01:48		
Static	H	1	Recognised	4	01:84	$\frac{1.84 + 1.84 + 0.73 + 1.34}{4} = 1.44s$	$\frac{1}{1.44} = 0.69s^{-1}$
		2	Recognised		01:84		
		3	Recognised		00:73		
		4	Recognised		01:34		
Static	I	1	Recognised	4	01:11		$\frac{1}{1.84}$
		2	Recognised		04:05		

		3	Recognised		00:96	$1.11 + 4.05 + 0.96$	$= 0.54s^{-1}$
		4	Recognised		01:25	$\frac{+1.25}{4}$	
						$= 1.84s$	
Static	L	1	Recognised	4	01:48	$1.48 + 1.67 + 1.06$	$\frac{1}{1.25}$
		2	Recognised		01:67	$+0.77$	
		3	Recognised		01:06	$\frac{4}{4}$	$= 1.25s$
		4	Recognised		00:77		$= 0.8s^{-1}$
Static	N	1	Recognised	4	00:97	$0.97 + 2.77 + 1.24$	$\frac{1}{1.76}$
		2	Recognised		02:77	$+2.04$	
		3	Recognised		01:24	$\frac{4}{4}$	$= 1.76s$
		4	Recognised		02:04		$= 0.57s^{-1}$
Static	O	1	Recognised	4	00:73	$0.73 + 5.10 + 1.12$	$\frac{1}{2.05}$
		2	Recognised		05:10	$+1.25$	
		3	Recognised		01:12	$\frac{4}{4}$	$= 2.05s$
		4	Recognised		01:25		$= 0.49s^{-1}$
Static	S	1	Recognised	4	01:34		$\frac{1}{1.27}$
		2	Recognised		01:32		

		3	Recognised		01:01	$1.34 + 1.32 + 1.01$	$= 0.79s^{-1}$
		4	Recognised		01:40	$\frac{+1.40}{4}$ $= 1.27s$	
Static	T	1	Recognised	4	01:26	$1.26 + 2.57 + 1.25$	$\frac{1}{1.68}$ $= 0.60s^{-1}$
		2	Recognised		02:57	$\frac{+1.64}{4}$	
		3	Recognised		01:25	$= 1.68s$	
		4	Recognised		01:64		
Static	U	1	Recognised	4	00:84	$0.84 + 1.13 + 1.67$	$\frac{1}{1.15}$ $= 0.87s^{-1}$
		2	Recognised		01:13	$\frac{+0.94}{4}$	
		3	Recognised		01:67	$= 1.15s$	
		4	Recognised		00:94		
Moving	Best	1	Recognised	4	01:02	$1.02 + 1.66 + 1.95$	$\frac{1}{1.61}$ $= 0.62 s^{-1}$
		2	Recognised		01:66	$\frac{+1.81}{4}$	
		3	Recognised		01:95	$= 1.61s$	
		4	Recognised		01:81		
Moving	Birthday	1	Recognised	4	02:59		$\frac{1}{1.95}$
		2	Recognised		02:29		

		3	Recognised		00:96	$2.59 + 2.29 + 0.96$	$= 0.51s^{-1}$
		4	Recognised		01:96	$\frac{+1.96}{4}$	
						$= 1.95s$	
Moving	Please	1	Recognised	3	02:92	$2.92 + 15 + 12.88$	$\frac{1}{8.45}$ $= 0.12s^{-1}$
		2	Unrecognised		15:00	$+2.99$	
		3	Recognised		12:88	$\frac{4}{4}$	
		4	Recognised		02:99	$= 8.45s$	
Moving	Happy	1	Recognised	4	02:43	$2.43 + 12.59 + 6.59$	$\frac{1}{6.28}$ $= 0.16s^{-1}$
		2	Recognised		12:59	$+3.51$	
		3	Recognised		06:59	$\frac{4}{4}$	
		4	Recognised		03:51	$= 6.28s$	
Moving	Hearing	1	Recognised	4	01:65	$1.65 + 1.74 + 11.14$	$\frac{1}{4.04}$ $= 0.25s^{-1}$
		2	Recognised		01:74	$+1.64$	
		3	Recognised		11:14	$\frac{4}{4}$	
		4	Recognised		01:64	$= 4.04s$	

Moving	Like	1	Recognised	4	01:29	$\frac{1.29 + 6.80 + 5.21 + 2.08}{4}$ $= 3.85\text{s}$	$\frac{1}{3.85}$ $= 0.26\text{s}^{-1}$
		2	Recognised		06:80		
		3	Recognised		05:21		
		4	Recognised		02:08		
Moving	Feel	1	Unrecognised	0	15:00	$\frac{15 + 15 + 15 + 15}{4}$ $= 15\text{s}$	$\frac{1}{15}$ $= 0.07\text{s}^{-1}$
		2	Unrecognised		15:00		
		3	Unrecognised		15:00		
		4	Unrecognised		15:00		

Based on the collection of data and evaluation metrics from above, detailed evaluation results, insights gained from testers' feedback, are analysed.

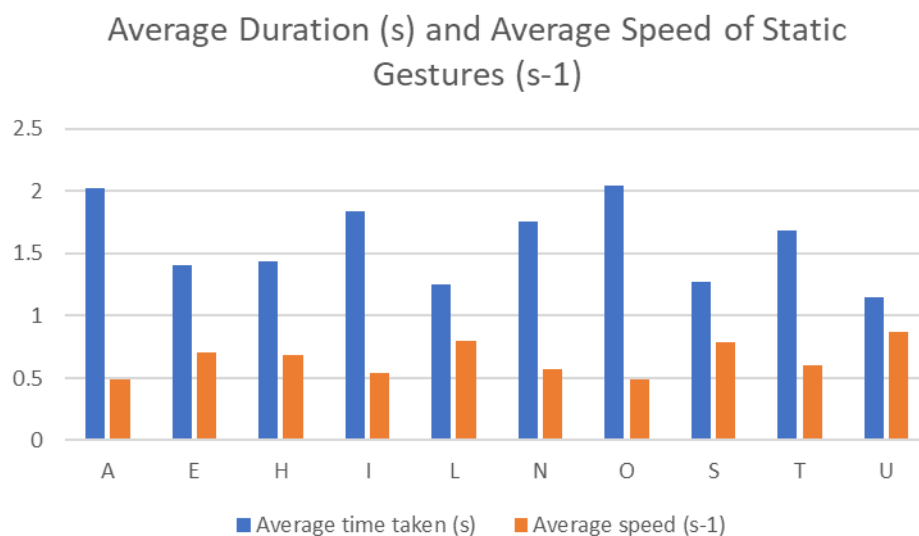


Figure 4.3.1: Bar Chart of Average Duration (s) and Average Speed of Static Gestures (s⁻¹)

The bar chart in Figure 4.3.1 above displays the average duration (in seconds) and average speed (in seconds per gesture) of various static gestures, including A, E, H, I, L, N, O, S, T, and U.

Gesture "U" stands out as the fastest with shortest time taken, and records on an average recognition time of 1.15 seconds per gesture. Gestures "A" and "O" have the slowest recognition times, both with an average of 2.03 seconds per gesture.

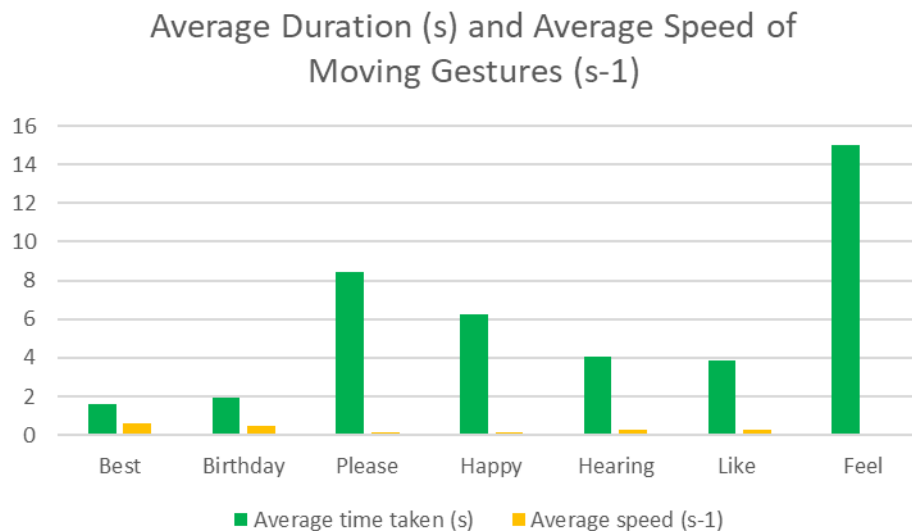


Figure 4.3.2: Bar Chart of Average Duration (s) and Average Speed of Moving Gestures (s⁻¹)

Based on Figure 4.3.2, "Please" has the longest average recognition time of 8.45 seconds, while "Best" is the fastest, with an average recognition time of 1.61 seconds. This suggests that "Please" is recognized slowly, whereas "Best" is recognized more quickly. More complex gestures, such as "Birthday" and "Hearing," take longer to recognize on average, possibly due to the intricacies of the hand movements involved. "Feel" exceeds the time limit for detection specified at 15 seconds. This indicates that more training may be required for the model to recognize this gesture accurately.

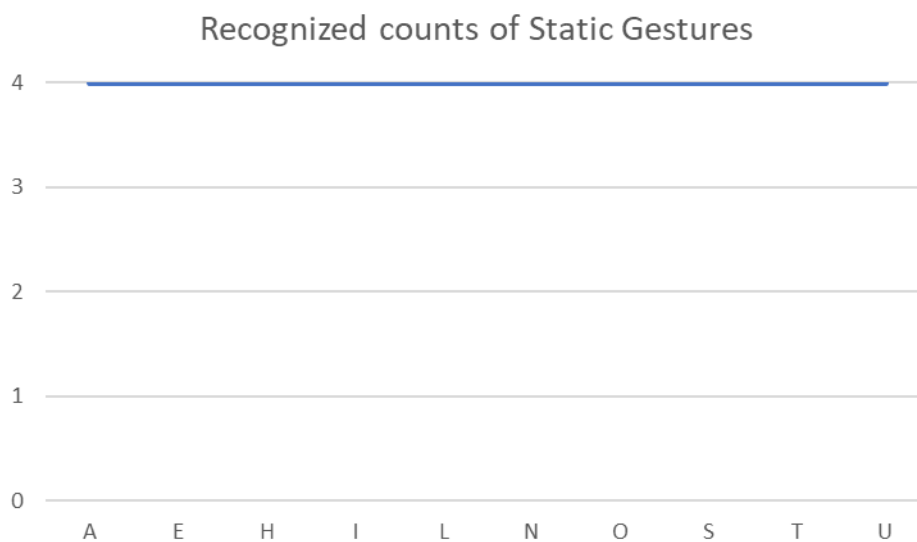


Figure 4.3.3: Line Graph of Recognized counts of Static Gestures

This graph in Figure 4.3.3 demonstrates that all tested static gestures achieved consistent recognition performance, each with four successful recognitions. This uniformity suggests that the model exhibits robust recognition capabilities for the static sign language gestures.

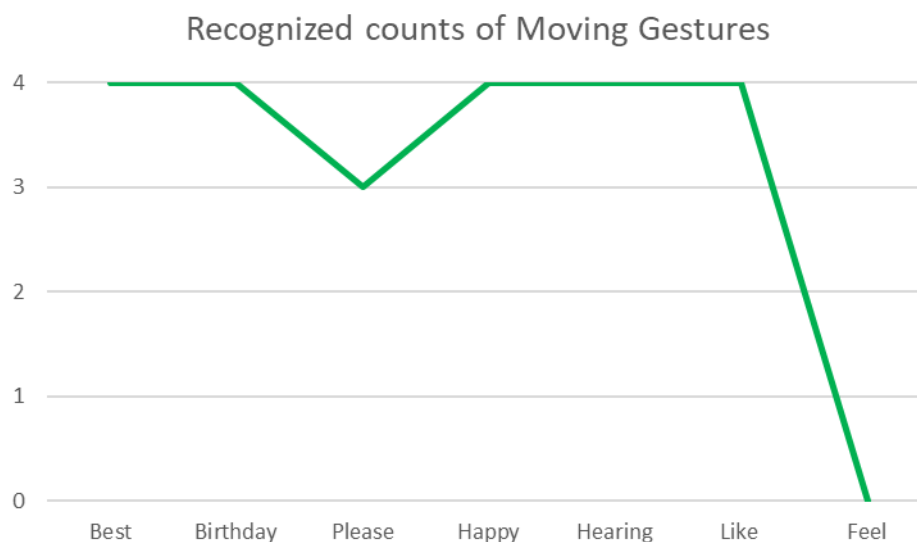


Figure 4.3.4: Line Graph of Recognized counts of Moving Gestures

From the graph in Figure 4.3.4, it is evident that the "Feel" gesture had no successful recognitions during testing, as indicated by a recognized count of 0. In contrast, the "Best," "Birthday," "Happy," "Hearing," and "Like" gestures were recognized four times each, demonstrating consistent and successful recognition. The "Please" gesture had a slightly lower recognition count of 3, indicating that it was recognized with a slightly lower frequency compared to the others.

This data suggests that the model performed well in recognizing most of the moving sign language gestures, except for the "Feel" gesture, which requires further improvement in recognition accuracy.

Tester Feedback

Table 4.3.1 below summarizes feedback and comments received from testers regarding their experiences with the system.

Table 4.3.1: Feedback and Comments by Testers

Tester	Feedback and Comments
Tester 1	Motions which are fast are not detected. Unrecognised moving gestures are classified as 'Best'
Tester 2	The system seems to detect gestures effectively only when the user is in specific body positions.
Tester 3	Two hands not able to detect, complicated hand gestures are detected slower
Tester 4	Complex gestures take long time to detect.

The feedback by Tester 1 suggests the need for improvements in gesture recognition speed and accuracy. The feedback by Tester 2 emphasizes the importance of ensuring the system's robustness across a range of user positions. Tester 3's feedback indicates a need for enhancements in recognizing complex and multi-hand gestures. Tester 4's feedback shows importance of optimizing the system's performance for quicker recognition of complex gestures.

In response to this feedback, Chapter 5 will provide comprehensive recommendations on refining the system. These recommendations will focus on addressing these issues and enhancing the overall performance of the system, ensuring that it can accurately and efficiently detect all types of gestures.

b. For self

Lighting and Distance Variations

The impact of lighting conditions and distances (36cm vs. 53cm) on gesture recognition performance are analysed as shown in Table 4.2.6.2.

Table 4.3.2: Recognition state under different Lighting and Distances

		Recognized (1) / Unrecognized (0)			
		36cm – from table Light condition	53cm – from table Light condition	36cm – from table Dim condition	53cm – from table Dim condition
Static	A	1	1	1	1
	E	1	1	1	1
	H	1	1	1	1
	I	1	1	1	1
	L	1	1	1	1
	N	1	1	1	1
	O	1	1	1	1
	S	1	1	1	1
	T	1	1	1	1
U	1	1	1	1	
Moving	Best	1	1	1	1
	Birthday	1	1	1	1
	Please	1	1	1	1
	Happy	1	1	1	1
	Hearing	1	1	1	1
	Like	1	1	1	1
	Feel	0	0	0	0

In general, the system exhibits a robust performance, with gestures (other than 'Feel') being successfully recognized in both regular and dim lighting conditions, as well as at varying distances from the table.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, this project has successfully achieved the objectives stated earlier in the development of the ASL recognition system. The primary goal was to design and develop an efficient algorithm for the hand gesture recognition, reducing the reliance on trained interpreters for individuals with hearing impairments. The successfully built a TensorFlow Keras API neural network model integrated with OpenCV Mediapipe, achieving an impressive accuracy rate of 96% for static gestures and 53% for dynamic, moving gestures. In addition, based on the testings results, the system could work real-time, under different lightings and distances from the webcam, enabling efficiency to be used in real-time communication scenarios in Objective 2. It is also can be used by various users from different demographics, either gender, age, ethnicity, hand dominance, at all levels of proficiency of ASL, addressing the past limitations and leading to inclusivity, fulfilling Objective 3.

5.2 Recommendations for future work

Based on the findings from Chapter 3, the following recommendations are made for future work:

1. Enhance Point History Classifier

Continue to refine and optimize the Point History Classifier to improve training accuracy. Additional training epochs and fine-tuning may be necessary to achieve higher accuracy levels.

2. Gesture Recognition Speed

Investigate methods to enhance gesture recognition speed, especially for more complex gestures like "Feel." Optimization techniques and model architecture adjustments may be explored.

3. Robustness Across User Positions

Conduct further research and development to ensure the system's robustness across a wide range of user positions and orientations. This may involve collecting additional data from diverse user scenarios.

4. Complex Gesture Recognition

Focus on enhancing the model's ability to recognize complex and multi-hand gestures. This could involve data augmentation, specialized model architectures, or more extensive training.

5. Feedback Integration

Continuously gather feedback from users and testers to iteratively improve the system's performance. Feedback should be carefully analysed and used to guide future development efforts.

By implementing these recommendations, it is expected that the system's overall performance will be significantly enhanced, ensuring accurate and efficient detection of all types of gestures, thereby improving its utility and user satisfaction.

REFERENCES

- Bellugi, U. and Fischer, S., 1972. *A comparison of sign language and spoken language*. *Cognition*, 1(2–3), pp.173–200.
- Bird, J.J., Ekárt, A. and Faria, D.R., 2020. British sign language recognition via late fusion of computer vision and leap motion with transfer learning to american sign language. *Sensors (Switzerland)*, 20(18), pp.1–19.
- Chong, T.W. and Lee, B.G., 2018. American sign language recognition using leap motion controller with machine learning approach. *Sensors (Switzerland)*, 18(10).
- De Coster, M., Van Herreweghe, M. and Dambre, J., Isolated Sign Recognition from RGB Video using Pose Flow and Self-Attention,
- Farooq, U., Asmat, A., Rahim, M.S.B.M., Khan, N.S. and Abid, A., 2019, November. *A comparison of hardware-based approaches for sign language gesture recognition systems*. In *2019 International Conference on Innovative Computing (ICIC)* (pp. 1-6). IEEE.
- Haj, A., Ghoul, O., and Jemni, M., 2017. *Toward sign language handshapes recognition using Myo armband*. *2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA)* (pp. 1-6). IEEE.
- Hou, J. et al., 2019. *SignSpeaker: A real-time, high-precision smartwatch-based sign language translator*. *The 25th Annual International Conference on Mobile Computing and Networking* (pp. 1-15).
- Ismail, A.P., Aziz, F.A.A., Kasim, N.M. and Daud, K., 2021. *Hand gesture recognition on python and opencv*. *IOP Conference Series: Materials Science and Engineering*, 1045(1), p.012043.

- Jiang, S. et al., 2021. *Skeleton Aware Multi-modal Sign Language Recognition. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3413-3423).
- Lee, B.G. and Lee, S.M., 2018. Smart Wearable Hand Device for Sign Language Interpretation System with Sensors Fusion. *IEEE Sensors Journal*, 18(3), pp.1224–1232.
- Lupinetti, K., Ranieri, A., Giannini, F. and Monti, M., 2020. *3D dynamic hand gestures recognition using the Leap Motion sensor and convolutional neural networks.* Available at: <http://arxiv.org/abs/2003.01450>.
- Mohammed, A.A.Q., Lv, J. and Islam, M.D.S., 2019. *A deep learning-based end-to-end composite system for hand detection and gesture recognition. Sensors (Switzerland)*, 19(23).
- National Association of the Deaf, 2023. *National Association of the Deaf - NAD [Online].* Available at: <https://www.nad.org/about-us/position-statements/position-statement-on-health-care-access-for-deaf-patients/> [Accessed: 24 March 2023].
- Naglot, D., Scholar, R. and Kulkarni, M., Real Time Sign Language Recognition using the Leap Motion Controller,
- O' Mahony, N. et al., 2019. *Deep Learning vs. Traditional Computer Vision.* In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1 2020* (pp. 128-144). Springer International Publishing.
- Pop, D.P. and Altar, A., 2014. *Designing an MVC model for rapid web application development. Procedia Engineering.* 2014 Elsevier Ltd, pp. 1172–1179.

- Pragati G., Naveen A. and Sanjeev S.,2009. *Vision Based Hand Gesture Recognition*. International Journal of Computer and Information Engineering, 3(1), pp.186-191.
- Quer, J. and Steinbach, M., 2019. *Handling sign language data: The impact of modality*. *Frontiers in Psychology*, 10, p.483.
- Riaz S., 2022. *A Hand Gesture Sign Language to Text Real Time Interpreter using Google Mediapipe Artificial Intelligence by Riaz Sulaimi | MLearning.ai | Medium [Online]*. Available at: <https://medium.com/mlearning-ai/a-hand-gesture-sign-language-to-text-real-time-interpreter-using-google-mediapipe-artificial-dfb395c42a23> [Accessed: 24 March 2023].
- Shukor, A.Z. et al., 2015. A New Data Glove Approach for Malaysian Sign Language Detection. *Procedia Computer Science*. 2015 Elsevier B.V., pp. 60–67.
- Sulaimi, R., 2022, *A Hand Gesture Sign Language to Text Real Time Interpreter using Google Mediapipe Artificial Intelligence | by Riaz Sulaimi | MLearning.ai | Medium [Online]*. Available at: <https://medium.com/mlearning-ai/a-hand-gesture-sign-language-to-text-real-time-interpreter-using-google-mediapipe-artificial-dfb395c42a23> [Accessed: 21 April 2023].
- Suryateja, C.M., Boppu, S., Cenkeramaddi, L.R. and Ramkumar, B., 2022. Hand Gesture Recognition System in the Complex Background for Edge Computing Devices. *Proceedings - 2022 IEEE International Symposium on Smart Electronic Systems, iSES 2022*. 2022 Institute of Electrical and Electronics Engineers Inc., pp. 13–18.
- Starner, T., Weaver, J. and Pentland, A., 1996. *Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video*. *IEEE Transactions on pattern analysis and machine intelligence*, 20(12), pp.1371-1375.

United Nations, 2022. *International Day of Sign Languages* | United Nations [Online]. Available at: <https://www.un.org/en/observances/sign-languages-day> [Accessed: 24 March 2023].

Woll, B., Sutton-Spence, R. and Elton, F., 2001. *Multilingualism: The global approach to sign languages. The sociolinguistics of sign languages*, 8, p.32.

World Health Organization, n.d. *Hearing loss* [Online]. Available at: https://www.who.int/health-topics/hearing-loss#tab=tab_1 [Accessed: 24 March 2023].

Zaidman-Zait, A. and Dotan, A., 2017. *Everyday stressors in deaf and hard of hearing adolescents: The role of coping and pragmatics. Journal of Deaf Studies and Deaf Education*, 22(3), pp.257–268.