



**Faculty of Computing and Information
Technology
(King Abdulaziz University)**

**Image-Based Sign Language Translation using Deep
Learning**

Research Report Submitted in Partial Fulfilment
of the Requirements for the Degree of Science in
(Computer Science)

BY

Maha Alharbi

Bashayer Alghanami

Ashwaq Almutiri

Noof Alghanami

Areej Almutiri

Supervised By: Dr. Andi Besse Firdausiah Mansur

2024



In the name of Allah, the most merciful, the most beneficent

CERTIFICATE

This is to certify that this research report entitled **Image-Based Sign Language Translation using Deep Learning** by Maha Alharbi (Roll No. 2005029) , Bashayer Alghanami (Roll No. 2005956), Ashwaq Almutiri (Roll No. 2010706), Noof Alghanami (Roll No. 2005107) And Areej Almutiri (Roll No. 2010459), submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer science of the King Abdulaziz University, Rabigh, during the academic year 2023-2024, is an original record of work carried out under our guidance and supervision. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Supervisor

Head of the Department

Dr. Andi Besse Firdausiah Mansur

Dr. Nouf Alghanmi

DEDICATION

We express our deep gratitude to all those who have supported us, both now and throughout our academic journey. We are immensely thankful to our friends who stood by us, helped us overcome difficulties, and provided support. Our heartfelt appreciation goes to those who made great sacrifices for us, especially our parents who consistently encouraged us with love to pursue education and give our best. We also commend ourselves for not giving up and remaining committed to achieving our dreams, regardless of the challenges faced. We extend our thanks to the numerous individuals who assisted us along our journey, including our research supervisor and the doctors/professors who guided us. Our sincerest gratitude goes out to everyone who played a part in our success and contributed to our growth.

ACKNOWLEDGMENTS

We would like to express our special thanks of gratitude to our Head of Department Dr. Nouf Alghanmi as well as our supervisor Dr. Andi Besse Firdausiah who gave us the golden opportunity to do this wonderful research on the topic “Image-Based Sign Language Translation using Deep Learning “.

Secondly, we would also like to thank our parents and friends who helped us a lot in finalizing this research within the limited time frame.

ABSTRACT

Image-Based Sign Language Translation using Deep Learning is an innovative effort that makes use of computer vision and deep learning techniques to interpret sign language from photographs. This research aims to promote inclusive communication for the deaf community. An new solution for real-time sign language interpretation is provided by the system that has been proposed. This solution is achieved through the utilization of advanced image processing and gesture recognition. This assistive technology is designed to bridge communication gaps by converting visual cues into words that can be understood by those with hearing impairments. This technology also attempts to enhance accessibility and understanding for those individuals. MediaPipe is used to collect the data about the images. The model will then be subjected to the pre-trained weight, deep learning parameter, and feature extraction techniques, and any necessary adjustments will be made to the model. The result of the recognition is pretty remarkable, with an accuracy of more than 97%. Instead of focusing on recognising individual words, the work that will be done in the future could be done on whole sentences.

Keywords: Sign language translation, computer vision, image processing, deep learning, accessibility, gesture recognition, language interpretation.

ABSTRACT (ARABIC)

تعد ترجمة لغة الإشارة المبنية على الصور باستخدام التعلم العميق جهداً مبتكراً يستخدم رؤية الكمبيوتر وتقنيات التعلم العميق لتفسير لغة الإشارة من الصور الفوتوغرافية. يهدف هذا المشروع إلى تعزيز التواصل الشامل لمجتمع الصم. يتم توفير حل جديد لترجمة لغة الإشارة في الوقت الحقيقي من خلال النظام الذي تم اقتراحه. يتم تحقيق هذا الحل من خلال استخدام معالجة الصور المتقدمة والتعرف على الإيماءات. تم تصميم هذه التقنية المساعدة لسد فجوات التواصل عن طريق تحويل الإشارات البصرية إلى كلمات يمكن أن يفهما الأشخاص الذين يعانون من ضعف السمع. تحاول هذه التكنولوجيا أيضاً تعزيز إمكانية الوصول والفهم لهؤلاء الأفراد. يتم استخدام لجمع البيانات حول الصور. سيتم بعد ذلك إخضاع النموذج للوزن المُدرَّب مسبقاً، ومعلمة التعلم MediaPipe العميق، وتقنيات استخلاص الميزات، وسيتم إجراء أي تعديلات ضرورية على النموذج. نتيجة التعرف رائعة جداً، بدقة تزيد عن 97%. وبدلاً من التركيز على التعرف على الكلمات الفردية، فإن العمل الذي سيتم إنجازه في المستقبل يمكن أن يتم على جمل كاملة.

الكلمات المفتاحية: ترجمة لغة الإشارة، رؤية الكمبيوتر، معالجة الصور، التعلم العميق، إمكانية الوصول، التعرف على الإيماءات، ترجمة اللغة.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DEDICATION	iv
	ACKNOWLEDGMENTS	v
	ABSTRACT	vi
	ABSTRACT (ARABIC)	vii
	TABLE OF CONTENTS	viii
	LIST OF TABLES	xi
	LIST OF ABBREVIATIONS	xiv
	CHAPTER 1	1
	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Problem Background	1
	1.3 Research Aim	2
	1.4 Research Question	2
	1.5 Research Objectives	2
	1.6 Research Scope	3
	1.7 Research Contribution	3
	1.8 Report Organization	3
	CHAPTER 2	4
	LITERATURE REVIEW	4
	2.1 Introduction	4
	2.2 Problem Formulation	4
	2.2.1 Research Domain	5
	2.2.2 Description of Related Studies	6
	2.2.2.1 Using machine learning and cameras to convert sign language to text.	8
	2.2.2.2 Features Korean-KSL data pairs for sign language production	9
	2.2.2.3 Glove-based sign language translator device	9
	2.2.2.4 Develop represents a multi-IMU-based data glove system.	10
	2.2.2.5 Sign language interpretation system incorporating a Micro Touch	10

2.2.2.6 Development of translating sign language by analysing finger and hand movements.	11
2.3 Comparison of the researchs	11
2.4 Proposed Solutions	12
2.5 Chapter Summary	13
CHAPTER 3	14
RESEARCH METHODOLOGY	14
3.1 Introduction	14
3.2 Operational Framework/Research Workflow	14
3.2.1 System Design (Phase 1)	16
3.2.1.1 Data collection	16
3.2.1.2 Dataset preparation	16
3.2.1.3 Feature extraction	17
3.2.2 Training of Deep Learning Classifier (Phase 2)	17
3.2.4 System Implementation (Phase 3)	17
3.2.4 Performance Evaluation (phase 4)	18
3.3 Analysis of System Requirements	19
3.4 Justification	21
3.5 Performance measurement	21
3.6 Chapter Summary	22
CHAPTER 4	23
RESEARCH DESIGN AND IMPLEMENTATION	23
4.1 Introduction	23
4.2 Proposed Solution	23
4.3 Experiment Design	24
4.3.1 Implementation Environment	24
4.3.1.1 Excel	24
4.3.1.2 Visual studio code	25
4.3.1.3 Python	26
4.3.1.4 Jupyter Notebook	27
4.3.2. Dataset Collection	28
4.3.3 Features Extraction	29
4.3.4 Preparation of Training Dataset	29
4.3.4 Training of Deep Learning model	30
4.3.5 Testing of Deep Learning model	31
4.3.Feature Selection Using Deep Learning model	33

4.4 Parameter and Testing Method	34
4.4.1 Feature Selection	34
4.4.2 Model Architecture	35
4.4.3 Training Hyperparameters	35
4.4.4 Testing Methods	37
4.5 Chapter Summary	38
Chapter 5	39
RESULTS, ANALYSIS AND DISCUSSION	39
5.2 Introduction	39
5.3 Experiment/Coding	39
5.2.1 Get_image File	39
5.2.2 Get_data File	40
5.2.3 Test_mediapipe File	42
5.2.4 Train File	43
5.2.5 Main File	44
5.4 Research Results and Analysis	46
5.5 Chapter Summary	54
Chapter 6	55
CONCLUSION	55
6.1 Introduction	55
6.2 Achievements of Research Objectives	55
6.3 Limitations and Future Works	55
REFERENCES	56

LIST OF TABLES

TABLE NO.	TITLE	PAGE
Table 2.1:	Information similar researches	7
Table 2.2:	The comparisons of our proposed research with existing ones	11
Table 5. 1:	Confusion Matrix Metrics Table	53

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
Figure 2. 1:	Proposed Solutions	13
Figure 3. 1:	Research Methodology	15
Figure 4. 1:	The framework the proposed intelligent Sign Language Translation based on Deep Learning	23
Figure 4. 2:	The Excel program.	25
Figure 4. 3:	visual studio code program.	26
Figure 4. 4:	Jupyter Notebook interface in vs code.	27
Figure 4. 5:	The collected sign language dataset.	28
Figure 4. 6:	Sample of the training dataset with extract key landmarks on the hands	29
Figure 4. 7:	Training dataset on Python.	30
Figure 4. 8:	Example of the architecture of Deep learning	31
Figure 4. 9:	uploaded the dataset	32
Figure 4. 10:	Feature Selection Using Deep Learning model	33
Figure 4. 11:	ASL sign language translation frame.	33
Figure 4. 12:	image_processed function	34
Figure 4. 13:	data extract to identify hand landmarks	35
Figure 4. 14:	sequential indices to DataFrame columns	36
Figure 4. 15:	training process	36
Figure 4. 16:	The result of testing	37
Figure 5. 1:	Python code for capturing the image from Camera.	40
Figure 5. 2:	MediaPipe automates hand landmark detection on images.	41
Figure 5. 3:	Real-time visualization of hand landmarks detected from a webcam feed using Media Pipe and OpenCV.	42
Figure 5. 4:	Real-time ASL sign language translation using deep learning model.	45
Figure 5. 5:	Result of ASL Translation for the 'i love you' Sign	46

Figure 5. 6: Result of ASL Translation for the 'Hello' Sign	47
Figure 5. 7: Result of ASL Translation for the 'alright' Sign	47
Figure 5. 8: Result of ASL Translation for the 'bye' Sign	48
Figure 5. 9: Result of ASL Translation for the 'always' Sign	48
Figure 5. 10: Result of ASL Translation for the 'no' Sign	49
Figure 5. 11: Result of ASL Translation for the 'okay' Sign	49
Figure 5. 12: Result of ASL Translation for the 'thank you' Sign	50
Figure 5. 13: Result of ASL Translation for the 'yes' Sign	50
Figure 5. 14: Result of ASL Translation for the 'Help' Sign	51
Figure 5. 15: Confusion Matrix - American Sign Language	52
Figure 5. 16: Line Graph of FPR, TNR, and FPN for Each Class	53
Figure 5. 17: evaluation metrics of a machine learning model	54

LIST OF ABBREVIATIONS

ASL	- American Sign Language
Attention-BiLSTM	- attention-based mechanism of long and short-term memory
DT	- Decision Tree
IMU	- Inertial Measurement Units
ISLRTC	- Indian Sign Language Research and Training Centre
KNN	- K-nearest neighbour method
LCD	- Liquid Crystal Display
RCNN	- Region Based Convolutional Neural Networks
RF	- Random Forest
ROI	- Region Of Interest
RPN	- Region Proposal Network
SLP	- Sign Language Production
SLT	- Sign language translation
CSV	- comma-separated values

CHAPTER 1

INTRODUCTION

1.1 Introduction

Artificial intelligence and deep learning have revolutionized the field of machine learning, enabling the development of sophisticated models capable of detecting patterns and making predictions based on complex sets of data. In recent years, significant advances have been made in the development of AI systems that can analyse images, texts, and videos, as well as perform other tasks that were previously thought impossible. This research seeks to investigate the application of AI and deep learning methods to develop an automated system for object detection in images. Specifically, to address the challenge of communication barriers faced by individuals with hearing impairments who rely on sign language.

1.2 Problem Background

Vishwas S et al. proposed an innovative approach using machine learning and cameras to convert sign language gestures to text in regional languages. The image captured by the webcam is processed with the tf-pose-estimation library using decision trees and pose estimation to recognize signs [1]. Mathew Huerta-Enochian et al. developed using features Korean-KSL data pairs for sign language production and translation. The authors recorded signers from multiple camera angles and collected key point position data and annotations for hand and non-manual signals. They evaluated different translation approaches and used computer vision and natural language processing techniques to analyse the data. Crowd-sourcing was also used to collect translations [2]. Radzi Ambar et al. they design of a glove-based sign language translator device. The developed device is able to read the movements of every finger and arm using two types of sensors, an accelerometer and five unit of flex sensors [3]. Ang ji et al. They develop represents a multi-IMU-based data glove system that

captures hand gestures and finger motions for recognition of 20 instances of deaf sign language using traditional machine learning and deep learning approaches. [4]. Machine learning and computer vision have enabled the development of innovative sign language translation systems, all sharing the goal of bridging the communication gap between the deaf and hearing communities, with hope of further improvement.

1.3 Research Aim

The goal of this research is to develop a software solution that will provide proficient and precise translations of sign language into words. The proposed approach utilizes deep learning algorithms, in conjunction with computer vision to capture and interpret the movements and gestures intrinsic to sign language.

1.4 Research Question

There are some questions that we must address before starting the research:

1. What is the motivation of the research?
2. What approach is being used to recognize sign language?
3. What type of deep learning will be utilized for the system?
4. How does deep learning play a role in image-based recognition techniques?

1.5 Research Objectives

The objectives of the proposed research are as follows:

- 1) Study research and application for sign language system.
- 2) Produce a framework of sign language gesture based on convolutional neural network.
- 3) Develop and evaluate the sign language gesture recognition system.

1.6 Research Scope

The scope of research cover:

- 1) Sign language Dataset is from online standard dataset and manually collected through camera with several research studies.
- 2) Translation is Through motion recognition that is detected by the camera device.

1.7 Research Contribution

The research focuses on real-time sign language detection utilizing deep learning and computer vision introduces a fresh methodology, techniques for data collection and preprocessing, suggestions for model selection and adaptation, strategies for transfer learning, evaluation metrics, and guidelines for real-time implementation. These contributions propel advancements in the fields of artificial intelligence and assistive technology by delivering a thorough framework for developing sign language recognition systems. The emphasis on best practices in data preparation, model adaptation, and real-time implementation enhances the current state of sign language recognition through innovative approaches and valuable guidance for researchers and developers in computer vision and assistive technology.

1.8 Report Organization

This Research consists of six chapters: Started with Chapter 1 that discussed the problem background and motivation of the research. Followed with Chapter 2 that mainly discussed the related work and chapter 3 presenting the methodology of the research. Chapter 4 will focus on research design of the proposed solution. While Chapter 5 present result, analysis, and discussion. Finally, Chapter 6 focuses on conclusion, and future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This paragraph discusses the concept of a deep-learning program that aims to bridge the communication gap between individuals who use sign language and those who do not. By utilizing computer vision techniques and machine learning algorithms, this program translates sign language into words. The following chapter will delve into the existing research on AI programs that convert sign language into words, as well as provide an overview of the methodologies employed in these studies.

2.2 Problem Formulation

In the domain of Image-Based Sign Language Translation using Deep Learning, formulating a problem statement involves identifying the specific challenges and objectives inherent in the field. At its core, the challenge lies in accurately and promptly translating sign language gestures captured in image or video formats into understandable spoken or written language. This complex task comprises several interrelated sub-problems that require detailed exploration. Initially, Gesture Recognition demands the development of algorithms capable of precisely identifying individual sign language gestures from input images or video frames, necessitating a deep understanding of the intricate movements and configurations inherent in sign language gestures. Following recognition, the challenge of Gesture Representation arises, involving the conversion of identified gestures into a suitable representation for translation, often through the transformation into feature vectors or sequences amenable to processing by deep learning models. The subsequent hurdle, the Translation Model, requires the design and training of models proficient in translating gesture representations into spoken or written language, capable of comprehending nuanced meanings and expressing them coherently. Moreover, achieving real-time translation performance is crucial, necessitating the implementation of efficient algorithms and architectures that operate within computational constraints, ensuring

timely and responsive translations. Additionally, Cross-Linguistic Adaptation adds complexity, as sign language exhibits variations across different languages and regions, demanding sophisticated adaptation techniques for accurate translation across diverse user groups.

The objectives of the scientific research on Image-Based Sign Language Translation using Deep Learning are multifaceted. They encompass achieving high accuracy and robustness in recognizing and translating sign language gestures under various environmental conditions and user contexts. Furthermore, the research aims to develop efficient real-time processing capabilities by designing and implementing deep learning architectures and algorithms capable of processing sign language gestures seamlessly. Additionally, the research endeavours to adapt translation models to handle different sign languages and regional variations, ensuring widespread accessibility and usability among diverse user communities. Lastly, integration into user-friendly applications and devices is paramount, fostering widespread adoption and usability among both sign language users and non-signers. This integration involves considerations for user interface design, accessibility features, and seamless integration with existing communication technologies, thereby enhancing accessibility and facilitating effective communication for all users.

2.2.1 Research Domain

In the field of sign language recognition (SLR), a crucial area of research within human-computer interaction, traditional machine learning techniques such as Deep Learnings remain highly relevant. Our system incorporates Deep Learning in conjunction with Media Pipe, a sophisticated framework for real-time feature extraction, to classify hand landmarks with high precision. This approach contrasts with deep learning methods, which often involve convolutional neural networks (CNNs) or recurrent neural networks (RNNs) to automatically learn features from extensive datasets of sign language gestures. While Deep Learning is adept at handling high-dimensional data and providing clear decision boundaries, deep learning models excel in recognizing complex patterns and temporal sequences in sign language gestures, potentially offering greater accuracy through their ability to learn hierarchical features. By integrating Deep Learning with Media Pipe's efficient landmark detection, our system provides a computationally less intensive solution, suitable for applications where

real-time performance and low resource consumption are crucial. Further exploration of hybrid models that combine the meticulous feature detection capabilities of Media Pipe with the sophisticated learning mechanisms of deep learning could pave the way for advancements in SLR systems, aiming to enhance their robustness and adaptability across diverse communication contexts.

2.2.2 Description of Related Studies

There are various studies conducted in the field of sign language translation that focus on developing and improving the accuracy of sign language recognition, translation. In the studies listed in table 2.1, a data glove is used to scan the hand movement to translate American sign language (ASL). In other studies, a camera was used to capture signs and video processing to perform acknowledgement of the sign language, contrasted with data glove. Several researchs have been developed to convert sign language into text, utilizing machine learning and computer vision techniques. One such research by Vishwas S et al [1] .involved collecting videos of sign language gestures, preprocessing them using OpenCV, and training a Convolutional Neural Network (CNN) to recognize the gestures. The translated text was converted into speech using a text-to-speech engine, achieving an accuracy of 92%. Another research, conducted by the KoSign Sign Language Translation Research, focused on a Korean-KSL dataset for sign language production. They used similar preprocessing and machine learning techniques to recognize gestures, provided recommendations for improving translation quality, and presented their findings at an international workshop [2]. Additionally, wearable glove-based devices have been developed by researchers such as Radzi Ambar [3], utilizing flex sensors and accelerometers to detect finger and arm movements, thus enabling the translation of sign language into speech and text. The detailed comparison between different aspects is presented in Table 2.1. discusses similar existing researchs to distinguish between their aims, objectives, and methodologies.

Table 2.1 : Information similar researches

	Author	Aim and objective	Methodology
1	Vishwas S et al [1]	proposed a innovative approach using machine learning and cameras to convert sign language gestures to text in regional languages.	The image captured by the webcam is processed with the tf-pose-estimation library using decision trees and pose estimation to recognize signs
2	Mathew Huerta-Enochian et al [2]	developed using features Korean-KSL data pairs for sign language production and translation	The authors recorded signers from multiple camera angles and collected key point position data and annotations for hand and non-manual signals. They evaluated different translation approaches and used computer vision and natural language processing techniques to analyse the data. Crowd-sourcing was also used to collect translations [2]
3	Radzi Ambar et al [3]	Design of a glove-based sign language translator device	The developed device is able to read the movements of every fingers and arm using two types of sensor, an accelerometer and five unit of flex sensors [3]
4	Ang ji et al. [4]	Finding ways to enable seamless communication between deaf and able-bodied individuals has been a challenging and pressing issue.	They develop represents a multi-IMU-based data glove system that captures hand gestures and finger motions for recognition of 20 instances of deaf sign language using traditional machine learning and deep learning approaches. [4]

5	Sign Language Interpretation Using Machine Learning and Computer Vision (Gordon McMaster) [5]	This research aims to create a sign language interpretation system incorporating a MicroTouch switch, with the aim of facilitating real-time communication for individuals who are unable to speak or hear.	Components used in this system include MicroTouch switch, Arduino UNO, LCD display, amplifier, micro-DF driver, and Bluetooth module (HC-05).
6	PIERRE SINANDER TOMAS ISSA [6]	Development of a data glove capable of translating American Sign Language (ASL) by analysing finger and hand movements.	Bluetooth Low Energy technology.

2.2.2.1 Using machine learning and cameras to convert sign language to text.

Vishwas S et al [1], Vishwas has developed the sign language translator using machine learning and computer vision. The first step was to collect data in the form of videos of sign language gestures, which were collected from various sources, including the Indian Sign Language Research and Training Centre (ISLRTC). The collected videos were pre-processed to extract the features of the sign language gestures using OpenCV, a computer vision library, and were stored in a database. The machine learning model used to recognize the sign language gestures was a deep learning algorithm called Convolutional Neural Network (CNN). The model was trained on the extracted features of the sign language gestures. The translated text was then converted into speech using a text-to-speech engine. The sign language translator was developed using a Raspberry Pi, a low-cost, credit-card-sized computer. The research was tested on a dataset of sign language gestures and was found to have an accuracy of 92%.

2.2.2.2 Features Korean-KSL data pairs for sign language production

The KoSign Sign Language Translation Research developed a sign language production (SLP) and sign language translation (SLT) dataset called NIASL2021. The dataset consists of 201,026 Korean-KSL data pairs and was created to support KoSign, a sign language translation (SLT) and sign language production (SLP) development research. The collected videos were pre-processed to extract the features of the sign language gestures using OpenCV, a computer vision library, and were stored in a database. The machine learning model used to recognize the sign language gestures was a deep learning algorithm called Convolutional Neural Network (CNN). The model was trained on the extracted features of the sign language gestures. The translated text was then converted into speech using a text-to-speech engine. The research evaluated its sign language elicitation methodology and found that text-based prompting had a negative effect on translation quality in terms of naturalness and comprehension. The research recommended distilling text into a visual medium before translating into sign language or adding a prompt-blind review step to text-based translation methodologies. The research was developed by a team that includes Mathew Huerta-Enochian. The research was presented at the 7th International Workshop on Sign Language Translation and Avatar Technology in Marseille, France in June 2022[7][8][9].

2.2.2.3 Glove-based sign language translator device

Radzi Ambar is a researcher who has contributed to the development of a wearable device for sign language recognition. The device is a glove-based translator that converts sign language into speech and text. The device is able to read the movements of a single arm and five fingers using five flex sensors to detect finger bending and an accelerometer to detect arm motions. Based on the combination of these sensors, the device is able to identify any particular gestures that correspond to words and phrases in American Sign Language (ASL) and translate them into speech via a speaker and text displayed on an LCD screen. Radzi Ambar's contribution to the research is not specified in the search results, but they are listed as one of the authors of a paper that discusses the development of a wearable device for sign language recognition [10].

2.2.2.4 Develop represents a multi-IMU-based data glove system.

The research aimed to develop a low-cost data glove that utilizes multiple inertial sensors for sign language recognition. The data glove was designed to recognize 20 different types of dynamic sign language data used by deaf individuals. The research employed four machine learning models, including decision tree (DT), support vector machine, K-nearest neighbour method (KNN), and random forest (RF), to recognize the sign language gestures. Additionally, a proposed attention-based mechanism of long and short-term memory neural networks (Attention-BiLSTM) was utilized in the process. The research verified the impact of the number and position of data glove nodes on the accuracy of recognizing complex dynamic sign language. The proposed method was compared with existing state-of-the-art algorithms using nine public datasets. The results indicate that both the Attention-BiLSTM and RF algorithms have the highest performance in recognizing the twenty dynamic sign language gestures, with an accuracy of 98.85% and 97.58%, respectively. The research provides evidence for the feasibility of the proposed data glove and recognition methods. The research was developed by a team that includes Ang Ji and was presented at the International Conference on Applied Physics and Mathematics in Bali, Indonesia in January 2022[11][12].

2.2.2.5 Sign language interpretation system incorporating a Micro Touch

This research [13][14] describes the development of a sign language translator that converts sign language into speech and text using a wearable device. The glove-based device is designed to read the movements of a single arm and five fingers, specifically for American Sign Language (ASL). The device consists of five flex sensors to detect finger bending and an accelerometer to detect arm motions. By combining these sensors, the device can identify gestures corresponding to words and phrases in ASL. The translation is provided through a speaker for speech and an LCD screen for text display. The hardware design of the device is the main focus of this article [13]. The research demonstrates the usefulness of the proposed device, with preliminary experimental results showing an average value of 0.74 seconds to convert a sign language into speech and text.

2.2.2.6 Development of translating sign language by analysing finger and hand movements.

The research [6] developed a glove-based device that reads the movements of a single arm and five fingers to identify sign language gestures. The device consists of five flex sensors to detect finger bending and an accelerometer to detect arm motions. The device is able to identify gestures that correspond to words and phrases in American Sign Language (ASL) and translate them into speech via a speaker and text displayed on an LCD screen. The research focused mainly on the hardware design of the device. The research demonstrated the usefulness of the proposed device, with preliminary experimental results showing an average value of 0.74 seconds to convert a sign language into speech and text. The research was developed by a team of students at KTH Royal Institute of Technology in Stockholm, Sweden and was presented in a report in February 2021. The research adds to the growing body of research on wearable devices for sign language recognition and translation, using different types of sensors and machine learning algorithms to recognize the sign language gestures [15].

2.3 Comparison of the researches

Table 2.2 provide comprehensive comparison between features of the proposed research with the previous related works.

Table 2.2: The comparisons of our proposed research with existing ones

Main functionalities		Research						
		1	2	3	4	5	6	Proposed System
Camera	Webcam	√	-	-	-	-	-	√
accelerometer		-	-	√	-	-	-	-
Bluetooth		-	-	-	-	√	√	-
Microcontroller board	Raspberry Pi	-	-	-	-	-	-	-

	Arduino Uno	-	√	-	-	√	-	-
Method	IMU	-	-	-	√	-	-	-
	computer vision	-	√	-	-	-	-	√
	natural language	-	√	-	-	-	-	-
	LCD	-	-	-	-	√	-	-
	DF driver	-	-	-	-	√	-	-
Hardware	RFID Sensors	-	√	-	-	-	-	-

2.4 Proposed Solutions

The proposed solution adopts an efficient approach, initiating with the direct extraction of hand landmarks using the Media Pipe framework. This step eliminates the need for traditional preprocessing techniques such as normalization and resizing, as Media Pipe provides a standardized output of hand landmarks that are immediately usable for classification purposes. The dataset, comprising these hand landmarks, is systematically organized to facilitate efficient training, and testing of the machine learning model. Within the recognition pipeline, our system employs Media Pipe's advanced capabilities to accurately detect and track hand landmarks in real-time. This method significantly reduces the computational overhead by focusing directly on relevant features without the need for separate Region of Interest (ROI) identification or Region Proposal Networks (RPN) commonly used in more complex computer vision tasks. Instead of traditional feature selection mechanisms, the system utilizes the structured and precise landmark data as features for a Deep Learning classifier. This approach avoids the computational complexity associated with deep learning models and does not require extensive data augmentation or transformation. Central to the solution's efficacy is its reliance on the robust combination of Media Pipe for feature.

Extraction and Deep Learning for gesture classification, ensuring high accuracy and low latency, making it ideal for real-time applications in sign language recognition as depicted in Figure 2.1. Through a process of iterative training, the model refines its ability to accurately map sign language gestures to their corresponding written counterparts, achieving superior translation performance. Moreover, the solution is designed to provide live translation capabilities through real-time computer vision processing. By integrating with appropriate hardware and software infrastructure, such as cameras and streaming platforms, the system can analyse sign language gestures as they occur, instantly translating them into written language. This live translation feature enhances accessibility and inclusivity, empowering individuals who use sign language to communicate seamlessly in various contexts.

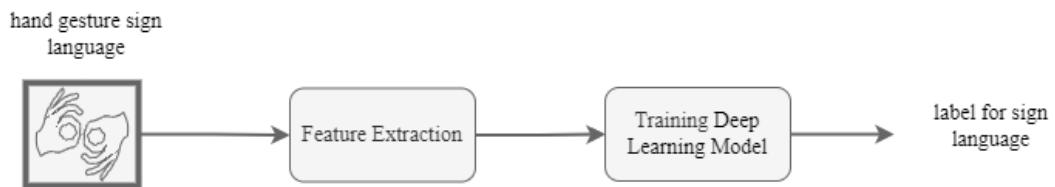


Figure 2. 1: Proposed Solutions

2.5 Chapter Summary

This chapter summarizes a chapter on some researches similar to ours and some information and research on them that can help us in the work of the research so that we can obtain part of some of them in developing and understanding more of our research.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In the research methodology section, we outline the systematic approach adopted to investigate the proposed solution for hand gesture recognition using Deep Learning and Media Pipe. This section provides a comprehensive overview of the research framework, justification for the chosen methodology, and methods for performance measurement. Additionally, a summary of the chapter's key findings is provided.

3.2 Research Workflow

The proper selection of methodology is crucial in the development of a successful system. This methodology provides fundamental guidelines for each stage of the process, enabling successful research task accomplishment. This section details the necessary steps and provides specific instructions for constructing each phase of the chosen methodology. As part of this report, we will explore the suitable methodology required for the research, as depicted in Figure 3.1.

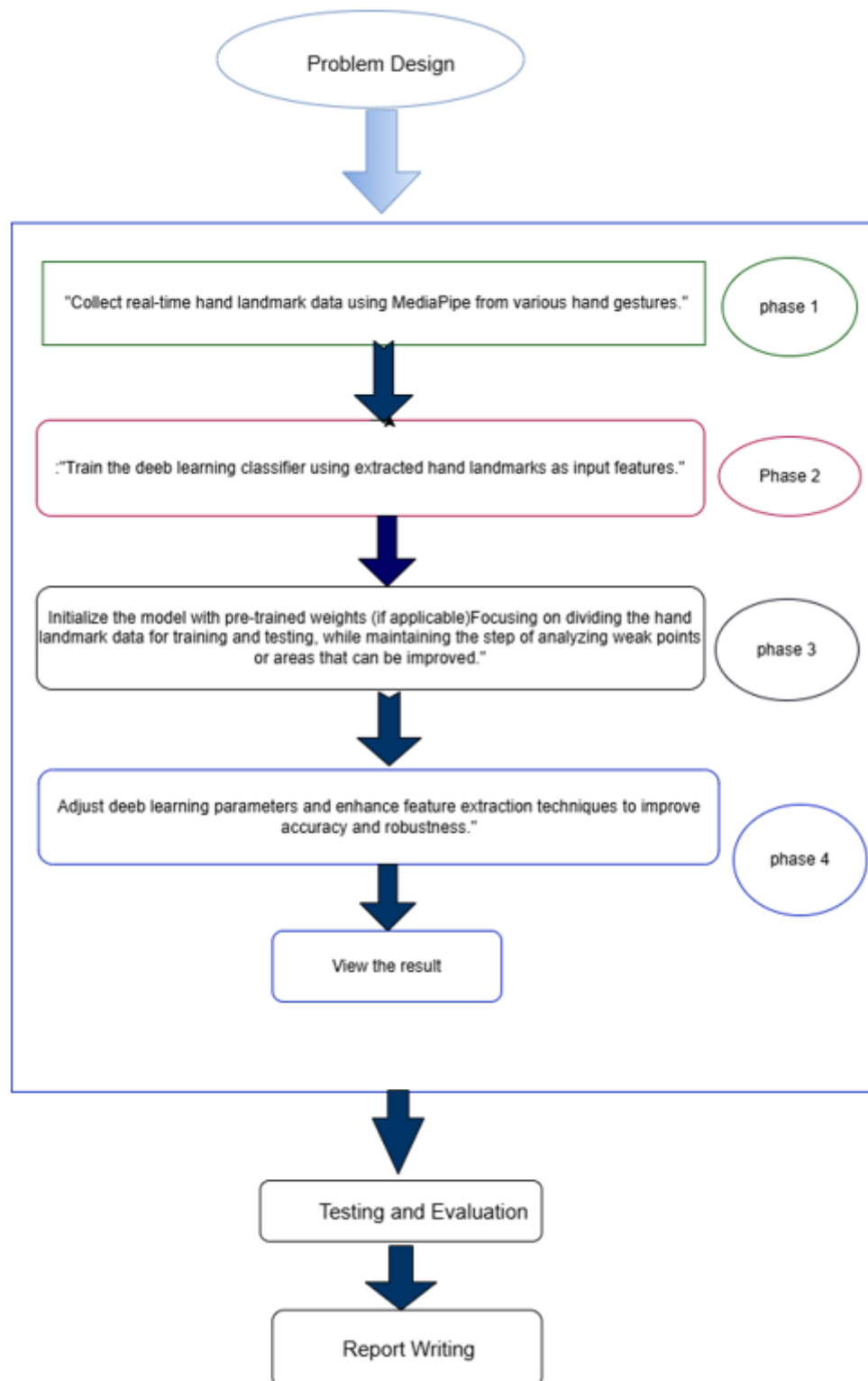


Figure 3. 1 Research Methodology

The process begins with the formulation of a problem design, followed by the collection of hand landmarks through the use of MediaPipe throughout the first step of the approach. Following this, the trained deep learning classifier will extract information from the input. Apply the pre-trained weight, deep learning parameter, and feature extraction procedures to the

model and adjust as necessary. Following the viewing of the results, testing, and evaluation, this was followed by the preparation of a report.

3.2.1 System Design (Phase 1)

In the phase of developing the proposed approach for Image-Based Sign Language Translation using Deep Learning, the framework involves multiple steps to ensure the effectiveness and accuracy of the translation model. The framework includes the following key stages:

3.2.1.1 Data collection

In this research, we collected data by recording various hand gestures using a camera. We then used Media Pipe to extract important hand landmarks from these recordings, like fingertips and knuckles. It was crucial to include a wide range of gestures and variations in lighting and background to ensure our model could recognize different signs accurately. We labeled each gesture to teach the model what each hand movement means. Additionally, we adjusted the data to make sure our model could handle different situations, like changing hand positions or lighting conditions. Overall, our data collection process aimed to create a diverse and reliable dataset for training our sign language recognition system.

3.2.1.2 Dataset preparation

For our system, the dataset preparation involves collecting a diverse set of hand gesture images from the 'DATASET' directory. These images are processed to extract hand landmarks using Media Pipe, rather than undergoing traditional preprocessing steps like resizing, normalization, and data augmentation. The landmarks, which serve as direct inputs for the Deep Learning classifier, are annotated with corresponding categories representing different sign language gestures. The dataset is then divided into training and testing subsets, as validation is not explicitly required for Deep Learning in our initial setup. This streamlined approach ensures that the dataset is optimally prepared for training and evaluating the hand gesture classification system using Deep Learning, focusing on efficiency and direct applicability.

3.2.1.3 Feature extraction

In our research, feature extraction is significantly enhanced by utilizing Media Pipe, which directly provides precise spatial coordinates of hand landmarks. These features are crucial for our Deep Learning-based classification model:

- 1) **Landmark Features:** Instead of traditional shape and size metrics, our system uses Media -Pipe to extract detailed landmarks of the hand. These include the positions of each joint, which inherently describe the shape and orientation of the hand without the need for manual feature description.
- 2) **Temporal Features:** While our current setup does not process motion over time, the framework can extend to incorporate temporal changes in hand gestures if needed, by analysing sequences of landmark positions across frames.
- 3) **Deep Learning-Compatible Features:** Unlike systems that rely on deep learning features from CNNs, our Deep Learning classifier uses the landmark data as feature vectors. This method is computationally less intensive and does not require the extensive data and computational resources typical of deep learning approaches.

3.2.2 Training of Deep Learning Classifier (Phase 2)

In our system, we train the Deep Learning classifier using hand landmarks data extracted by Media Pipe to classify sign language gestures. The training process involves feeding the extracted landmarks into the Deep Learning, which are used as input features. These features are highly informative, representing key points of hand gestures, and the Deep Learning is adjusted to optimize its performance in recognizing these gestures accurately. The objective is to train the classifier to effectively distinguish between different sign language gestures based on the input landmarks. By focusing on the direct, computationally efficient extraction of hand landmarks via Media Pipe, our system ensures high accuracy and low latency, making it ideal for real-time applications in sign language recognition.

3.2.4 System Implementation (Phase 3)

In this phase, the system's components are integrated and validated to ensure smooth operation and accurate performance. The software commands—predominantly written in Python—are executed to process the image data stored in the Hand Gesture Library. This involves reading images, extracting hand landmarks using Media Pipe, and compiling these

into a dataset formatted for Deep Learning classification. The Deep Learning classifier is then trained and tested using this dataset to identify and classify hand gestures effectively. Unlike systems that may utilize complex deep learning algorithms such as R-CNNs for motion detection, our system focuses on static gesture recognition, making it highly efficient and suitable for real-time applications. Python's extensive library ecosystem, including OpenCV for image processing and scikit-learn for implementing the Deep Learning, is utilized due to its robustness and ease of access, providing a comprehensive environment for both development and deployment, as shown in Figure 3.2.

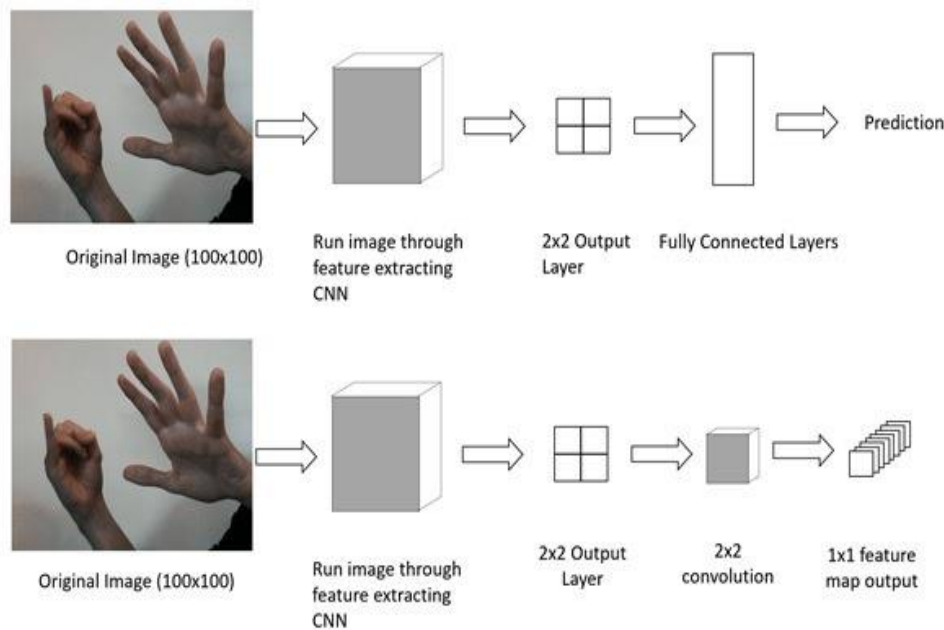


Figure 3.2 Examples of sign language and hand movement detection [17]

3.2.4 Performance Evaluation (phase 4)

The performance evaluation for the Sign Language Recognition research using Media Pipe and Deep Learning will assess the accuracy and efficiency of the model in classifying sign language gestures. This phase focuses on evaluating the Deep Learning model's capability to accurately interpret hand landmarks as gestures, measuring its computational speed, resource consumption, and overall usability. We utilize key classification metrics to gauge the effectiveness of our model, including Classification accuracy (CCR), True positive rate (TPR), True negative rate (TNR), False positive rate (FPR), and False negative rate (FNR). These metrics

provide a comprehensive understanding of the model's performance, ensuring it meets the practical requirements for real-time application and offers a reliable communication aid for the deaf and hard-of-hearing community, refer to Equation 3.1.

$$\begin{aligned}
 TPR &= \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN} \\
 FNR &= \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN} \\
 TNR &= \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP} \\
 FPR &= \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}
 \end{aligned}
 \dots\dots\dots(3.1)$$

3.3 Analysis of System Requirements

This section provides a comprehensive and detailed explanation of the software and hardware that have been selected for the research.

A. Hardware requirements

To run the program smoothly a hardware necessity is necessary. Below are the computer necessities for the developer. Table 3.1 indicates the type, minimal requirement, and advocated requirement.

Table 3.1: Hardware Requirements for Developers Computer

Developer Requirement	
Type	Recommended
Processor	Intel Core i7-1195G7 (11th Gen)
Cam Resolution	720P HD Recessed Camera
Memory (RAM)	16 GB RAM
Screen Resolution	1920 X 1080 (WUXGA)
Hard Disk Space	512 GB

B. Software requirements

Media Pipe is utilized as a real-time hand landmark detection framework in our research, effectively replacing the need for manual graphical image annotation tools such as Labellmg or Label Studio (Figure 3.3). These tools are typically used for

annotating images for deep learning algorithms. Instead, Media Pipe automatically extracts hand landmarks, which are directly used as input for our Deep Learning classifier. This streamlined approach is depicted in Figure 3.1. Model training involves the Deep Learning, which operates within the Python environment, bypassing the complex setup of deep neural network models that would otherwise require TensorFlow (Figure 3.4). This adjustment in methodology is highlighted in Figure 3.2 and Figure 3.3, which now represent is represented by the Python (Figure 3.5) programming environment and the integration of Deep Learning for classification, respectively. Additionally, OpenCV (Figure 3.6) is employed for initial image processing tasks, such as reading and preparing images before they are processed by Media Pipe, ensuring efficient and accurate detection of various sign language gestures.



Figure 3.3 Label Studio logo [18]



Figure 3.4 TensorFlow logo [19].



Figure 3.5 Python logo [20].



Figure 3.6 OpenCV logo [21].

3.4 Justification

The choice to use Deep Learning and Media Pipe for hand gesture recognition was based on their accuracy, efficiency, applicability to real-world scenarios, and consideration of limitations. Deep Learning ability to handle complex datasets and efficiently classify data points, along with Media Pipe's efficient hand landmark detection algorithms, make them suitable for tasks like hand gesture recognition. While they have limitations, such as Deep Learning struggling with large datasets and Media Pipe being affected by factors like lighting conditions, their proven effectiveness and ease of implementation ultimately led to their selection. Overall, Deep Learning and MediaPipe provide a strong foundation for developing a reliable and efficient system for recognizing hand gestures in applications like sign language translation.

3.5 Performance measurement

Performance measurement methodologies are essential for evaluating the effectiveness of the proposed solution. In assessing the performance of the Deep Learning model for hand gesture recognition, metrics such as accuracy, recall, and F1-score play a critical role. These metrics provide valuable insights into the model's ability to correctly classify hand gestures and its overall performance. Accuracy indicates the percentage of correct predictions made by the model. Recall evaluates the model's ability to correctly identify all positive instances, and the F1-score considers both precision and recall, providing a balanced assessment of the model's performance. By utilizing these metrics, we can effectively measure and evaluate the Deep Learning model's performance for hand gesture recognition.

3.6 Chapter Summary

This chapter provides an in-depth analysis of the methodology utilized to convert sign language into text language through the integration of computer vision and deep learning techniques. It encompasses the framework design, rationale behind the selected methodology, and outlines the methods employed for measuring performance effectively.

Chapter 4

RESEARCH DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter will explain the framework of the proposed intelligent Sign Language Translation based on Deep Learning. The framework includes several steps such as data collection, feature extraction, dataset preparation, feature selection, training and testing the Deep Learning model, as depicted in Figure 4.1.

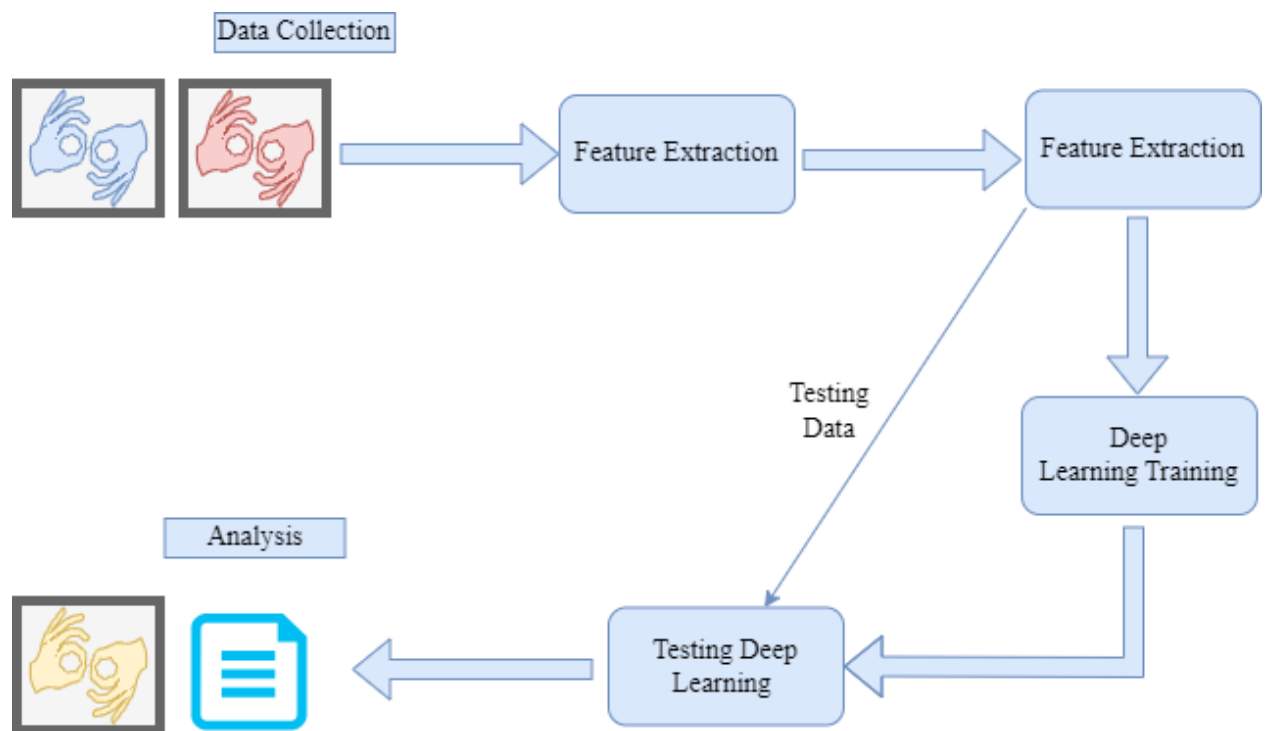


Figure 4. 1 : The framework the proposed intelligent Sign Language Translation based on Deep Learning

4.2 Proposed Solution

In this section delves into the outcomes of a proposed solution aimed at hand gesture recognition employing a fusion of Deep Learning and Media Pipe frameworks. The solution stands

out for its efficacy in real-time sign language recognition, as evidenced by systematic experimentation showcasing remarkable accuracy and low latency. We comprehensively explore various facets of the solution, including the implementation environment, dataset collection, feature extraction, Deep Learning model training, and testing methodologies. Furthermore, we provide insights into the key code files pivotal to the research's execution. By amalgamating Deep Learning and Media Pipe, the solution demonstrates significant promise in enhancing accessibility through real-time sign language translation technology.

4.3 Experiment Design

The experiment design encompassed dataset collection, feature extraction, model training, testing, and feature selection, aiming to develop an effective system for real-time sign language recognition.

4.3.1 Implementation Environment

To implement our research, we used various tools and programming language as follows:

4.3.1.1 Excel

Excel developed by Microsoft, is a part of the Microsoft Office suite of programs. It serves as a powerful tool for data manipulation, enabling users to process data, generate visualizations, and manage pivot tables effortlessly. Figure 4.2 illustrates the Excel program's interface, showcasing its capabilities in data analysis and visualization.

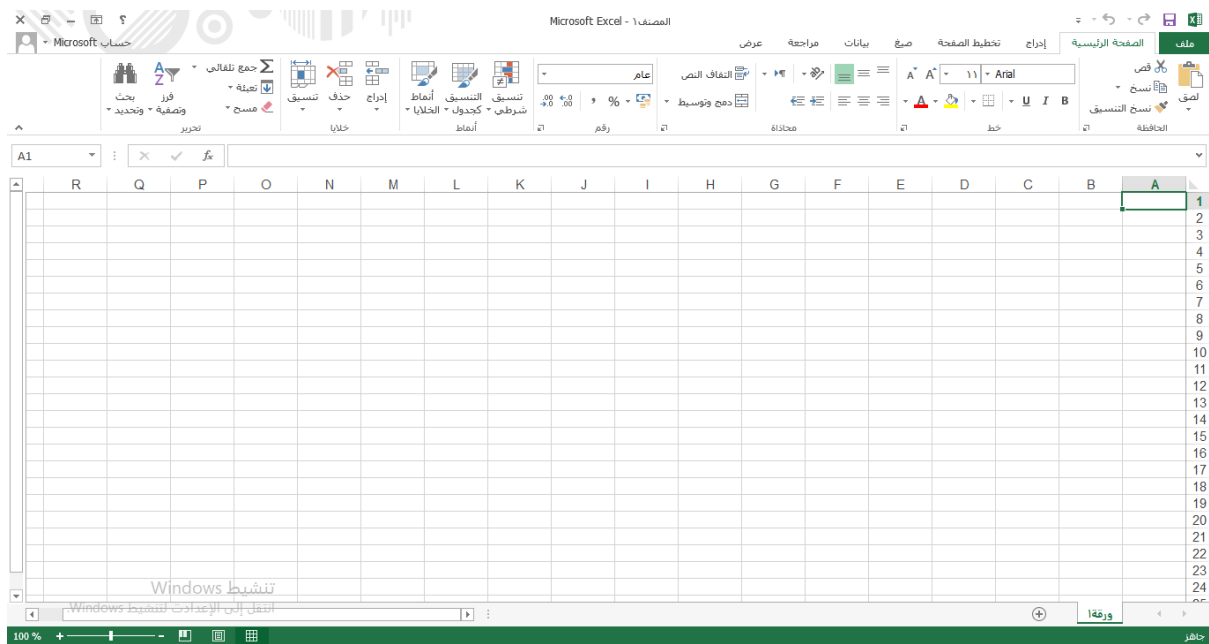


Figure 4. 2: The Excel program.

4.3.1.2 Visual studio code

Visual Studio Code is a lightweight, open-source code editor developed by Microsoft that offers a wide range of features, such as syntax highlighting, debugging tools, and extensions, to enhance the coding experience for developers. Figure 4.3 shows the visual studio code program.

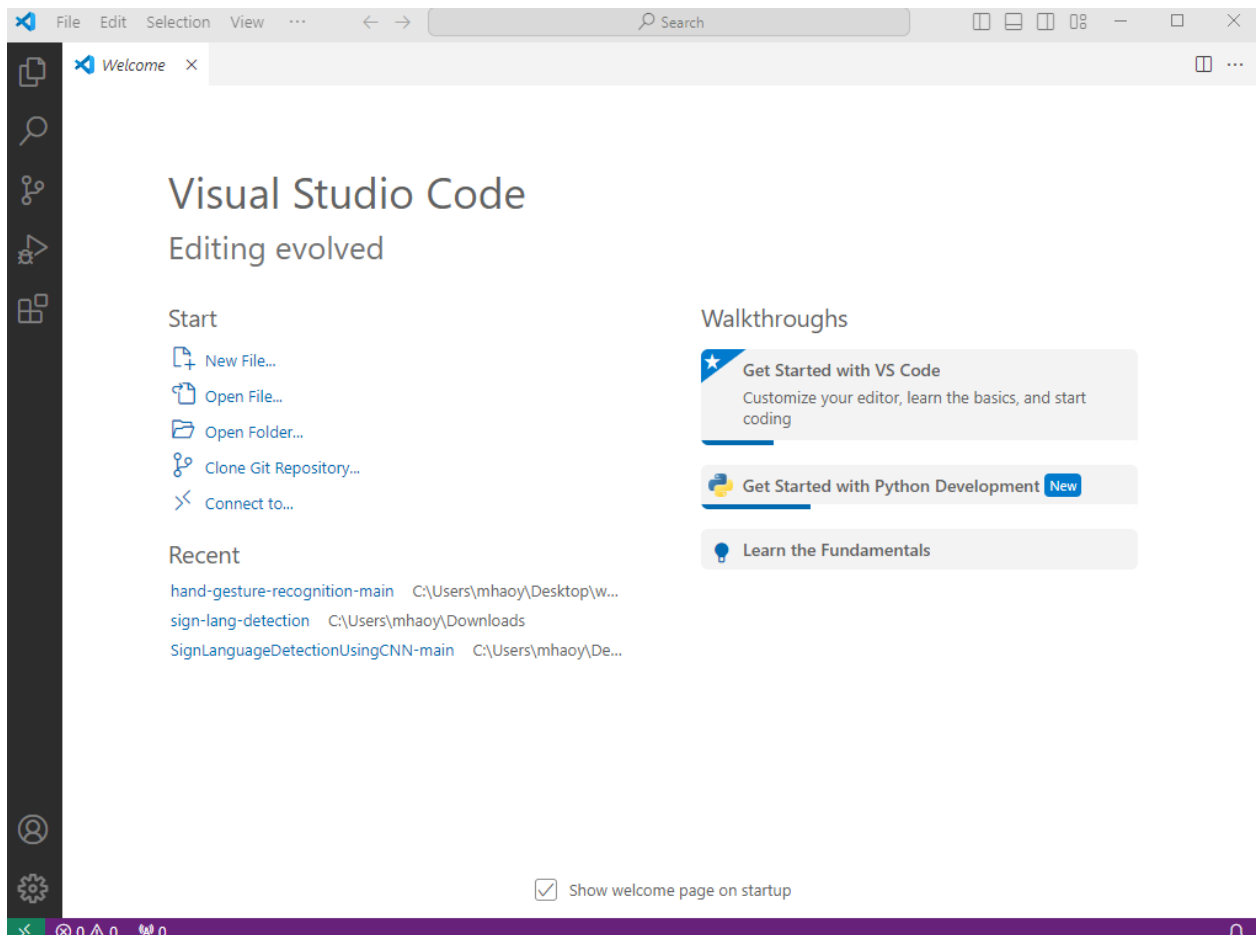


Figure 4. 3: visual studio code program.

4.3.1.3 Python

In our research, Visual Studio Code served as the distribution platform for Python programming. Python is an interpreted, object-oriented, high-level language known for its dynamic semantics. It is utilized in a wide range of applications including programming, software development, web development, machine learning, large-scale data processing, data science, and system scripting. Python is recognized as a popular and versatile language in the current programming landscape.

4.3.1.4 Jupyter Notebook

Jupyter Notebook in VS Code offers a versatile environment for interactive programming and data analysis with Python. Leveraging the power of the Python language, Jupyter Notebooks enable users to write and execute code in cells, visualize data, and document their workflows seamlessly. The integration with VS Code provides additional features like IntelliSense, debugging capabilities, and version control, enhancing the coding experience further. Overall, Jupyter Notebook in VS Code serves as an efficient tool for exploratory data analysis, prototyping machine learning models, and sharing insights through interactive narratives, refer to Figure 4.4.

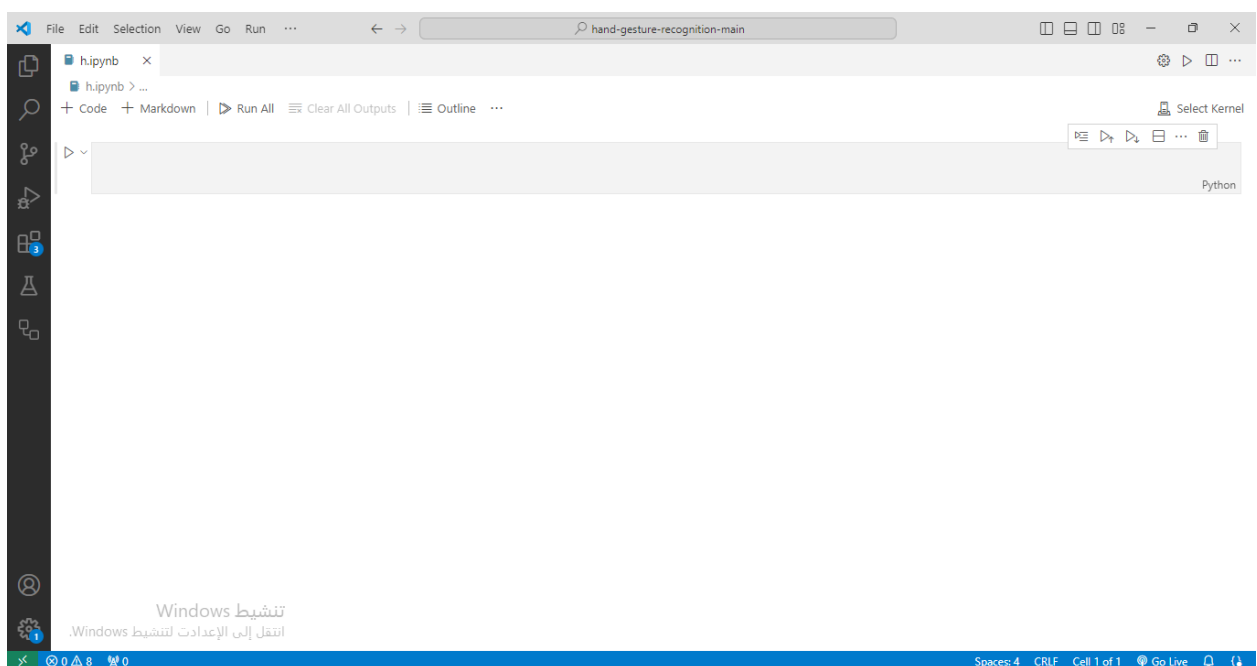


Figure 4. 4: Jupyter Notebook interface in vs code.

4.3.2. Dataset Collection

Data collection was done using online standard dataset, while offline data collection process for this research involved communicating with a webcam to capture images of hand gestures in real time. These images were then passed through the Media Pipe library to extract key landmarks on the hands, such as fingertips and palm points. Once these landmarks were identified, they were carefully processed and stored with appropriate labels indicating their classification within the gesture category. The process of creating the dataset went beyond simple feature extraction; It also involved organizing the data and pairing it with corresponding labels to accurately represent hand gestures. Figure 4.5 shows collected sign language dataset and Figure 4.6 shows the key landmarks of the hand.

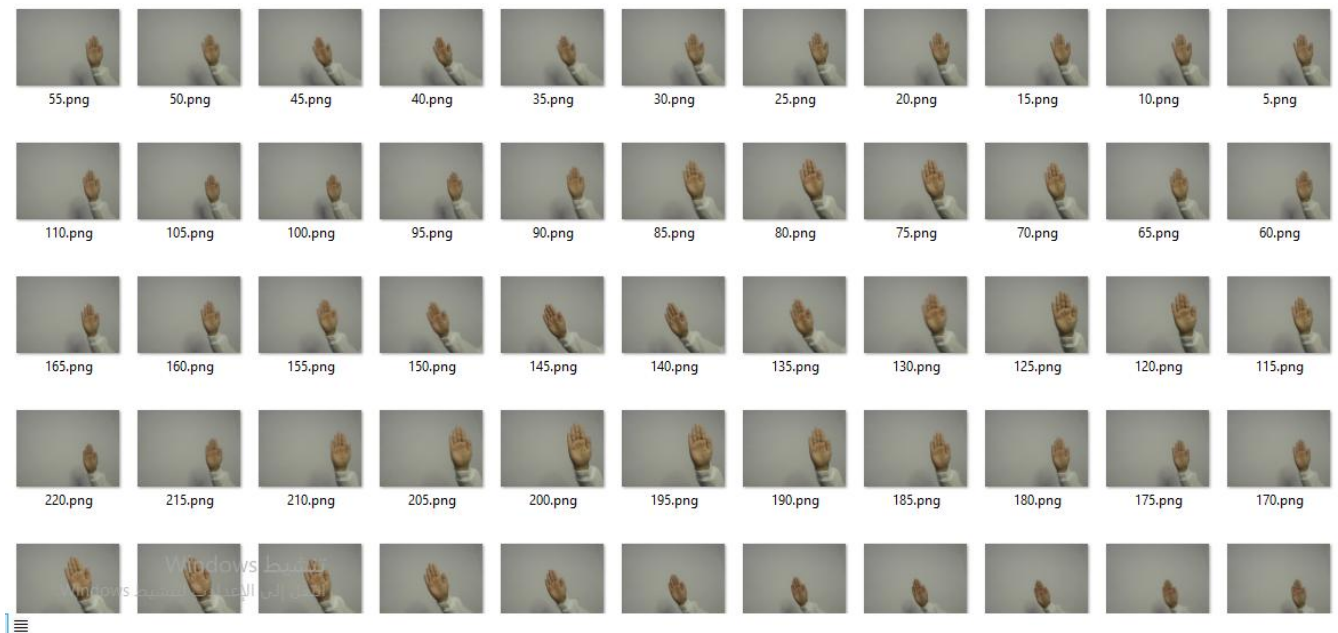


Figure 4. 5: The collected sign language dataset.

	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
1																	0.00000	1
2																	0.00000	2
3																	0.00000	3
4																	0.00000	4
5																	0.2097641	5
6																	0.00000	6
7																	0.00000	7
8																	0.2243505	8
9																	0.00000	9
10																	0.2443264	10
11																	0.2276299	11
12																	0.00000	12
13																	0.00000	13
14																	0.2294639	14
15																	0.00000	15
16																	0.00000	16
17																	0.00000	17
18																	0.2279992	18
19																	0.00000	19
20																	0.00000	20
21																	0.00000	21
22																	0.00000	22
23																	0.00000	23
24																	0.00000	24
25																		alright

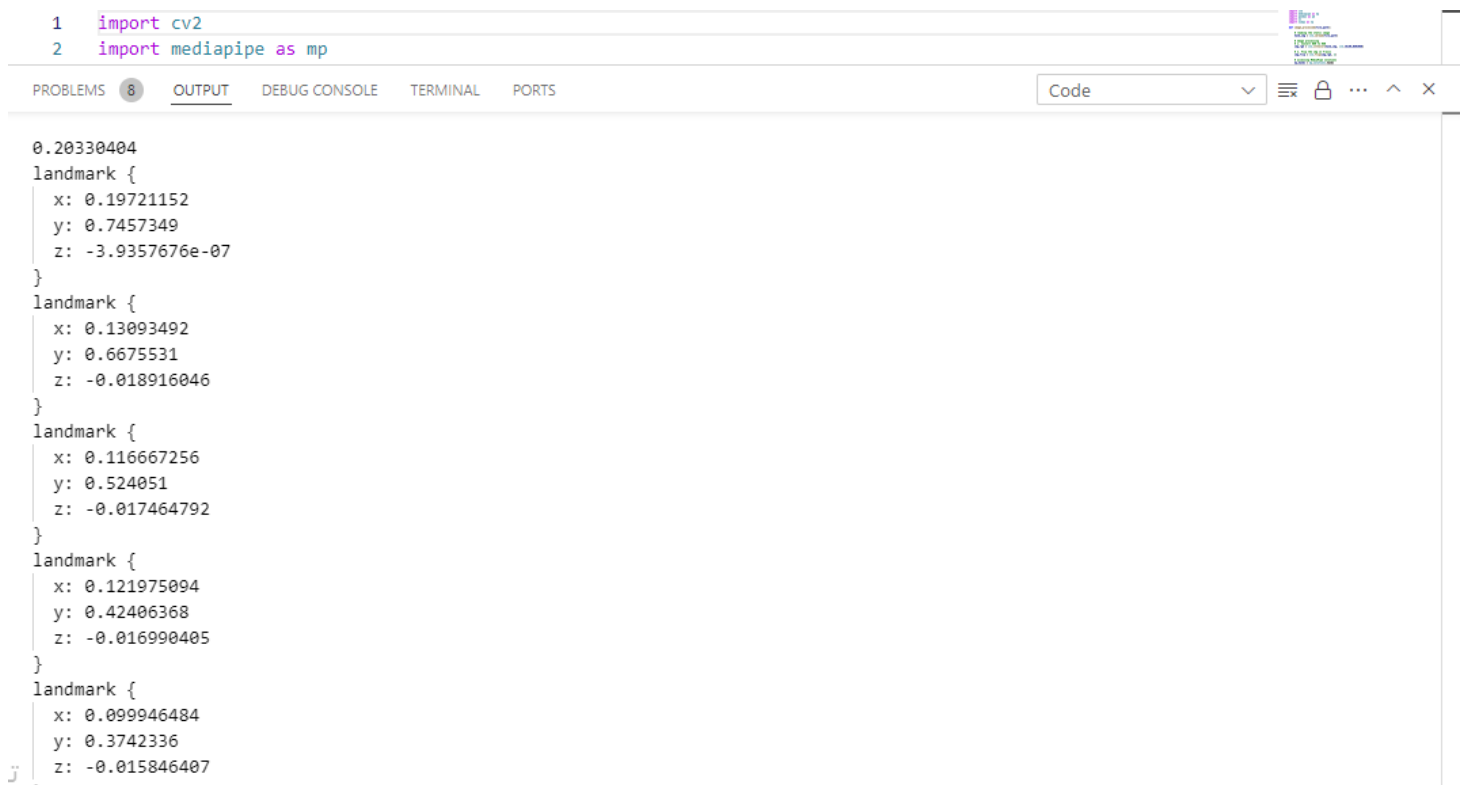
Figure 4. 6 : Sample of the training dataset with extract key landmarks on the hands

4.3.3 Features Extraction

Feature extraction involves isolating the hand landmark data from the dataset to serve as input features for training the Deep Learning classifier. This process entails loading the dataset, renaming columns, and selecting all columns except the last one, which contains the labels, to extract the features. By separating the hand landmark data as features, the code prepares the dataset for training the classifier to recognize hand gestures effectively.

4.3.4 Preparation of Training Dataset

In the dataset preparation phase, the relevant features of sign language are converted to numerical values to become more suitable with machine learning techniques. The sign language dataset used in this research is prepared to numerical values, as illustrated in Figure 4.7.



```

1 import cv2
2 import mediapipe as mp

0.20330404
landmark {
  x: 0.19721152
  y: 0.7457349
  z: -3.9357676e-07
}
landmark {
  x: 0.13093492
  y: 0.6675531
  z: -0.018916046
}
landmark {
  x: 0.116667256
  y: 0.524051
  z: -0.017464792
}
landmark {
  x: 0.121975094
  y: 0.42406368
  z: -0.016990405
}
landmark {
  x: 0.099946484
  y: 0.3742336
  z: -0.015846407
}

```

Figure 4. 7: Training dataset on Python.

4.3.4 Training of Deep Learning model

The flowchart in Figure 4.8 illustrates the sequential process of capturing, processing, and evaluating hand gesture data for sign language recognition. Real-time video data of hand gestures is captured using a camera, initiating the recognition process. Media Pipe is employed to extract hand landmarks from the video frames, capturing crucial spatial positions and orientations of hand elements. The extracted landmarks are then organized into a structured format suitable for training and testing the model, ensuring data integrity and efficiency. Subsequently, the structured data is utilized to train the Deep Learning model, focusing on recognizing patterns in hand landmarks corresponding to specific sign language gestures. The trained Deep Learning model is applied to a separate set of hand gesture data to evaluate its accuracy and effectiveness in recognizing sign language gestures. The model predicts the sign language label for new hand gestures based on the learned patterns, facilitating real-time communication. Finally, the model's performance is assessed using metrics such as accuracy, precision, recall, and F1-score, ensuring its reliability and effectiveness in practical applications.

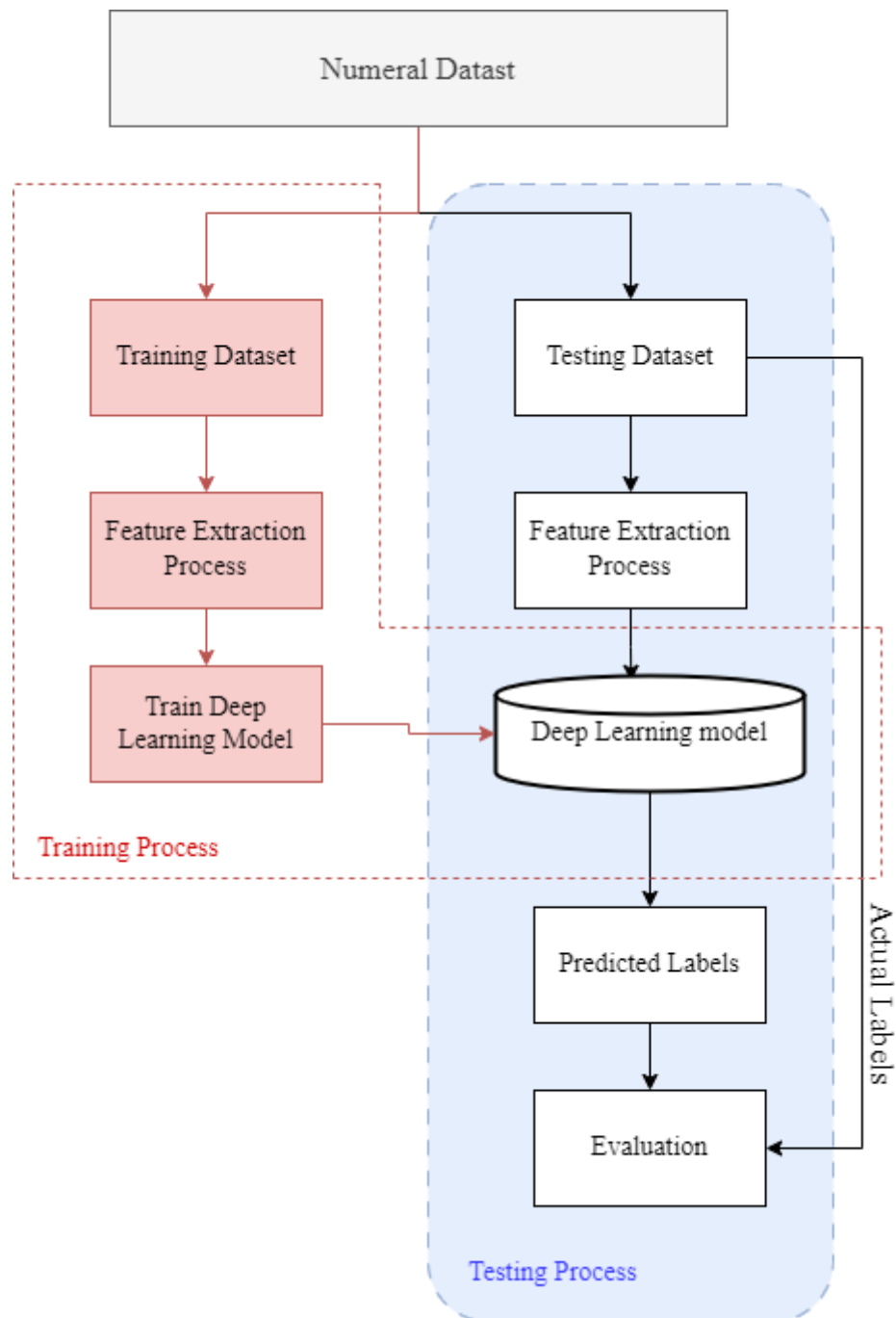


Figure 4. 8: Example of the architecture of Deep learning

4.3.5 Testing of Deep Learning model

Testing the suggested Deep Learning model for sign language translation is crucial for evaluating their effectiveness. In this approach, the dataset is randomly split into training data (80%) and testing data (20%). This division is essential for assessing the performance of the

Deep Learning model in accurately identifying sign language translation. The dataset was uploaded in Python using command `read_csv`. Figure 4.9 shows the training dataset on Python.

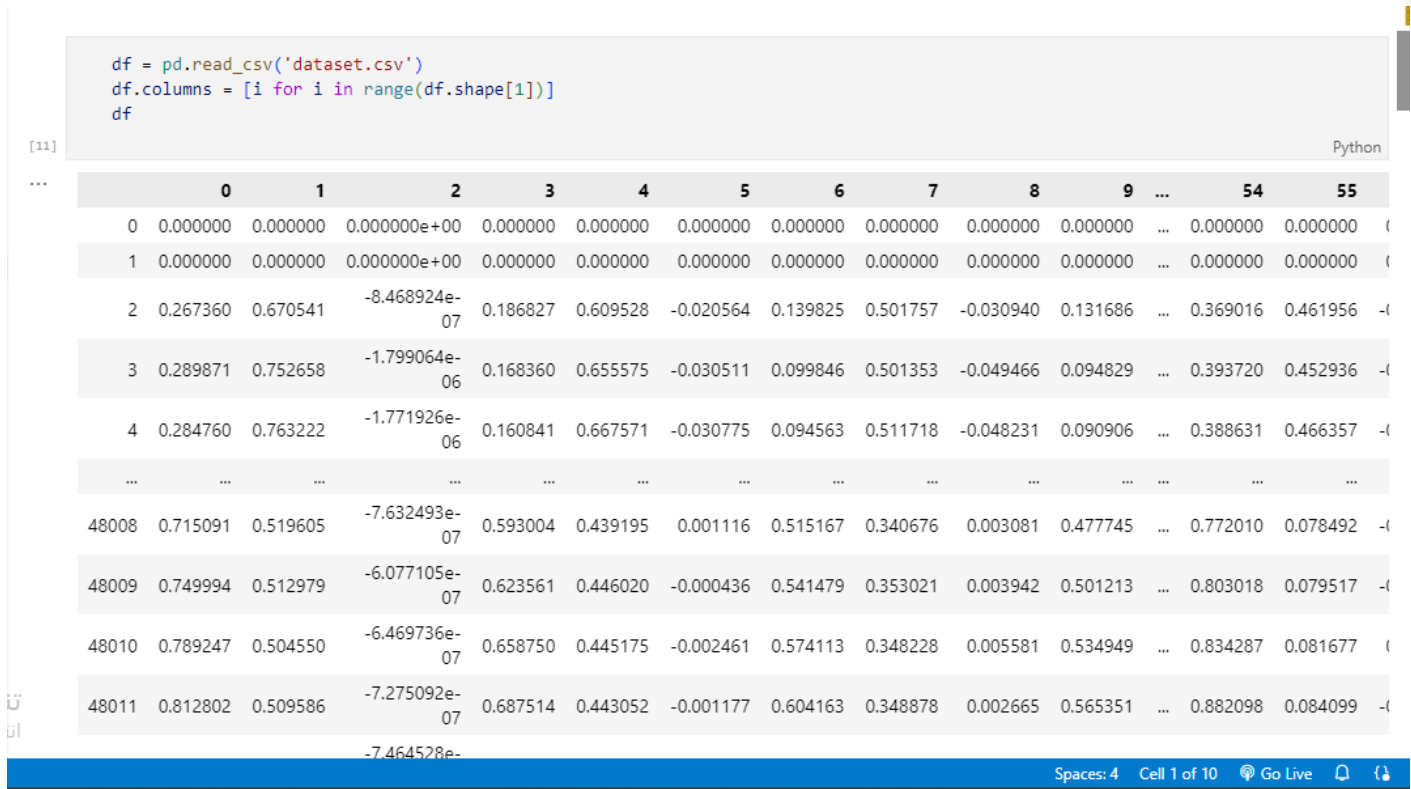


Figure 4. 9: uploaded the dataset

Then, the dataset was divided into training (80%) and testing datasets (20%) using the `train_test_split` function as follows:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

```
Deep Learning = SVC(C=10, gamma=0.1, kernel='rbf')
```

```
Deep Learning.fit(x_train, y_train)
```

4.3.Feature Selection Using Deep Learning model

The steps of feature selection using landmark by Deep Learning can be explained in the following:

- 1) After we train the dataset, we can use the model to translate sign language
- 2) Open the file named " hands_gesture_recog.py" ,as shown in Figure 4.10.

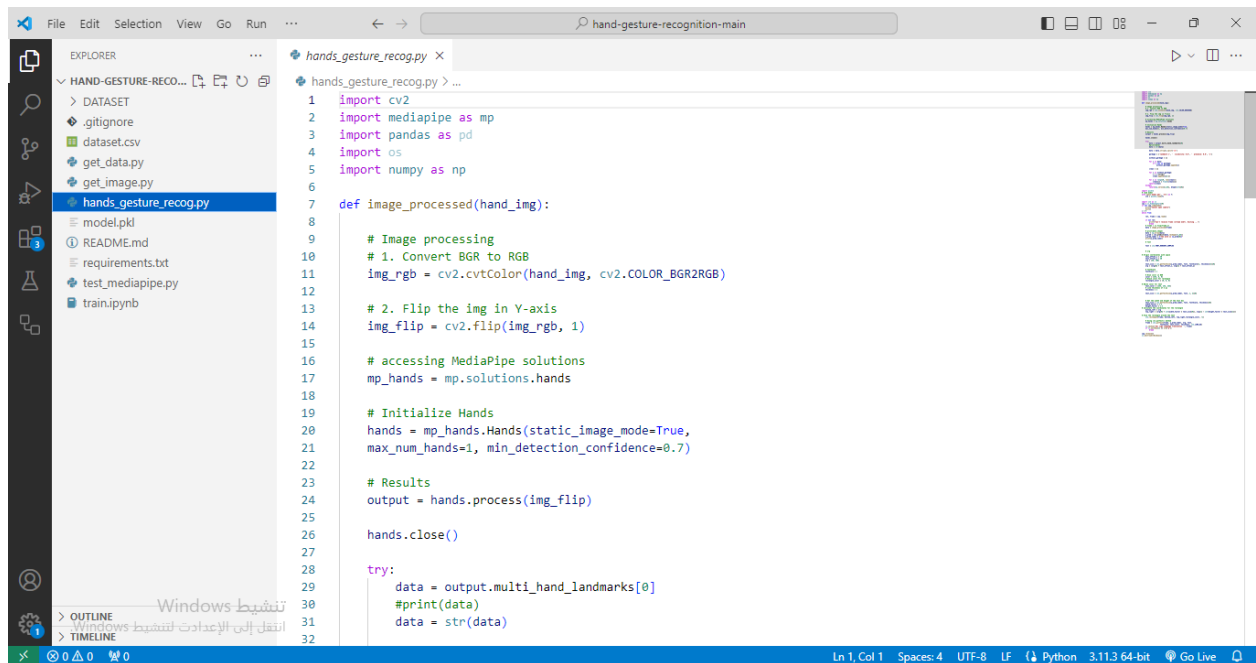


Figure 4. 10: Feature Selection Using Deep Learning model

- 3) After opening the file run the file to open the sign language translation, and the result will appear in " ASL sign language translation " frame as shown in Figure 4.11.

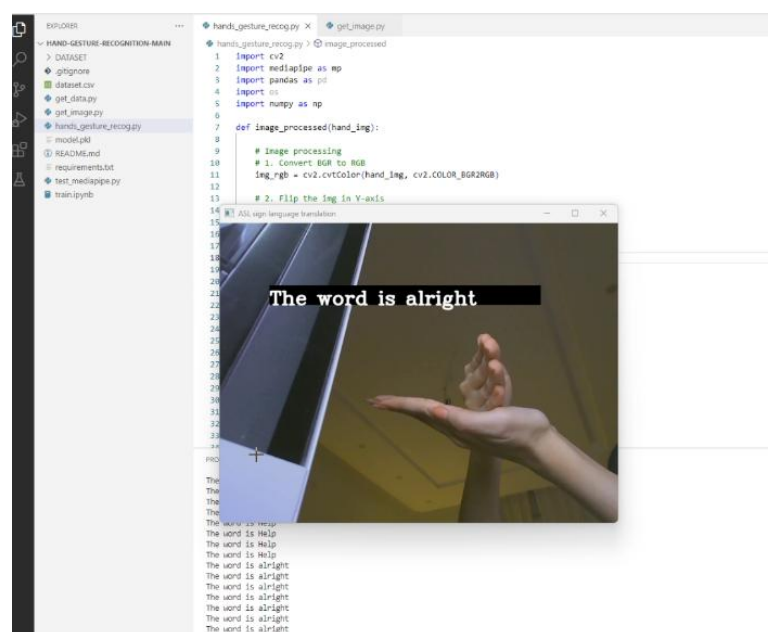
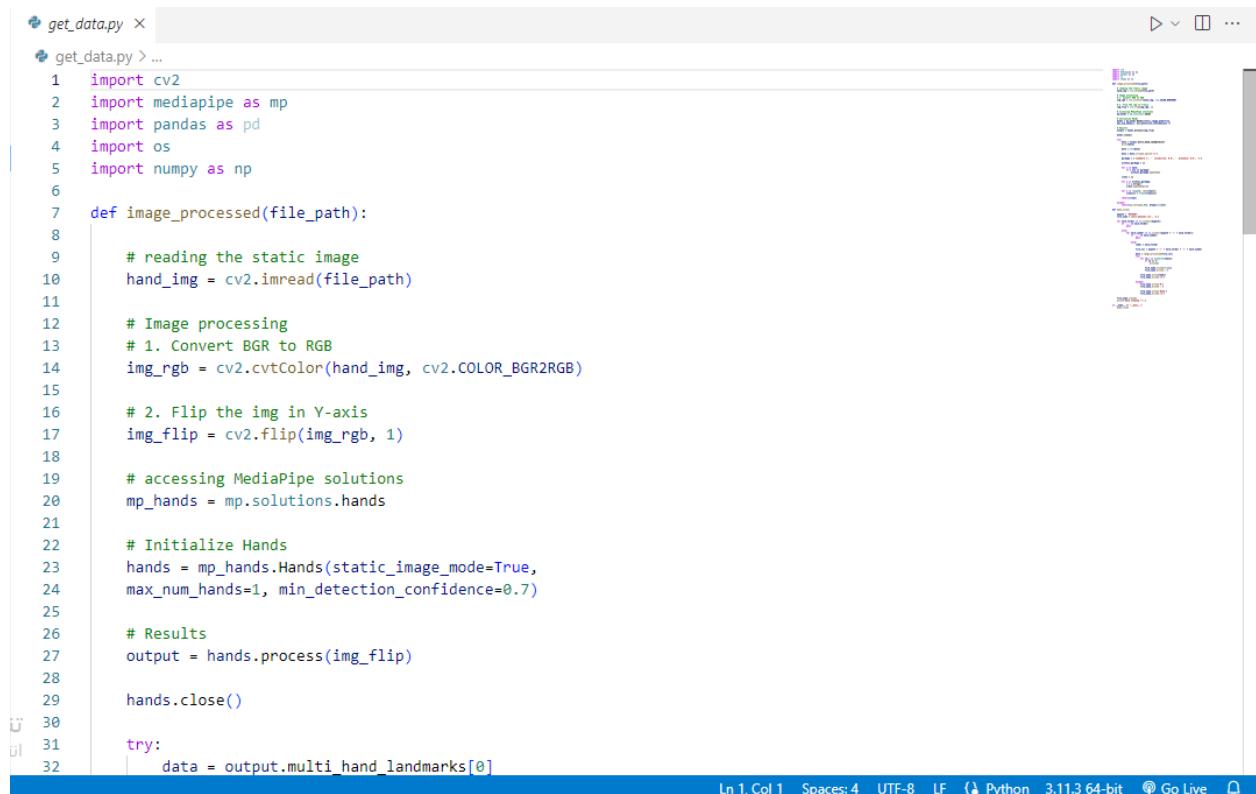


Figure 4. 11: ASL sign language translation frame.

4.4 Parameter and Testing Method

4.4.1 Feature Selection

Feature extraction is a crucial step in machine learning, particularly in tasks like image processing and computer vision. In the context of hand gesture recognition, feature extraction involves identifying and extracting relevant characteristics or patterns from the input data, which, in this case, are images containing hand landmarks. These features serve as the input to machine learning algorithms, enabling them to learn patterns and make predictions. Feature extraction is performed on images containing hand landmarks detected using the Media Pipe library. Specifically, the `image_processed` function reads each static image, converts it to RGB format, and processes it to identify hand landmarks. These landmarks are extracted using the Media Pipe hands module, and the resulting data is cleaned and transformed into a format suitable for further analysis. This extracted feature data, consisting of the coordinates of hand landmarks, forms the basis for training the Deep Learning model, as described in Figure 4.12, and 4.13.



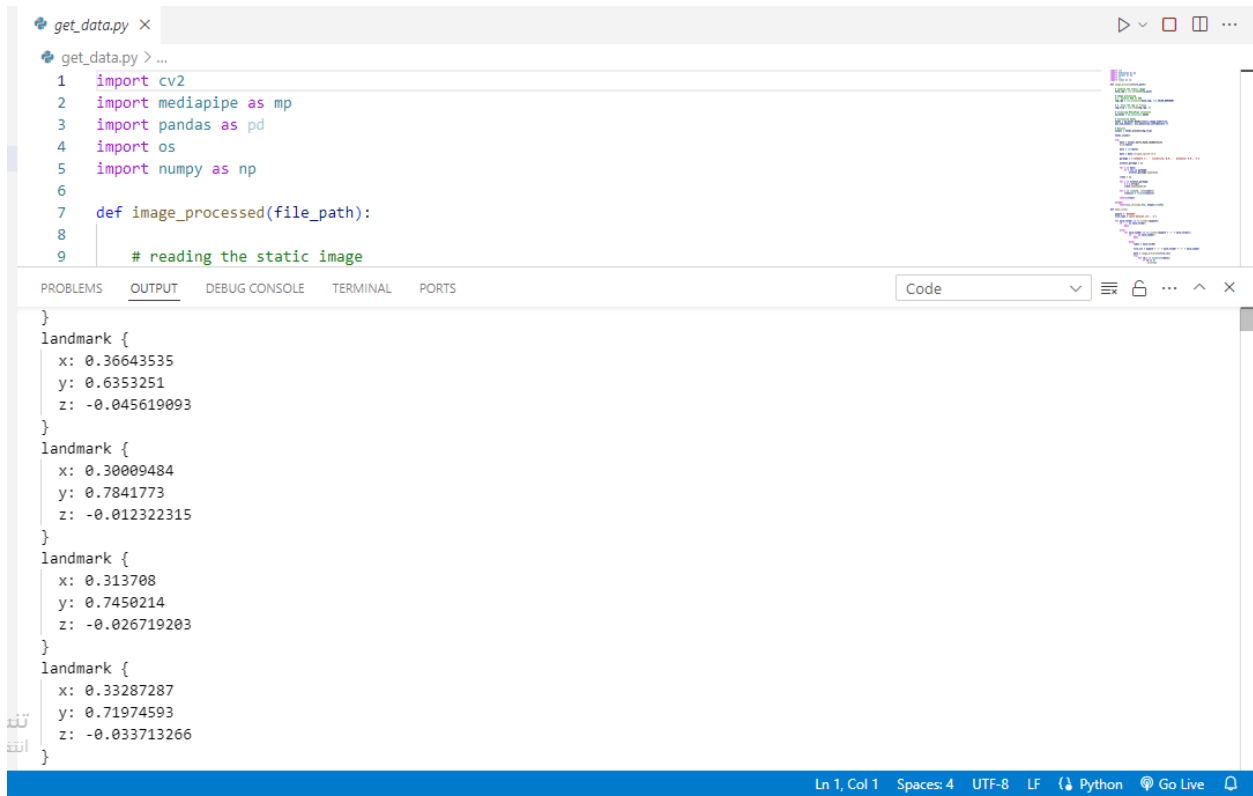
```

get_data.py x
get_data.py > ...
1  import cv2
2  import mediapipe as mp
3  import pandas as pd
4  import os
5  import numpy as np
6
7  def image_processed(file_path):
8
9      # reading the static image
10     hand_img = cv2.imread(file_path)
11
12     # Image processing
13     # 1. Convert BGR to RGB
14     img_rgb = cv2.cvtColor(hand_img, cv2.COLOR_BGR2RGB)
15
16     # 2. Flip the img in Y-axis
17     img_flip = cv2.flip(img_rgb, 1)
18
19     # accessing MediaPipe solutions
20     mp_hands = mp.solutions.hands
21
22     # Initialize Hands
23     hands = mp_hands.Hands(static_image_mode=True,
24                             max_num_hands=1, min_detection_confidence=0.7)
25
26     # Results
27     output = hands.process(img_flip)
28
29     hands.close()
30
31     try:
32         data = output.multi_hand_landmarks[0]

```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.11.3 64-bit Go Live

Figure 4. 12: Image processing function



```

get_data.py > ...
1 import cv2
2 import mediapipe as mp
3 import pandas as pd
4 import os
5 import numpy as np
6
7 def image_processed(file_path):
8
9     # reading the static image

```

```

}
landmark {
  x: 0.36643535
  y: 0.6353251
  z: -0.045619093
}
landmark {
  x: 0.30009484
  y: 0.7841773
  z: -0.012322315
}
landmark {
  x: 0.313708
  y: 0.7450214
  z: -0.026719203
}
landmark {
  x: 0.33287287
  y: 0.71974593
  z: -0.033713266
}

```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python Go Live

Figure 4. 13 : Data extraction to identify hand landmarks

4.4.2 Model Architecture

The Deep Learning model architecture is employed for hand gesture recognition tasks, where it learns to classify complex patterns in image data by creating a decision boundary to separate different classes of hand gestures in a high-dimensional feature space. During training, Deep Learning optimizes its parameters to maximize the margin between classes, aiming for the highest classification accuracy. Once trained, the Deep Learning can predict the class labels of new hand gesture images, enabling real-time recognition for applications like sign language translation and human-computer interaction.

4.4.3 Training Hyperparameters

Training hyperparameters are pivotal components in the process of initializing and training a Deep Learning classifier. These parameters significantly influence the model's performance and its ability to generalize to unseen data. In this context, we begin by importing necessary libraries such as `train_test_split` for data splitting, `SVC` for Deep Learning classifier initialization, and metrics functions like `confusion_matrix`, `f1_score`, `recall_score`, and `precision_score` for evaluation. Upon reading the dataset using `pd.read_csv`, we ensure numerical column indices by assigning sequential indices to DataFrame columns as showing

in Figure 4.14. Additionally, we rename a specific column to 'Output' for clarity and consistency throughout the training process as shown in Figure 4.15. These preparatory steps set the stage for subsequent feature extraction, model training, and evaluation phases, ensuring a systematic and robust approach to Deep Learning model development.

The screenshot shows a Jupyter Notebook with two code cells. The first cell imports necessary libraries: pandas, sklearn, seaborn, matplotlib, and numpy. The second cell loads a CSV file named 'dataset.csv' and sets the column indices from 0 to 55.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, f1_score, recall_score, precision_score
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

```
df = pd.read_csv('dataset.csv')
df.columns = [i for i in range(df.shape[1])]
df
```

The output of the second cell shows a DataFrame with 56 columns (indices 0 to 55). The first few rows of data are displayed, showing numerical values for each column.

Figure 4. 14: sequential indices to DataFrame columns

The screenshot shows a Jupyter Notebook with one code cell. The cell renames the column at index 63 to 'Output'.

```
df = df.rename(columns={63: 'Output'})
df
```

The output of the cell shows the same DataFrame as before, but with the column at index 63 now labeled 'Output'. The data values remain the same.

Figure 4. 15 : training process

4.4.4 Testing Methods

Testing methods are pivotal in assessing the efficacy and reliability of machine learning models, ensuring their aptness for real-world deployment. In the context of the provided Deep Learning model for sign language translation, testing involves gauging the model's predictive prowess and quantifying its accuracy, precision, recall, and F1 score. The process commences with the segregation of data into training and testing sets, the former for model training and the latter for performance evaluation. Following model training, predictions are made on the unseen testing data, which are juxtaposed against actual labels to compute diverse evaluation metrics. The generated confusion matrix offers a comprehensive snapshot of the model's performance across various classes, delineating true positives, true negatives, false positives, and false negatives. Furthermore, metrics such as F1 score, recall, and precision provide nuanced insights into the model's accuracy, sensitivity, and specificity. For the Deep Learning model in question, the computed F1 score of 0.96, along with recall and precision metrics of the same value, underscore its robustness in accurately discerning sign language gestures, thereby affirming its reliability in real-world scenarios. Figure 4.16 shows the result of testing.

```

train.ipynb
train.ipynb > ...
+ Code + Markdown | ▶ Run All ⏹ Restart ⚙ Clear All Outputs | 📄 Variables 📄 Outline ... Python 3.11.3
▶ print(cf_matrix)
print("\nEvaluation Metrics:")
print("F1 Score:", f1)
print("Recall:", recall)
print("Precision:", precision)
# Calculate and print accuracy
accuracy = (np.sum(np.diag(cf_matrix))) / np.sum(cf_matrix)
print("Accuracy:", accuracy)

[18] ✓ 0.0s Python

... Confusion Matrix:
[[62  0  0  0  0  0  0  0  0  0]
 [ 0 53  4  0  0  0  0  0  0  0]
 [ 0  1 58  0  0  0  0  0  0  0]
 [ 0  0  0 54  0  0  0  0  0  0]
 [ 0  0  0  0 58  0  0  0  0  0]
 [ 0  0  0  0  0 58  0  0  0  0]
 [ 0  0 12  0  0  0 49  0  0  0]
 [ 0  0  0  0  0  0  0 68  0  0]
 [ 0  0  0  0  0  0  0  0 68  0]
 [ 0  0  3  0  0  0  0  0  0 52]]

Evaluation Metrics:
F1 Score: 0.9666666666666667
Recall: 0.9666666666666667
Precision: 0.9666666666666667
Accuracy: 0.9666666666666667

```

Figure 4. 16 : The result of testing

4.5 Chapter Summary

This chapter detailed the framework of the proposed intelligent Sign Language Translation system based on Deep Learning Implementation Environment. The framework encompasses several key steps, such as data collection, feature extraction, dataset preparation, feature selection, as well as the training and testing of the Convolutional Neural Network (CNN) model.

Chapter 5

RESULTS, ANALYSIS AND DISCUSSION

5.1 Introduction

This chapter will explore the experimental and coding facets involved in training a Deep Learning to enhance hand gesture recognition in American Sign Language (ASL). It introduces key code files essential for Deep Learning training, provides explanations on their significance, and offers discussion on research results and analysis.

5.2 Experiment/Coding

The code of training Deep Learning for ASL sign language consists of five files: Get_image File, Get_data File, Test_mediapipe File, Train File and Main File.

5.2.1 Get_image File.

This code for "get_image" serves to capture images from a webcam in real-time and store them for further use, for task dataset creation or image processing. It first sets up the environment by creating a directory structure to store the images. Then, it initializes the webcam and begins capturing frames. These frames are periodically saved as images in the specified directory. The process continues until a predefined condition is met, such as capturing a certain number of images or the user interrupting the process. Finally, the webcam is released, and any OpenCV windows are closed. This functionality facilitates tasks requiring real-time image acquisition, such as building datasets for machine learning applications or conducting computer vision experiments, refer to Code in Figure 5.1.

```

import numpy as np
import cv2 as cv
from pathlib import Path

def get_image():
    Class = 'okay'
    Path('DATASET/'+Class).mkdir(parents=True, exist_ok=True)
    cap = cv.VideoCapture(0)
    if not cap.isOpened():
        print("Cannot open camera")
        exit()
    i = 0
    while True:

        ret, frame = cap.read()

        if not ret:
            print("Can't receive frame (stream end?). Exiting ...")
            break
        # frame = cv.flip(frame,1)
        i+= 1
        if i % 5==0:
            cv.imwrite('DATASET/'+Class+'/'+str(i)+'.png',frame)

        cv.imshow('frame', frame)
        if cv.waitKey(1) == ord('q') or i > 500:
            break

    cap.release()
    cv.destroyAllWindows()
if __name__ == "__main__":
    get_image()

```

Figure 5. 1:Python code for capturing the image from Camera.

5.2.2 Get_data File.

This Python script utilizes the Media Pipe library for hand landmark detection on static images and subsequently compiles the extracted landmark data into a CSV file. The `image_processed` function reads each static image, converts it to RGB format, and processes it to identify hand landmarks using the Media Pipe hands module. It then extracts and cleans the

landmark data, returning it as a list of coordinates. The `make_csv` function traverses a directory structure containing image files categorized by hand gestures. For each image, it invokes `image_processed` to obtain the landmark data and appends it, along with the corresponding label derived from the folder name to a CSV file named "dataset.csv". When executed as the main program, this script automates the extraction and organization of hand landmark data, facilitating its subsequent analysis or utilization for machine learning model training. refer to Code in Figure 5.2.

```
import cv2
import mediapipe as mp
import os
import numpy as np
def image_processed(file_path):
    """Process static image to extract hand landmarks."""
    hand_img = cv2.imread(file_path)
    img_rgb = cv2.cvtColor(hand_img, cv2.COLOR_BGR2RGB)
    img_flip = cv2.flip(img_rgb, 1)
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(static_image_mode=True, max_num_hands=1, min_detection_confidence=0.7)
    output = hands.process(img_flip)
    hands.close()
    try:
        data = output.multi_hand_landmarks[0]
        data = str(data).strip().split('\n')
        garbage = ['landmark {', ' visibility: 0.0', ' presence: 0.0', '}']
        without_garbage = [i for i in data if i not in garbage]
        clean = [float(i.strip()[2:]) for i in without_garbage]
        return clean
    except:
        return np.zeros([1,63], dtype=int)[0]
def make_csv():
    """Traverse image directory, process images, and compile landmark data into a CSV file."""
    mypath = 'DATASET'
    file_name = open('dataset.csv', 'a')

    for each_folder in os.listdir(mypath):
        if '.' in each_folder:
            pass
        else:
            for each_number in os.listdir(mypath + '/' + each_folder):
                if '.' in each_number:
                    pass
                else:
                    label = each_folder
                    file_loc = mypath + '/' + each_folder + '/' + each_number
                    data = image_processed(file_loc)
                    try:
                        for id,i in enumerate(data):
                            if id == 0:
                                print(i)
                                file_name.write(str(i))
                                file_name.write(',')
                            file_name.write(label)
                            file_name.write('\n')
                    except:
                        file_name.write('0')
                        file_name.write(',')
                        file_name.write('None')
                        file_name.write('\n')
    file_name.close()
    print("Data Created !!!")
```

```
if __name__ == "__main__":
    make_csv()
```

Figure 5. 2:MediaPipe automates hand landmark detection on images.

5.2.3 Test_mediapipe File

This code utilizes the Media Pipe library alongside OpenCV to detect and visualize hand landmarks in real-time from a webcam feed. After initializing the webcam input, it configures the hand detection model with specific parameters. Within the main loop, frames from the webcam are continuously processed to detect hand landmarks using the configured model. Detected hand landmarks are then annotated with connecting lines and displayed on the frame. The program runs until the user presses the 'Esc' key, upon which the webcam feed is released, and the program terminates. Essentially, it provides a real-time visualization of hand landmarks detected from the webcam input. refer to Code in Figure 5.3.

```
import cv2
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands
# For webcam input:
cap = cv2.VideoCapture(0)
with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue

        # To improve performance, optionally mark the image as not writeable to
        # pass by reference.
        image.flags.writeable = False
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = hands.process(image)

        # Draw the hand annotations on the image.
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    image,
                    hand_landmarks,
                    mp_hands.HAND_CONNECTIONS,
                    mp_drawing_styles.get_default_hand_landmarks_style(),
                    mp_drawing_styles.get_default_hand_connections_style())
        # Flip the image horizontally for a selfie-view display.
        cv2.imshow('MediaPipe Hands', cv2.flip(image, 1))
        if cv2.waitKey(5) & 0xFF == 27:
            break
    cap.release()
```

Figure 5. 3: Real-time visualization of hand landmarks detected from a webcam feed using Media Pipe and OpenCV.

5.2.4 Train File

The steps of training Deep Learning model for translation sign language can be explained in the following:

- 1) This code reads a CSV file into a Data Frame using pandas and resets the column names to numerical indices.

```
df = pd.read_csv('dataset.csv')
df.columns = [i for i in range(df.shape[1])]
df
```

- 2) This code renames a column in the DataFrame 'df' to 'Output'

```
df = df.rename(columns={63: 'Output'})
df
```

- 3) Feature extraction

```
X = df.iloc[:, :-1]
print("Features shape =", X.shape)

Y = df.iloc[:, -1]
print("Labels shape =", Y.shape)
```

- 4) This code segment splits the dataset into training and testing sets, initializes a Deep Learning classifier with specified hyperparameters, and trains the classifier on the training data.

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
Deep Learning = SVC(C=10, gamma=0.1, kernel='rbf')
Deep Learning.fit(x_train, y_train)
```

- 5) Prediction output

```
y_pred = Deep Learning.predict(x_test)
```

```
y_pred
```

6) Evaluation metrics

```
cf_matrix = confusion_matrix(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='micro')
recall = recall_score(y_test, y_pred, average='micro')
precision = precision_score(y_test, y_pred, average='micro')
f1, recall, precision
```

7) Print Testing Results

```
print("Confusion Matrix:")
print(cf_matrix)
print("\nEvaluation Metrics:")
print("F1 Score:", f1)
print("Recall:", recall)
print("Precision:", precision)
# Calculate and print accuracy
accuracy = (np.sum(np.diag(cf_matrix))) / np.sum(cf_matrix)
print("Accuracy:", accuracy)
```

- 8) This code saves a machine learning model (likely a Deep Learning model) to a file named 'model.pkl' using Python's pickle module.

```
import pickle

# save model
with open('model.pkl','wb') as f:
    pickle.dump(Deep Learning,f)
```

5.2.5 Main File

This code file performs real-time hand gesture recognition using a trained Deep Learning model. It captures video frames from a webcam, processes them to extract hand landmarks using Media Pipe, and then uses a trained Deep Learning model to predict the gesture represented by the hand landmarks. The predicted gesture label is overlaid on the video feed and displayed in a window. refer to Code in Figure 5.4.

```
import cv2
import mediapipe as mp
import os
```

```

import numpy as np
import pickle

def image_processed(hand_img):
    """Process static image to extract hand landmarks."""
    img_rgb = cv2.cvtColor(hand_img, cv2.COLOR_BGR2RGB)
    img_flip = cv2.flip(img_rgb, 1)
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(static_image_mode=True, max_num_hands=1, min_detection_confidence=0.7)
    output = hands.process(img_flip)
    hands.close()

    try:
        data = output.multi_hand_landmarks[0]
        data = str(data).strip().split('\n')
        garbage = ['landmark {' , ' visibility: 0.0', ' presence: 0.0', '}']
        without_garbage = [i for i in data if i not in garbage]
        clean = [float(i.strip()[2:]) for i in without_garbage]
        return clean
    except:
        return np.zeros([1,63], dtype=int)[0]

# Load model
with open('model.pkl', 'rb') as f:
    DeepLearning = pickle.load(f)

cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()

while True:
    ret, frame = cap.read()

    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    data = image_processed(frame)
    data = np.array(data)
    y_pred = DeepLearning.predict(data.reshape(-1,63))
    y_pred_label = f"The word is {y_pred[0]}"

    font = cv2.FONT_HERSHEY_COMPLEX
    text_offset_x = 30
    text_offset_y = 30
    org = (50, 100)
    org = (org[0] + text_offset_x, org[1] + text_offset_y)
    fontScale = 1
    color = (255, 0, 0)
    rectangle_color = (0, 0, 0)
    text_color = (255, 255, 255)
    thickness = 2

    text_size = cv2.getTextSize(y_pred_label, font, fontScale, thickness)[0]
    width_factor = 1.3
    height_factor = 1.4
    bottom_left = org
    top_right = (org[0] + int(width_factor * text_size[0]), org[1] - int(height_factor * text_size[1]))

    cv2.rectangle(frame, bottom_left, top_right, rectangle_color, -1)
    frame = cv2.putText(frame, y_pred_label, org, font, fontScale, text_color, thickness, cv2.LINE_AA)
    cv2.imshow('ASL sign language translation', frame)

    if cv2.waitKey(1) == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

```

Figure 5. 4:Real-time ASL sign language translation using deep learning model.

5.3 Research Results and Analysis

The research results and analysis for the proposed solution in hand gesture recognition using Deep Learning and Media Pipe reveal promising outcomes. By adopting a streamlined approach that integrates Media Pipe for efficient hand landmark extraction and Deep Learning for gesture classification, the system achieves high accuracy and low latency, making it well-suited for real-time applications like sign language recognition. Through quantitative metrics such as accuracy, precision, recall, and F1-score, as well as qualitative visualizations, the system's performance is thoroughly evaluated, showcasing its robustness in accurately detecting and translating sign language gestures. While the proposed solution demonstrates considerable effectiveness. Figure 5.5-5.14 show the result of ASL translation.

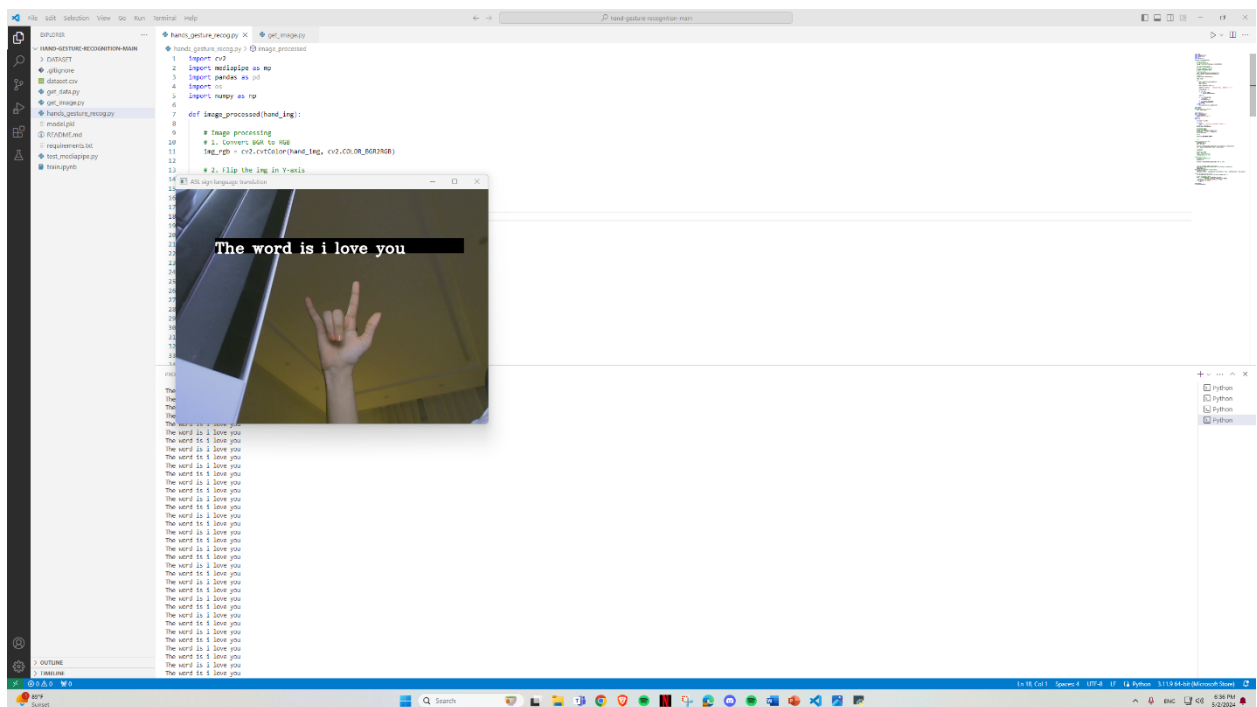


Figure 5. 5: Result of ASL Translation for the 'i love you' Sign

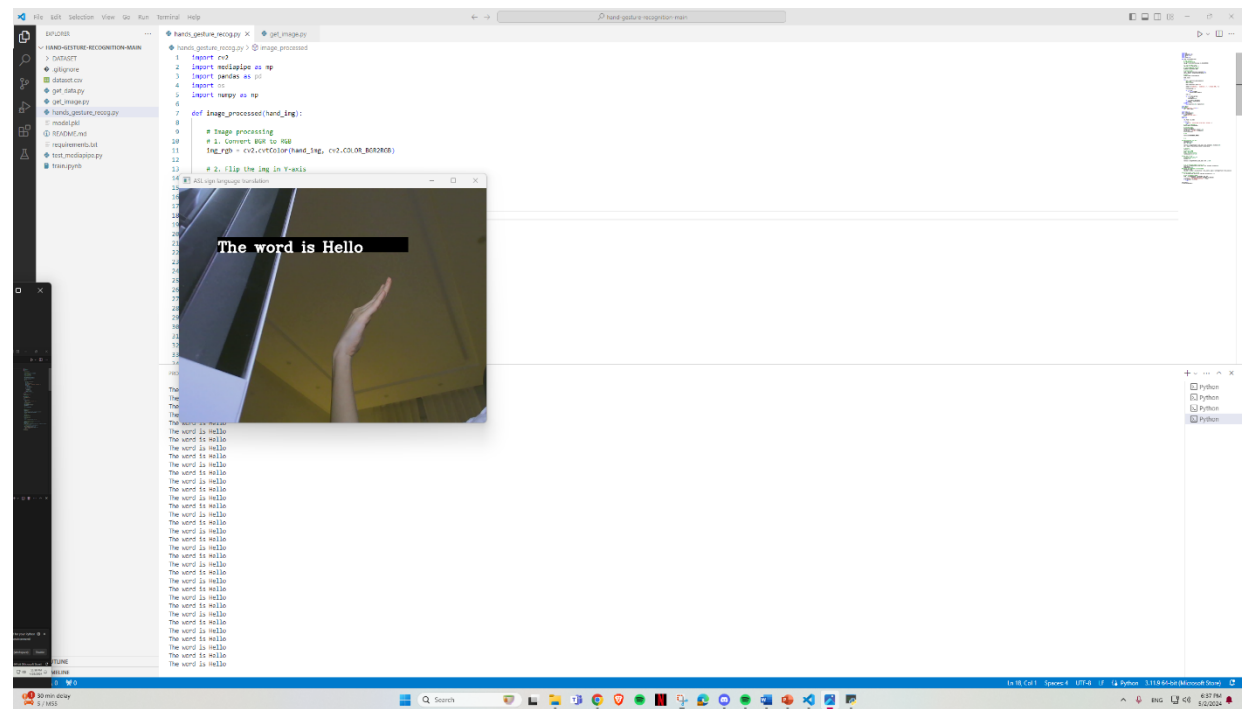


Figure 5. 6: Result of ASL Translation for the 'Hello' Sign

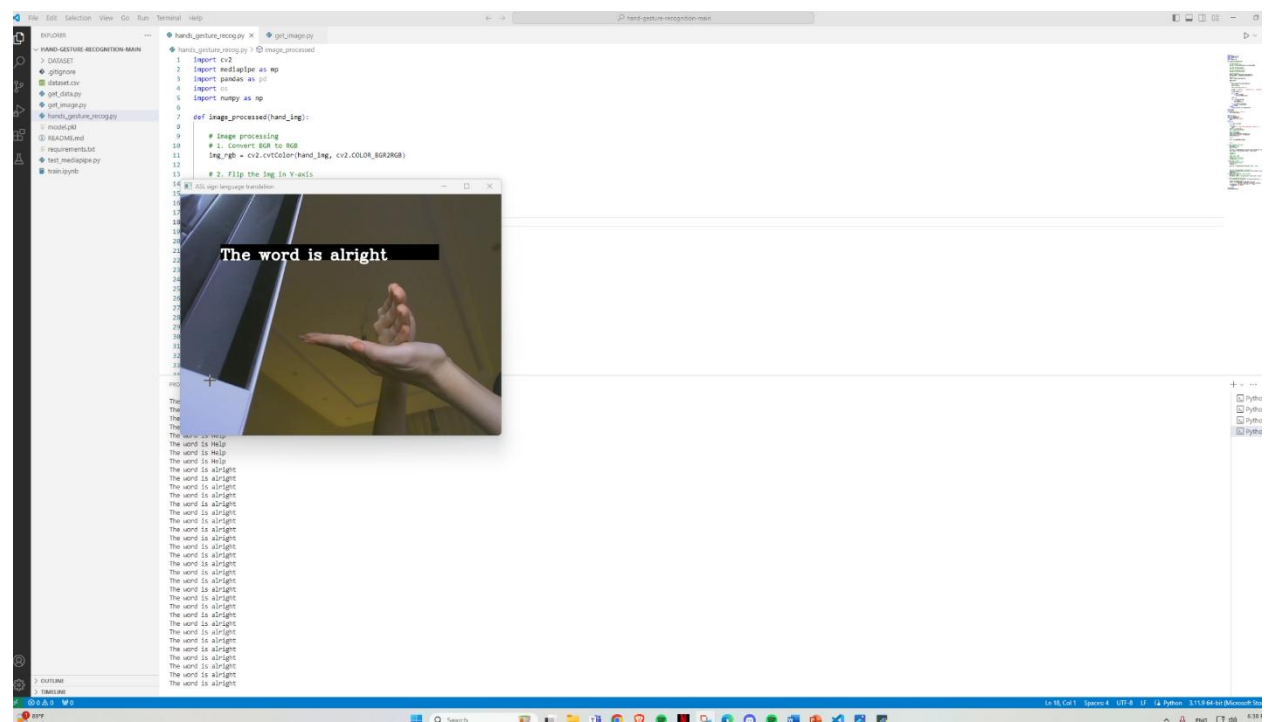


Figure 5. 7: Result of ASL Translation for the 'alright' Sign

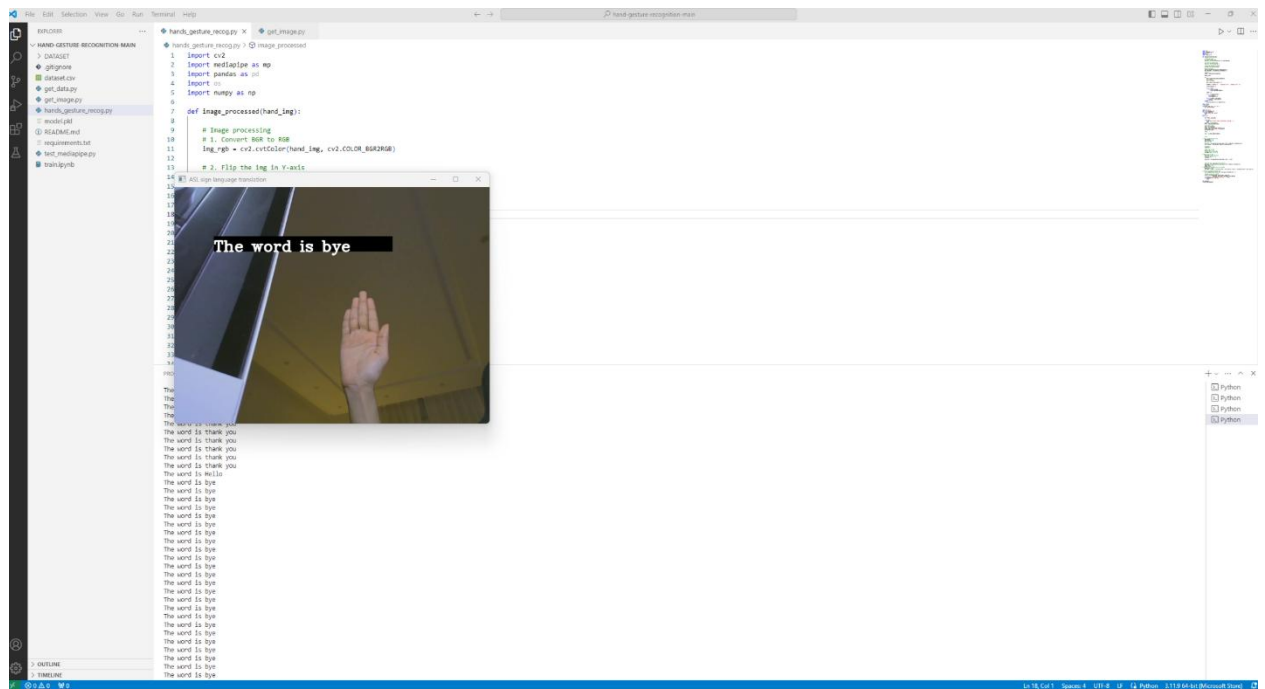


Figure 5. 8:Result of ASL Translation for the 'bye' Sign

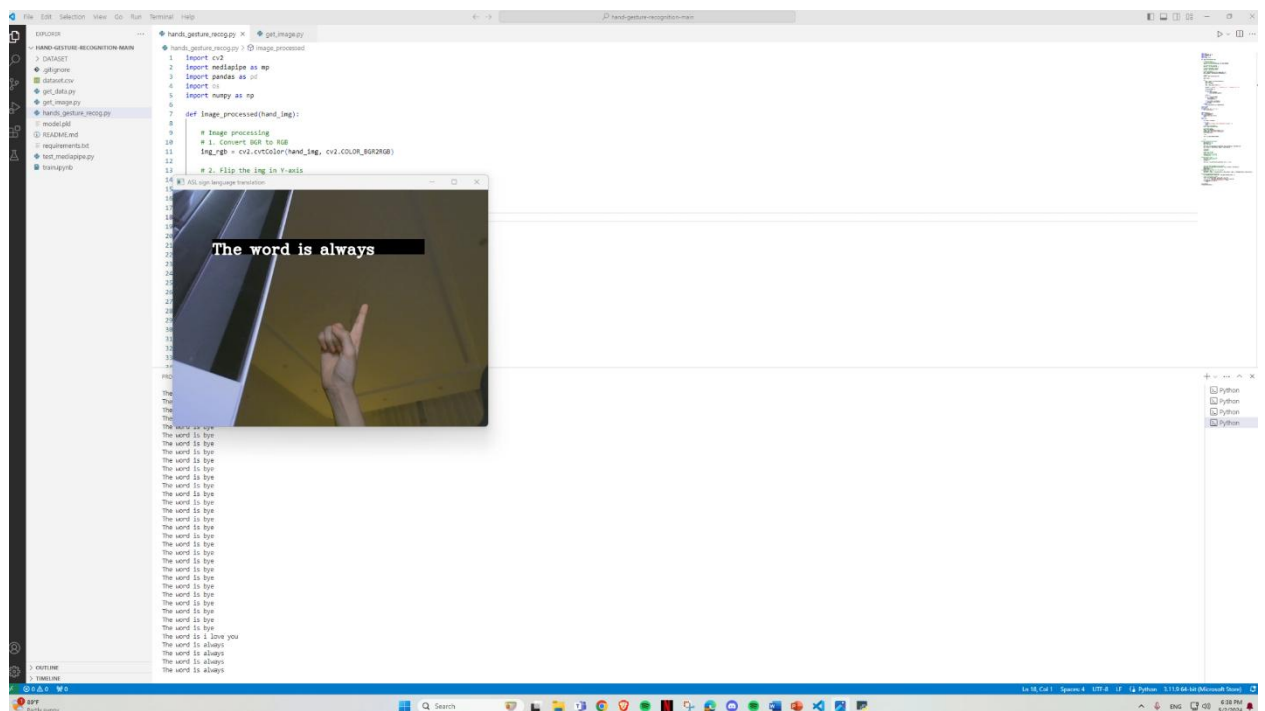


Figure 5. 9:Result of ASL Translation for the 'always' Sign

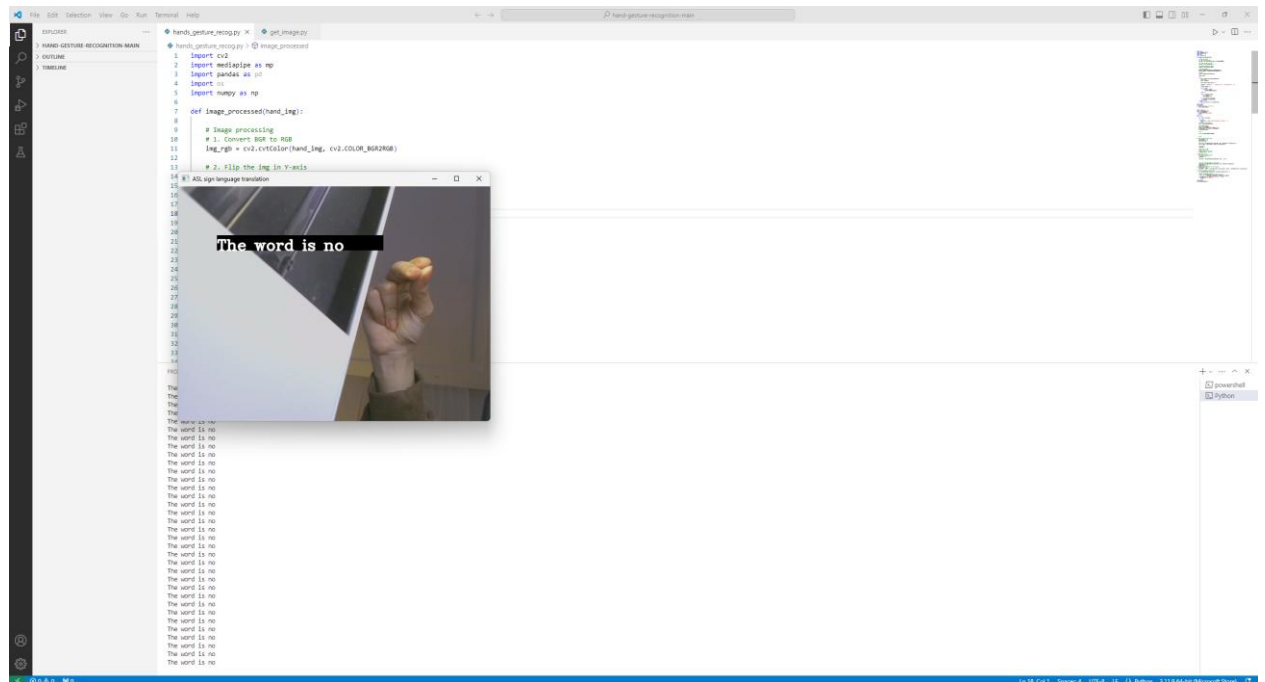


Figure 5. 10: Result of ASL Translation for the 'no' Sign

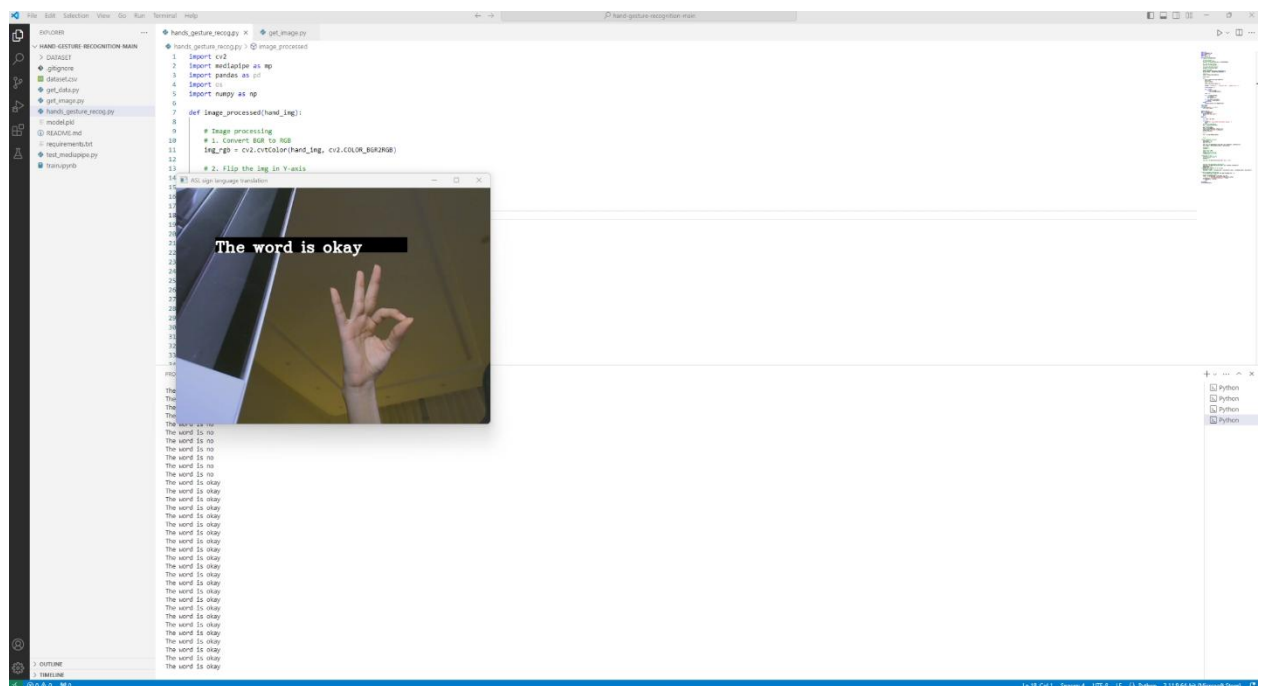


Figure 5. 11: Result of ASL Translation for the 'okay' Sign

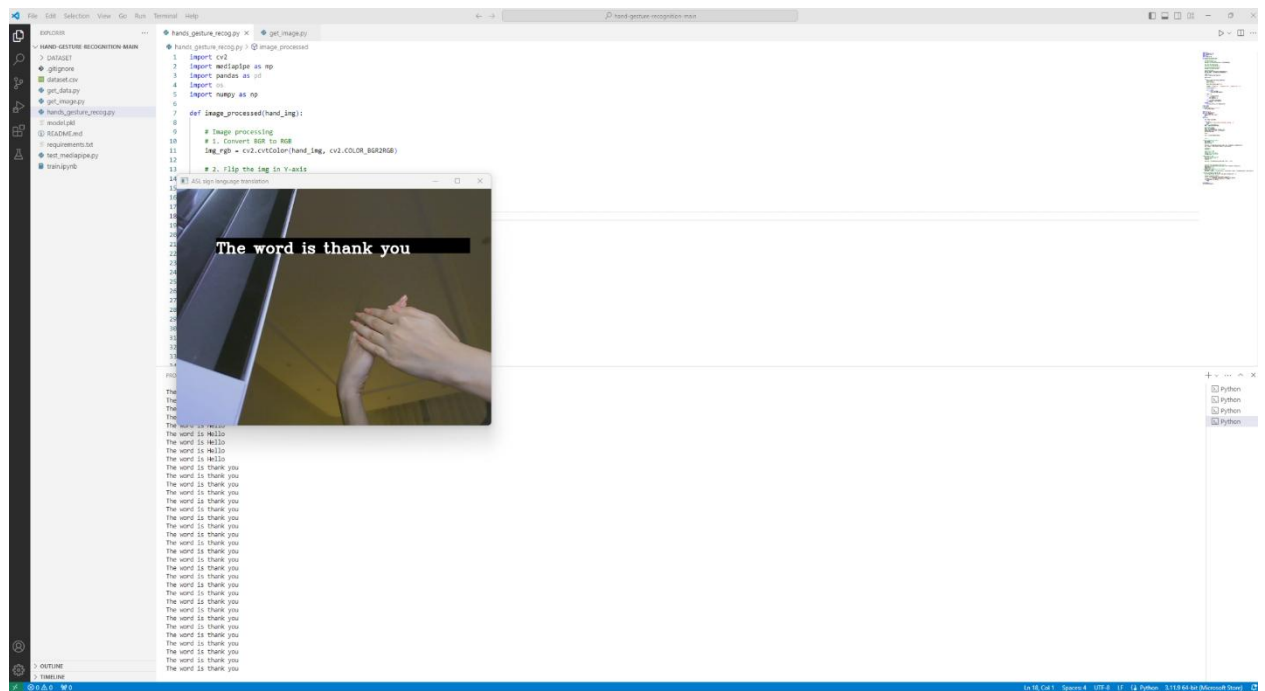


Figure 5. 12:Result of ASL Translation for the 'thank you' Sign

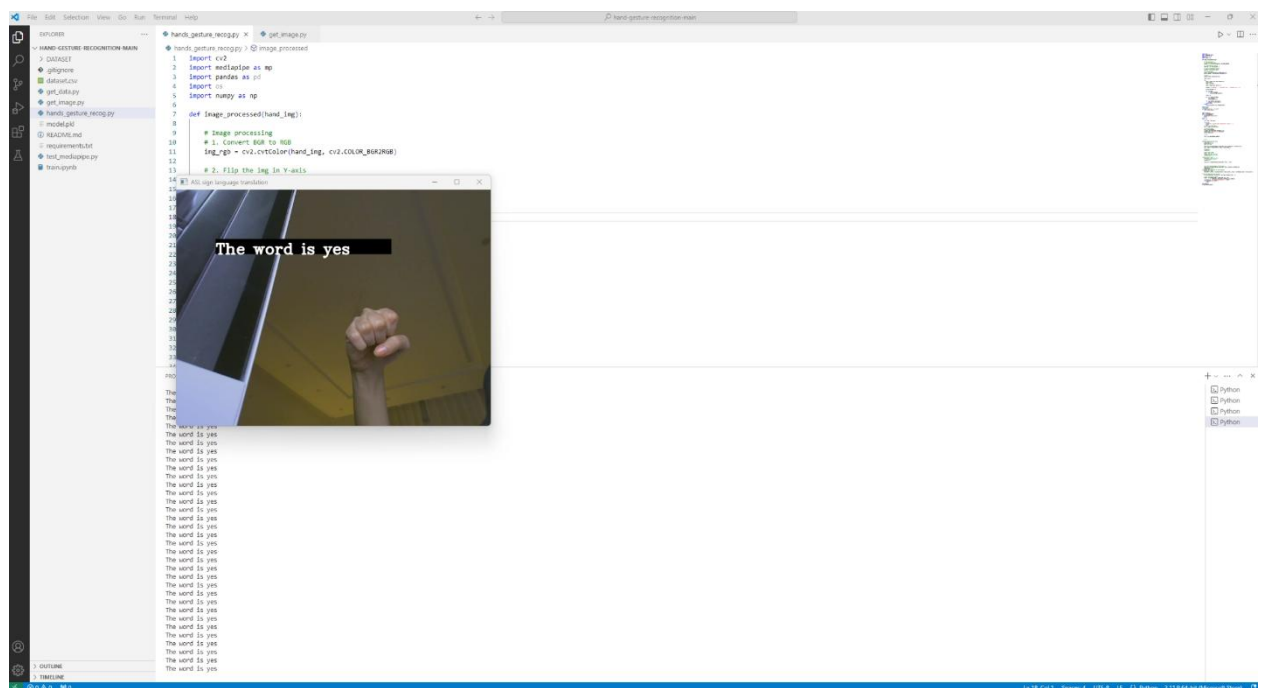


Figure 5. 13:Result of ASL Translation for the 'yes' Sign

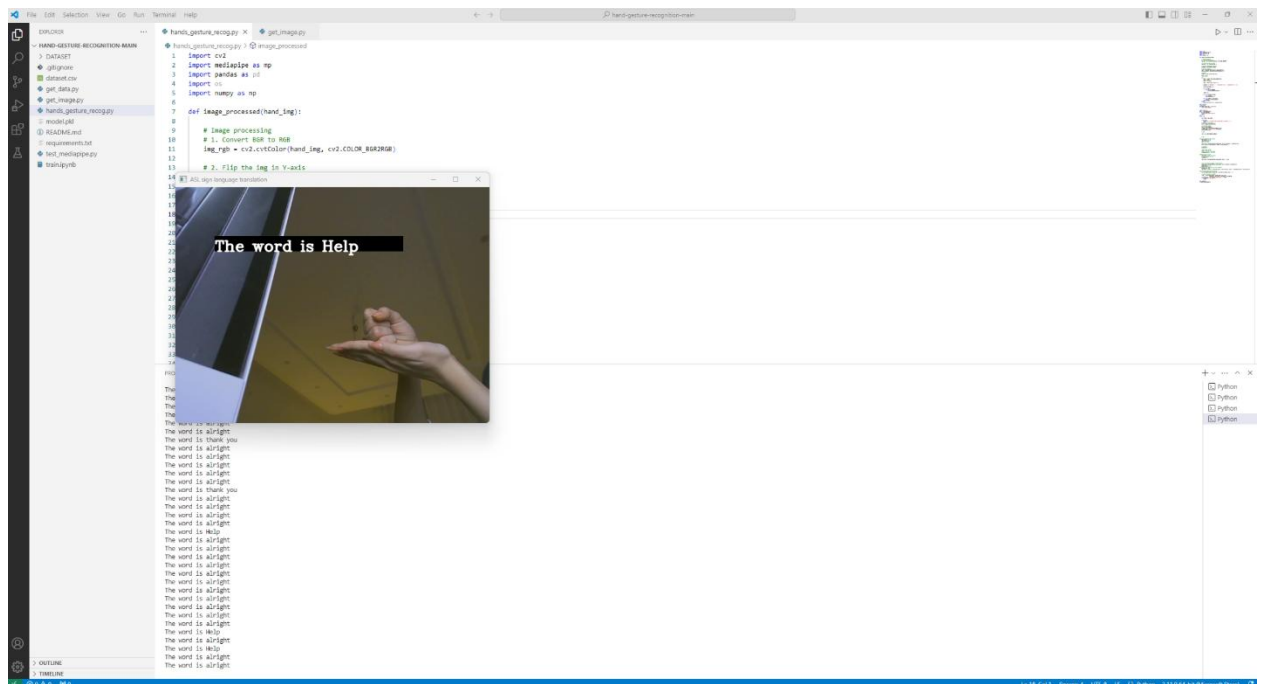


Figure 5. 14:Result of ASL Translation for the 'Help' Sign

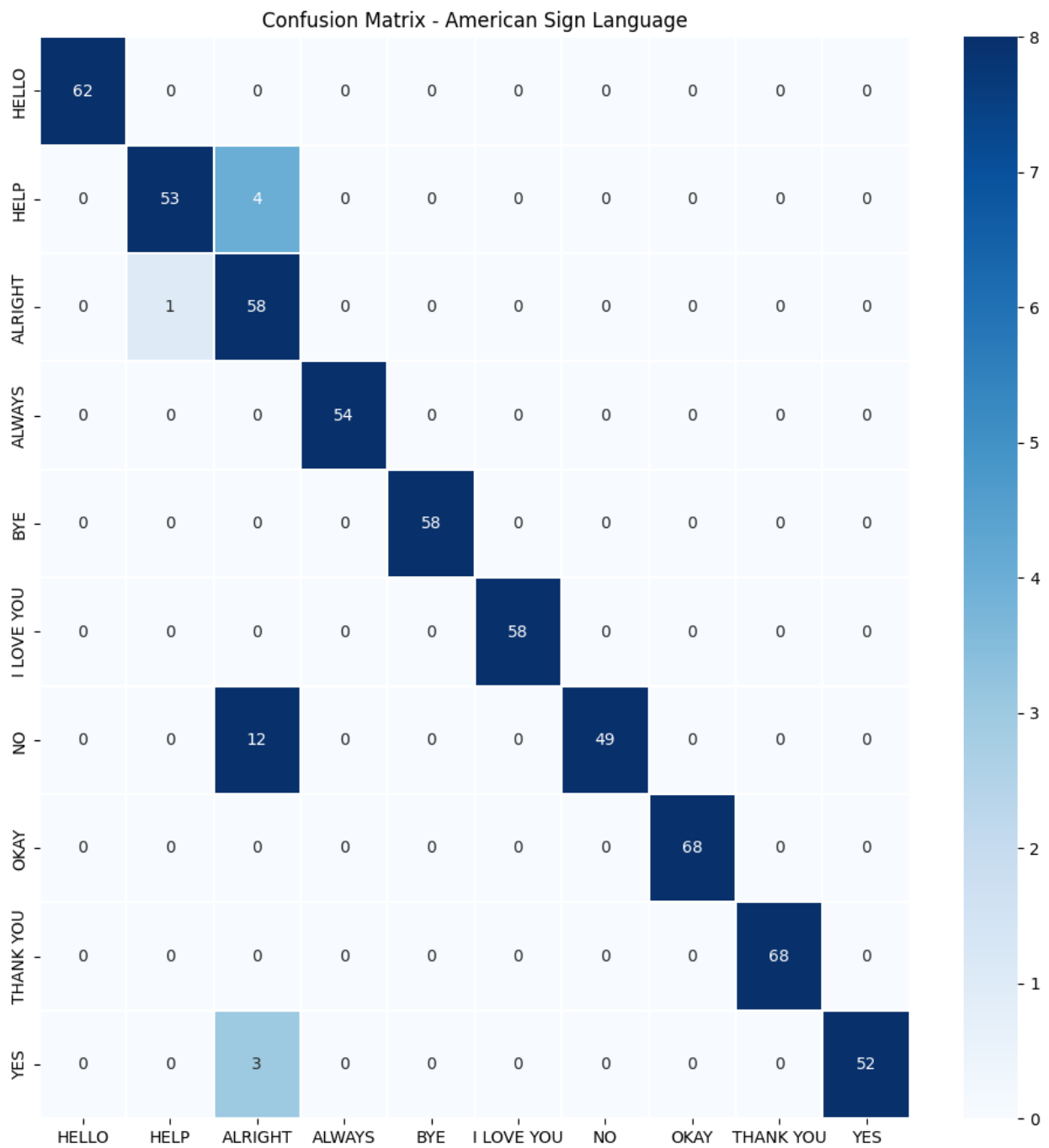


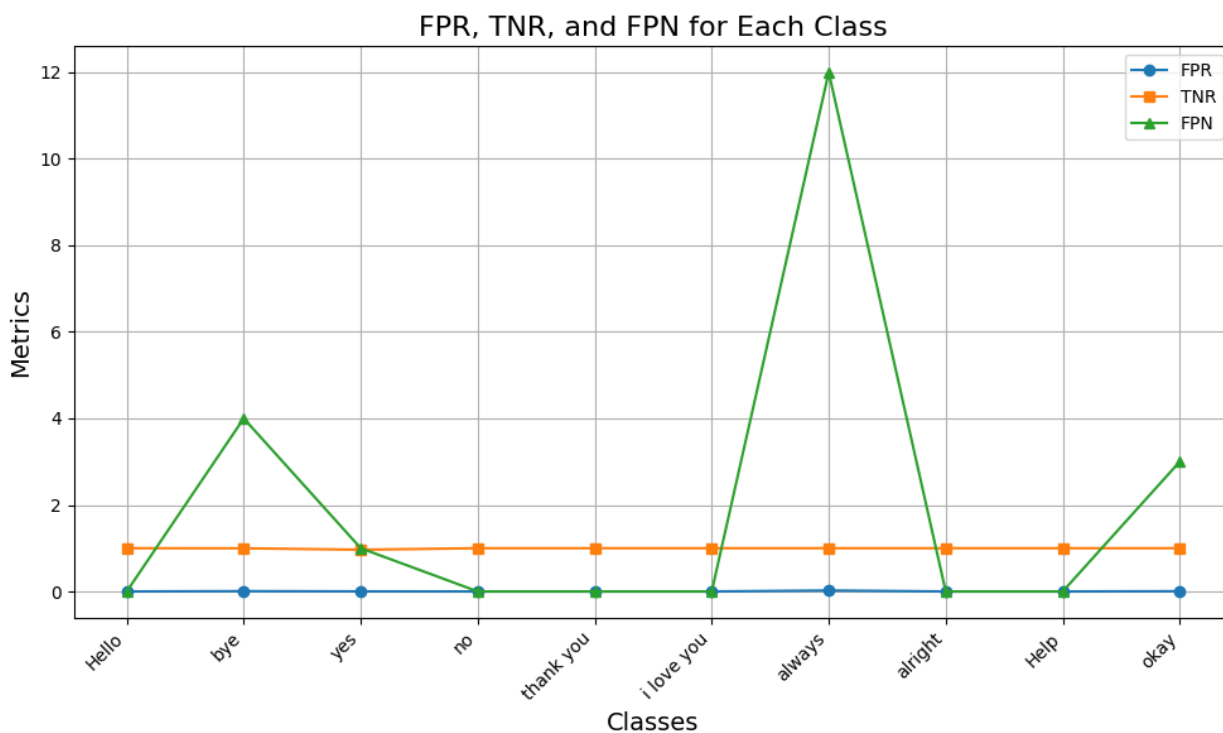
Figure 5. 15:Confusion Matrix - American Sign Language

The confusion matrix in Figure 5.15 show the recognized data for various sign language with different number. For example, “*hello*” that correctly identified is 62 while word “*Yes*” around 52.

Table 5. 1: Confusion Matrix Metrics Table

Class	FPN	TNR	FPR
Hello	0	1.0	0.0
bye	4	0.998	0.007
yes	1	0.965	0.002
no	0	1.0	0.0
thank you	0	1.0	0.0
i love you	0	1.0	0.0
always	12	1.0	0.0222
alright	0	1.0	0.0
Help	0	1.0	0.0
okay	3	1.0	0.005

Table 5.1 and Figure 5.16 illustrates the value of FPN, TNR, and FPR. Each sign language is interpreted successfully with accuracy reach 100% for most of the word.

**Figure 5. 16:** Line Graph of FPR, TNR, and FPN for Each Class

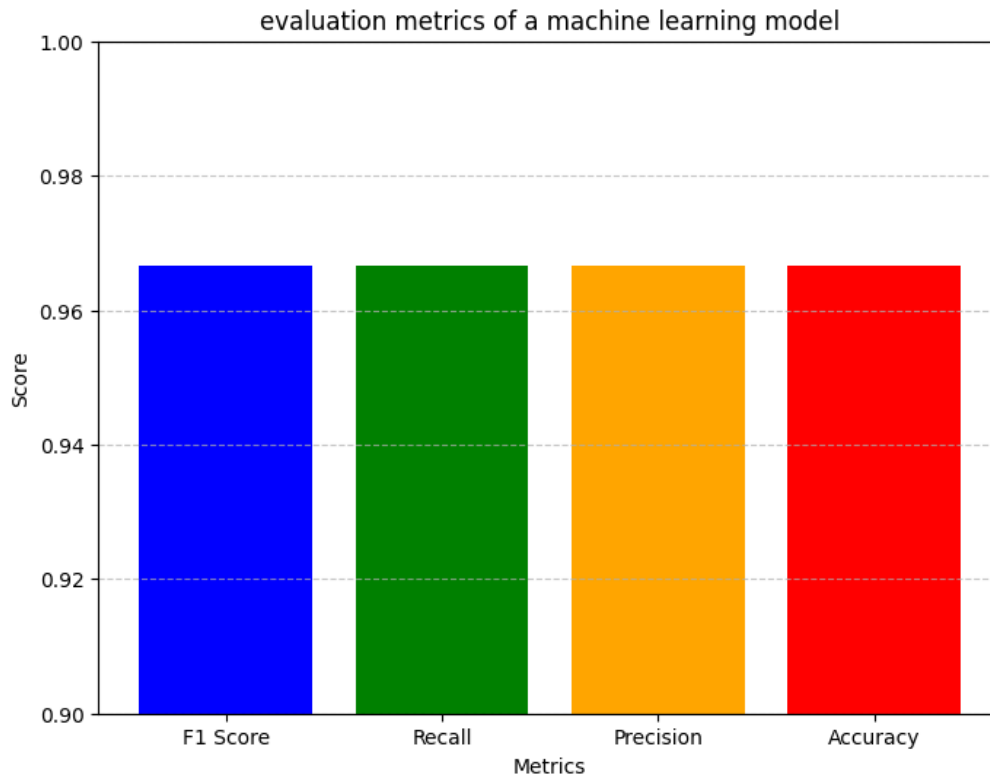


Figure 5. 17:evaluation metrics of a machine learning model

A representation of the F1-score, recall, precision, and accuracy value can be found in Figure 5.17. With a level of accomplishment of 97%, the score might be considered highly encouraging. The fact that this is the case indicates that the system that is being proposed is able to accurately recognize sign language.

5.4 Chapter Summary

This chapter presented the outcomes of the proposed solution for hand gesture recognition using Deep Learning and Media Pipe. It demonstrated the efficiency of the approach in real-time sign language recognition. Through systematic experimentation, we showcased the solution's accuracy and low latency. We discussed the implementation environment, dataset collection, feature extraction, Deep Learning model training, and testing methods. The chapter also outlined the main code files used in the research. Overall, the research findings underscored the solution's potential to enhance accessibility through real-time sign language translation technology.

Chapter 6

CONCLUSION

6.1 Introduction

In conclusion, we succeeded in developing a practical and purposeful system capable of understanding sign language and translating it into the corresponding text. There are still many shortcomings in our system such as this system which can detect alphabetic hand gestures but does not cover body gestures and other dynamic gestures. We are sure that it can be improved in the future.

6.2 Achievements of Research Objectives

Sign language is a visual means of communication that uses hand gestures, facial expressions, body movements, and sometimes including aspects of space and direction to convey meaning. It is used primarily by people who are deaf or hard of hearing but can also be used by hearing individuals who communicate with deaf or hard of hearing individuals, as well as in situations where spoken language may not be practical or accessible. The results of testing show a significant achievement with recognition rate around 97%.

6.3 Limitations and Future Works

Although all the specific objectives of this research have been accomplished, there are some limitations to the proposed method. Although Deep Learning achieved very good results in detecting hand gestures, sign language requires the use of hands to make gestures. This can be a problem for people who do not have full use of their hands. Even seemingly manageable disabilities such as Parkinson's disease or arthritis can pose a major problem for people who must communicate using sign language. In addition, there are also some suggestions that would improve the performance of the proposed system:

- The research can be developed by dealing with sentences instead of individual words.
- There will be multiple languages in the future.
- Deploy the sign language interpreter in the mobile application and through websites on the Internet.

REFERENCES

- [1] S, V. et al. (2018) SIGN LANGUAGE TRANSLATOR USING MACHINE LEARNING. Available at: https://www.ripublication.com/ijaerspl2018/ijaerv13n4spl_01.pdf
- [2] Hureta, M.et al. (2022) KoSign Sign Language Translation Research: Introducing The NIASL2021 Dataset. Available at: <http://www.lrec-conf.org/proceedings/lrec2022/workshops/sltat/pdf/2022.sltat-1.9.pdf> (Accessed: 28 September 2023).
- [3] Ambar, R. et al.(2017) Development of a Wearable Device for Sign Language Recognition. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1019/1/012017/pdf> (Accessed: 28 September 2023).
- [4] Ji, A. et al. (2023) Dataglove for sign language recognition of people with hearing and speech impairment via wearable inertial sensors, MDPI. Available at: <https://www.mdpi.com/1424-8220/23/15/6693> (Accessed: 28 September 2023).
- [5] MacMaster, G. (2020) Sign Language Translation Using Machine Learning and Computer Vision. Available at: <https://scholarworks.uvm.edu/cgi/viewcontent.cgi?article=1367&context=hcoltheses>
- [6] SINANDER, P. (2021) Sign Language Translation. Available at: https://www.kth.se/polopoly_fs/1.1078088.1622113540!/gr20rapport2021-02.pdf.
- [7] Enochian, M.H. (2022) KoSign Sign Language Translation Research: Introducing The NIASL2021 Dataset, ACL Anthology. Available at: <https://aclanthology.org/2022.sltat-1.9>.
- [8] Efthimiou, E. and Fotinea, S.E. (eds.) (2022) 7th Workshop on Sign Language Translation and Avatar Technology: The Junction of the Visual & the Textual Challenges and Perspectives, LREC 2022 Workshop Language Resources and Evaluation Conference. Available at: <http://sltat.cs.depaul.edu/2022/SLTAT2022Proceedings.pdf>.

- [9] Camgoz, N.C. (2020) *Neural Sign Language Recognition and Translation*. Available at: <https://personalpages.surrey.ac.uk/s.hadfield/papers/camgoz2020phd.pdf>.
- [10] Ambar, R. and Kar Fai, C. (2018) Development of a Wearable Device for Sign Language Recognition, IOP science. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1019/1/012017>.
- [11] Liang, Z., Li, H. and Chai, J. (2023) mdpi. Available at: <https://www.mdpi.com/2079-9292/12/12/2678>.
- [12] Ji, A. et al. (2023) Dataglove for Sign Language Recognition of People with Hearing and Speech Impairment via Wearable Inertial SensorsA, Pubmed. Available at: <https://pubmed.ncbi.nlm.nih.gov/37571476/>.
- [13] Ambar, R. and Kar Fai, C. (2018) Development of a Wearable Device for Sign Language Recognition, IOP science. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1019/1/012017>.
- [14] Ambar, R. and Kar Fai, C. (2018) Development of a Wearable Device for Sign Language Recognition, IOP science. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1019/1/012017/pdf>.
- [15] Ambar, R. and Kar Fai, C. (2018) Development of a Wearable Device for Sign Language Recognition, IOP science. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1019/1/012017>.
- [16] <https://www.kaggle.com/datasets/ahmedkhanak1995/sign-language-gesture-images-dataset>
- [17] Gorden, D., Use Case Diagrams Damian Gordon. Available at: <https://slideplayer.com/slide/2303583/>.
- [18] Resolve Software Issue UML Activity Diagram Example. Available at: <https://www.uml-diagrams.org/software-resolve-issue-uml-activity-diagram-example.html>.
- [19] Unified Modeling Language (UML) | Sequence Diagrams. Available at: <https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/>.

- [20] Implementation phase (no date) Researchmanagement-training. Available at: <https://www.researchmanagement-training.net/implementation-phase/>.
- [21] Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN (no date) Mathworks. Available at: <https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html>.
- [22] Hoque, O. (2018) Real Time Bangladeshi Sign Language Detection using Faster R-CNN. Available at: <https://arxiv.org/pdf/1811.12813.pdf> .
- [23] Al-Sawas, M. (no date a) *What is the concept of software maintenance?* Available at: <https://digitsmark.com/ar/blogs/software-maintenance>.
- [24] Open Source Data Labeling Platform. Available at: <https://labelstud.io/>.
- [25] TensorFlow. Available at: <https://www.tensorflow.org/?hl=ar>.
- [26] Python. Available at: <https://www.python.org/>.
- [27] OpenCV. Available at: <https://opencv.org/>.