



THESIS

Submitted in partial fulfilment of the requirements for a
Master's degree

Design and implementation of an Arabic Sign language recognition system

Field: Computer Science

Specialty: Intelligent Computer Systems

By

Ahmed Zaki Lamri

In front of the Jury composed of:

Quality	Name and Surname	Rank	University
President :	Mrs.Maataallah.M	MCB	Chadli Bendjedid El-Tarf
Supervisor :	Mrs.MAKHLOUF A	MCB	Chadli Bendjedid El-Tarf
Examiner :	Mr.BENMACHICHE A	MCA	Chadli Bendjedid El-Tarf

University Year: 2022/2023

Acknowledgment

I would like to begin by expressing my deepest gratitude to ALLAH for His unwavering guidance and blessings throughout my academic journey. His divine guidance has illuminated my path and granted me the strength and perseverance to pursue knowledge.

*I am immensely thankful to **Dr. MAKHLOUF AMINA** for her invaluable advice, guidance, and continuous support. Her expertise, insightful feedback, and dedication have played a pivotal role in shaping the direction and quality of this work. Her mentorship has been instrumental in my growth as a researcher.*

*I would also like to extend my sincere appreciation to all those who have contributed, both directly and indirectly, to the realization of this work. Their support, encouragement, and assistance have been invaluable. Specifically, I would like to express my gratitude to **Piso, Magino, Yasser, Ammar, Djaber, Sofiane, Mehdi, Ali, Larbi, Abderrahmane, Houdna, Ryhab, and Joumana** for their contributions in helping to create our dataset. Their willingness to collaborate and assist me in this endeavor has been instrumental in the success of this research.*

I am deeply indebted to my friends and family for their unwavering support, understanding, and encouragement throughout this challenging journey. Their belief in me and their constant encouragement have been a source of inspiration and motivation.

Furthermore, I would like to acknowledge the contribution of the unnamed individuals who have provided valuable insights, resources, and support during this research endeavor. Your assistance has been truly appreciated.

Finally, I would like to express my sincere gratitude to all the researchers, scholars, and authors whose works have served as a foundation for this study.

Their contributions to the field of Computer Vision have been instrumental in shaping my understanding and informing my research.

In conclusion, I am humbled and grateful for the support, guidance, and contributions of all those mentioned and unmentioned in this acknowledgment. Their involvement has enriched this research journey, and I am indebted to each and every one of them.

Lamri Ahmed Zaki

Dedication

I dedicate this modest work to:

- *My dear father, who has been a constant source of love, support, and inspiration in my life. Your guidance and encouragement have shaped me into the person I am today.*
- *My light and pillar of patience in life, my lovable mother. Your unwavering love, sacrifices, and unwavering belief in me have given me strength and determination throughout this journey.*
- *My brother Islam , who have always been there for me, offering support, encouragement, and camaraderie. Your presence has made my life richer and more meaningful.*
- *Itech Club, for providing a platform where I could enhance my technical skills, collaborate with passionate individuals, and foster a spirit of innovation and learning.*

May Allah bless and protect each one of you and grant you all happiness, success, and fulfillment in life.

الملخص

تتركز البحوث في السنوات الأخيرة على مجال معالجة الكلام وتطبيقاتها في تعلم الآلة والتعرف على الكلام، ولا سيما في تطوير نماذج عميقة في مجال التعلم العميق. حققت التقنيات العميقة نجاحًا كبيرًا في التعرف على الكلام وتحقيق نتائج متفوقة في مختلف المجالات. لذا، فإن هذه الدراسة تركز على تصميم نظام لغة الإشارة باستخدام نموذج الذاكرة القصيرة الطويلة المدى.

يهدف البحث إلى تطوير نظام قوي يستطيع تفسير إيماءات لغة الإشارة بدقة.

يتم استكشاف مجال المشكلة وتقديم فهم شامل لأحدث التطورات في تعرف لغة الإشارة، كما يتم استعراض الأعمال المتعلقة بالمجال.

تتناول الدراسة تصميم النظام من الناحية النظرية، وتقديم مجموعة البيانات المستخدمة، وتقنيات التعلم، وهندسة النموذج، وتقييم النموذج. يتم استخدام نموذج الذاكرة القصيرة الطويلة المدى للتعامل مع البيانات المتسلسلة، ويتم تدريبه لاستيعاب التبعيات الزمنية في النظام وتقييم أدائه باستخدام مجموعة بيانات مناسبة. تتم مناقشة النتائج المحصلة ومقارنتها مع الأداء الآخر.

يحقق نظام التعرف على لغة الإشارة القائم على نموذج الذاكرة القصيرة الطويلة المدى دقة مذهلة تبلغ 98.53 % تبرهن الدقة العالية على فعالية النموذج المختار وملائمته لمهام التعرف على لغة الإشارة.

تسهم نتائج هذا البحث في تطوير مجال التعرف على لغة الإشارة وتحمل إمكانات كبيرة لتطبيقات عملية في تحسين التواصل للأشخاص الصم وضعاف السمع.

الكلمات المفتاحية: معالجة اللغة الطبيعية، التعرف على لغة الإشارة، التعلم العميق، تعلم الآلة، الذاكرة القصيرة الطويلة المدى

Abstract

In recent years, significant research has been dedicated to machine learning techniques for speech processing, particularly in the field of speech recognition. However, more recently, researchers have shifted their focus towards deep learning approaches for speech-related applications. Deep learning has proven to deliver superior outcomes across various domains, including speech, making it an intriguing area of investigation. As deep learning emerged as a new frontier in machine learning, it opened up new possibilities for speech applications.

This thesis presents a design and implementation of a sign language recognition (SLR) system using a Long Short-Term Memory (LSTM) model. The aim of the research is to develop a robust system capable of accurately interpreting sign language gestures. The problem area of SLR is explored, and related work in the field is reviewed to provide a comprehensive understanding of the current state of the art. The LSTM-based sign language recognition system achieves an impressive accuracy of 98.53%. The high accuracy demonstrates the effectiveness of the chosen model and its suitability for sign language recognition tasks. The results of this research contribute to advancing the field of sign language recognition and hold great potential for practical applications in improving communication for the deaf and hard of hearing.

Keywords: Natural Language Processing, Sign Language Recognition, Deep Learning, Machine Learning, LSTM ,

Contents

Contents	i
List of Figures	v
List of Tables	ix
List of acronyms	x
General Introduction	1
1 State of art	3
1.1 Introduction	3
1.2 Sign language	3
1.2.1 History	3
1.2.2 Definitions	4
1.2.3 Type of Sign language	4
1.3 Recognition System	5
1.3.1 Definitions	5
1.3.2 Types of recognition systems	5
1.3.2.1 Face recognition	5
1.3.2.2 Voice recognition	6
1.3.2.3 Speech recognition	7
1.3.2.4 Fingerprint recognition	8
1.3.2.5 Iris recognition	8

1.3.2.6	Handwriting recognition	9
1.3.2.7	Object recognition	9
1.3.2.8	Signature recognition	9
1.3.2.9	Barcode recognition	10
1.3.2.10	OCR (Optical Character Recognition)	10
1.4	Sign language Recognition	11
1.4.1	Definitions	11
1.4.2	Motivation	11
1.4.3	Problem Area of SLR	12
1.4.3.1	Segmentation	12
1.4.3.2	Unrestricted environment	13
1.4.3.3	Size of dictionary or Limited training data	13
1.4.3.4	Invariance	13
1.4.3.5	Variety	14
1.4.3.6	Motion gestures	14
1.4.3.7	Feature Selection	14
1.4.3.8	Feature Extraction	15
1.4.3.9	User engagement and usability	15
1.4.4	Types of sign language recognition	15
1.4.4.1	Hand gesture recognition	15
1.4.4.2	Face and body posture recognition	15
1.4.4.3	Sign language translation	15
1.4.4.4	Sign language tutoring systems	15
1.4.4.5	Assistive technologies for deaf individuals	16
1.4.5	Sign Language Recognition System	16
1.4.5.1	Data acquisition	16
1.4.5.2	Feature Extraction	17
1.4.5.3	Recognition	22
1.4.6	SLR using Deep learning	23
1.4.6.1	Artificial neural networks (ANN)	23

1.4.6.2	Deep Neural Network (DNN)	23
1.4.6.3	Multilayer Perceptron (MLP)	24
1.4.6.4	Recurrent Neural Networks (RNN)	25
1.4.6.5	Convolutional Neural Networks (CNN)	28
1.4.6.6	Transformers	30
1.5	Related Work	31
1.6	Advantages and disadvantages	32
1.7	Conclusion	33
2	Conceptual Study	34
2.1	Introduction	34
2.2	System design	34
2.2.1	Data Acquisition	35
2.2.2	Preparing Data	35
2.2.3	Split Data into Train and Test	37
2.3	Presenting Dataset	38
2.4	Optimizer	39
2.5	Model architecture	46
2.5.1	LSTM model	46
2.5.2	Model configuration and Number of parameters	47
2.6	Model Evaluation	48
2.6.1	Metrics	48
2.7	Conclusion	50
3	Results and Implementation work	51
3.1	Introduction	51
3.2	Representation of the development tools	52
3.2.1	Physical environment	52
3.2.2	Software and libraries used in the implementation	52
3.3	Experiments, comparison and discussion the obtained results	54
3.3.1	Experiments and Results	54

3.3.1.1	In terms of models	55
3.3.1.2	In terms of optimizer:	71
3.4	Prediction Results And Model Evaluation	76
3.4.1	Accuracy :	76
3.4.2	Precision, Recall and F1 score :	76
3.4.3	Precision-Recall curve	77
3.4.4	Receiver Operating Characteristic (ROC)	78
3.4.5	Confusion Matrix :	80
3.5	Representing our system interface	81
3.6	Test	87
3.7	Conclusion	89
	Conclusions	91
3.8	Conclusion	91
3.9	Future work	92
	Annex: ArSLA28 Dataset	94
	Bibliography	99

List of Figures

1.1	Block diagram of the human face recognition system	6
1.2	Voice recognition process	7
1.3	Speech Recognition process	7
1.4	Process of fingerprint identifications	8
1.5	Iris recognition system	9
1.6	General scheme of a biometric recognition system	10
1.7	Block Diagram of OCR	11
1.8	Block Diagram of SLR	16
1.9	A typical artificial neural network (ANN).	23
1.10	Construction of the deep neural network (DNN) model.	24
1.11	Structure of a multilayer perceptron (MLP) algorithm	25
1.12	Schematic architecture of a recurrent neural network	25
1.13	Schematic architecture of a LSTM	26
1.14	Schematic architecture of a GRU	27
1.15	Details on Convolution layer	28
1.16	Stride 1, the filter window moves only one time for each connection	29
1.17	The max-pooling with 2x2 filter and stride 2 lead to down-sampling of each 2x2 blocks is mapped to 1 block (pixel).	29
1.18	The transformer model	30
2.1	Our system architecture	34
2.2	Attention Mesh: Overview of model architecture	35

2.3	33 pose landmarks.	36
2.4	21 hand landmarks.	37
2.5	Representation of the Arabic sign language alphabets	38
2.6	our Model architecture	46
2.7	The configuration of our model	47
3.1	Chart of accuracy and loss of Model 1	55
3.2	Chart of accuracy and loss of Model 2	56
3.3	Chart of accuracy and loss of Model 3	56
3.4	Chart of accuracy and loss of Model 4(a)	57
3.5	Chart of accuracy and loss of Model 4(b)	57
3.6	Chart of accuracy and loss of Model 4(c)	58
3.7	Chart of accuracy and loss of Model 4(d)	58
3.8	Chart of accuracy and loss of Model 4(e)	59
3.9	Chart of accuracy and loss of Model 4(f)	59
3.10	Chart of accuracy and loss of Model 5(a)	60
3.11	Chart of accuracy and loss of Model 5(b)	60
3.12	Chart of accuracy and loss of Model 5(c)	61
3.13	Chart of accuracy and loss of Model 5(d)	61
3.14	Chart of accuracy and loss of Model 5(e)	62
3.15	Chart of accuracy and loss of Model 5(f)	62
3.16	Chart of accuracy and loss of Model 6(a)	63
3.17	Chart of accuracy and loss of Model 6(b)	63
3.18	Chart of accuracy and loss of Model 6(c)	64
3.19	Chart of accuracy and loss of Model 6(d)	64
3.20	Chart of accuracy and loss of Model 6(e)	65
3.21	Chart of accuracy and loss of Model 6(f)	65
3.22	Chart of accuracy and loss of Model 7(a)	66
3.23	Chart of accuracy and loss of Model 7(b)	66
3.24	Chart of accuracy and loss of Model 7(c)	67
3.25	Chart of accuracy and loss of Model 7(d)	67

3.26	Chart of accuracy and loss of Model 7(e)	68
3.27	Chart of accuracy and loss of Model 7(f)	69
3.28	Accuracy (Model 7(Adadelta))	71
3.29	Chart of accuracy and loss of Model 7(Adafactor)	71
3.30	Chart of accuracy and loss of Model 7(Adagrad)	72
3.31	Chart of accuracy and loss of Model 7(RMSprop)	72
3.32	Chart of accuracy and loss of Model 7(Adamax)	73
3.33	Chart of accuracy and loss of Model 7(AdamW)	73
3.34	Chart of accuracy and loss of Model 7(Nadam)	74
3.35	Chart of accuracy and loss of Model 7(Ftrl)	74
3.36	Chart of accuracy and loss of Model 7(SGD)	75
3.37	Precision-Recall Curve	77
3.38	Receiver Operating Characteristic (ROC) curve	78
3.39	The confusion matrix of Our Model	80
3.40	Welcome Page	81
3.41	Home Page	82
3.42	Components Home Page	83
3.43	no hand detected	84
3.44	image with landmark	85
3.45	image without the landmark	85
3.46	Guide Page in Arabic	86
3.47	Home Page in Arabic	86
3.48	Output “Alif ”without Landmark	87
3.49	Output “Alif ”with Landmark	87
3.50	Sign “Alif ”	87
3.51	Output “BA ”without Landmark	87
3.52	Output “BA ”with Landmark	87
3.53	Sign “BA ”	87
3.54	Output “TA ”without Landmark	88
3.55	Output “TA ”with Landmark	88

3.56 Sign “TA ”	88
3.57 Output “SAD ”without Landmark	88
3.58 Output “SAD ”with Landmark	88
3.59 Sign “SAD ”	88
3.60 Output “LAM ”without Landmark	89
3.61 Output “LAM ”with Landmark	89
3.62 Sign “LAM ”	89
3.63 Sign Dha captured frames from the dataset.	94
3.64 Sign Sin captured frames from the dataset.	95
3.65 Sign Za captured frames from the dataset.	96
3.66 Sign Fa captured frames from the dataset.	97
3.67 Sign Lam captured frames from the dataset.	98

List of Tables

1.1	Feature extraction methods employed by various researchers	21
1.2	Classification methods used by researchers	22
1.3	Summary of related works	32
2.1	Confusion Matrix	49
3.1	Desktop Hardware	52
3.2	Model Performance	70
3.3	Comparison of different optimizers	75
3.4	Classification Results	77

List of acronyms

ANN Artificial neural network. 23, 25, 28,

CCH Chain code histogram. 31,

CNN Convolutional Neural Network. 24,

DNN Deep Neural Networks. 23,

GRU Gated recurrent unit. 25, 27,

HOG Histogram of oriented gradients. 31,

LBP local binary pattern. 31,

LSTM Long short-term memory. 25–27, 47,

MLP Multi-Layer Perceptron. 24,

NLP Natural Language Processing. 3, 23, 30,

OpenCV Open Source Computer Vision. 53,

PNN Probabilistic neural network. 31,

RNN Recurrent Neural Network. 24, 26, 27,

SLR Sign Language Recognition. 16,

SVM Support vector machine. 31, 32,

General Introduction

Sign language is a way of communicating using visual gestures, facial expressions, and body movements instead of spoken words. Each sign language is unique to a particular geographical region or cultural group, such as American Sign Language (ASL) and British Sign Language (BSL). Sign languages are whole and complicated languages, with their personal grammar, vocabulary, and syntax. They allow deaf, hard-of-hearing, and hearing individuals to communicate with one another and are widely used as the primary means of communication in deaf communities.

NLP is a field that bridges the gap between human language and computer understanding. It focuses on creating algorithms that allow computers to interpret, generate, and analyze human language. The ultimate goal is to improve communication between humans and computers, with applications such as sentiment analysis, translation, text classification, and entity recognition.

Sign language recognition is the process of using technology to identify and interpret sign language gestures and movements. It is a field of study that seeks to improve communication between the deaf community and the hearing world by developing systems that can accurately recognize sign language and translate it into spoken or written language. Sign language recognition systems are used in a variety of applications, including educational settings, healthcare, and public services. The challenge in this field lies in capturing the nuances and complexities of sign language, including gesture speed, movement direction, and facial expressions. Nevertheless, sign language recognition is a rapidly growing field with great potential to enhance communication and access to information for deaf individuals.

This thesis investigates the use of deep learning techniques to improve sign language recognition accuracy. The research will focus on exploring and creating deep learning models specifically designed for sign language recognition. The final intention is to beautify verbal exchange among the deaf network and the hearing world by providing a more accurate and efficient means of sign language interpretation. The study will also address any obstacles and limitations in the use of deep learning for sign language recognition and strive to provide innovative solutions.

The organization of this thesis is as follows:

- **State of the Art**

In this chapter, we provide an overview of the current state of sign language recognition (SLR). We delve into the fundamental concepts of sign language, system recognition, and the significance of SLR. The chapter explores the problem areas in SLR and the different types of SLR systems. Additionally, we review related works and research studies that have contributed to the advancement of SLR.

- **Conceptual Study**

The second chapter focuses on the conceptual study of our sign language recognition system. We discuss the system design, including the selection of appropriate development tools and frameworks. The chapter also covers the presentation of the dataset used for training and evaluation, as well as the optimization techniques employed to enhance the model's performance. Furthermore, we present the architecture of our deep learning model, Model evaluation methods and metrics are also discussed in this chapter.

- **Results and Implementation Work**

In last Chapter, we present the results and implementation work of our sign language recognition system. We discuss the representation of the development tools and frameworks used in implementing the system. Experimental results are presented, highlighting the accuracy and performance achieved our model. We compare and discuss the obtained results with previous approaches in the field. Additionally, we present the prediction results and model evaluation metrics to assess the system's effectiveness. Furthermore, we showcase the user interface of our system and discuss the testing process to validate its usability and functionality.

State of art

1.1 Introduction

The purpose of Sign Language and NLP (Natural Language Processing) is to increase the ease of communication between people and computers. Sign Language utilizes visual cues, such as gestures and facial expressions, to communicate with those who are deaf, hard-of-hearing, or hearing. On the other hand, NLP works on developing computer algorithms that allow for the understanding, generation, and analysis of human language, resulting in better communication between people and technology.

1.2 Sign language

1.2.1 History

Sign language stretches out back to the 17th century, when a French teacher named Charles-Michel de l'épée developed a sign language framework to train hard of hearing understudies in Paris. This framework, known as French Sign Language (LSF), laid out the establishment for the majority other sign languages that have emerged across the world throughout the long term.[3]

Sign language and deaf culture faced substantial threats in the late nineteenth and early twentieth century from oralist groups that attempted to restrict sign language and encourage the use of speech and lip reading instead. Despite these obstacles, sign language and deaf culture have thrived, and many governments now recognize sign languages as full-fledged languages.

Sign language is now used by millions of people worldwide and is evolving and changing as it is passed down from generation to generation. Its linguistics has also increased in popularity, with scholars investigating the linguistic structure and features of sign languages, as well as the social and cultural

contexts in which they are utilized.

1.2.2 Definitions

Sign languages are natural languages that have evolved independently in many cultures and places, with their own grammar, lexicon, and syntax. Deaf people have used sign languages for generations to communicate with each other and with hearing people. Each sign language has variations in hand shape, movement pattern, and position of the hand, face, and body parts that make up each sign.[79]

1.2.3 Type of Sign language

There are many types of sign languages used around the world, including:

- Arabic Sign Language (ArSL) - used in Arabic world.[5]
- American Sign Language (ASL) - used in the United States and Canada.[97]
- British Sign Language (BSL) - used in the United Kingdom.[91]
- Auslan - used in Australia.[51]
- French Sign Language (LSF) - used in France.[36]
- German Sign Language (DGS) - used in Germany.[80]
- Japanese Sign Language (JSL) - used in Japan.[19]
- Argentine Sign Language (LSA) - used in Argentina.[65]

Each sign language has its own unique grammar, vocabulary, and cultural history. The specific sign language used depends on the region and the deaf community in that area.

1.3 Recognition System

1.3.1 Definitions

A recognition system is a sort of innovation that is intended to distinguish, comprehend, and decipher a particular information. The info can be anything from a voice, a picture, a mark, or even biometric information, for example, fingerprints or iris examines. [94]

The objective of a recognition system is to coordinate the contribution with a pre-characterized set of information to decide whether it is perceived. For instance, a face recognition system would look at the info picture of a face with a data set of known countenances to decide whether it matches any of them.[57]

Recognition systems are utilized in various applications, including security, biometrics, picture and discourse recognition, and normal language handling. The exactness and unwavering quality of recognition systems rely upon the nature of the information, the calculations utilized, and the climate where they are utilized[57]

1.3.2 Types of recognition systems

There are many types of recognition systems, including:

1.3.2.1 Face recognition

Face recognition is a technology that recognizes people based on their facial features. It entails taking a picture or video of a person's face, manipulating the image to separate facial highlights such as the distance between the eyes, the state of the nose, and the ebb and flow of the lips, and comparing these elements to a database of known faces to determine whether there is a match.[46, 55]

Face recognition frameworks utilize mathematics to extract the unique characteristics of a person's face and create a biometric format, which is a numerical representation of the face. The framework then compares this layout to formats stored in a data collection to see if there is a match.[46, 55]

Face recognition technology is utilized in a wide range of applications, such as security, biometrics, and marketing. It may, for example, be used to unlock cellphones, safeguard building access, and monitor customer behavior.

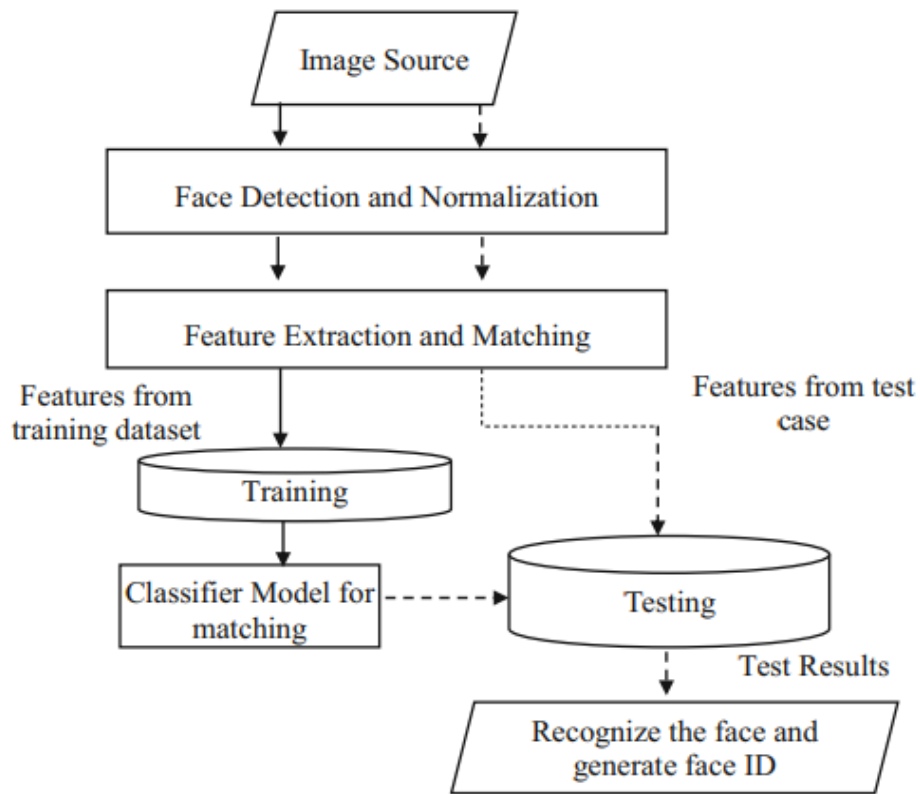


Figure 1.1: Block diagram of the human face recognition system [46]

1.3.2.2 Voice recognition

Voice recognition, also known as speech recognition, is a technological advancement that allows a machine to recognize and comprehend spoken communication. It entails capturing sound information of a person speaking, processing the sound information to remove significant speech instances, then comparing these examples to a data collection of realized speech examples to determine whether a match exists. [93]

Voice recognition frameworks study the exceptional aspects of an individual's voice, such as mood, pitch, and sound, and create a biometric layout, which is a numerical representation of the speech. The framework then compares this layout to formats saved in a data collection to determine whether there is a match.[76]

Security, accessibility, and natural language processing are just a few of the uses for voice recognition technology. It may, for example, be used to unlock cellphones, give hands-free device control, and transcribe voice into text.

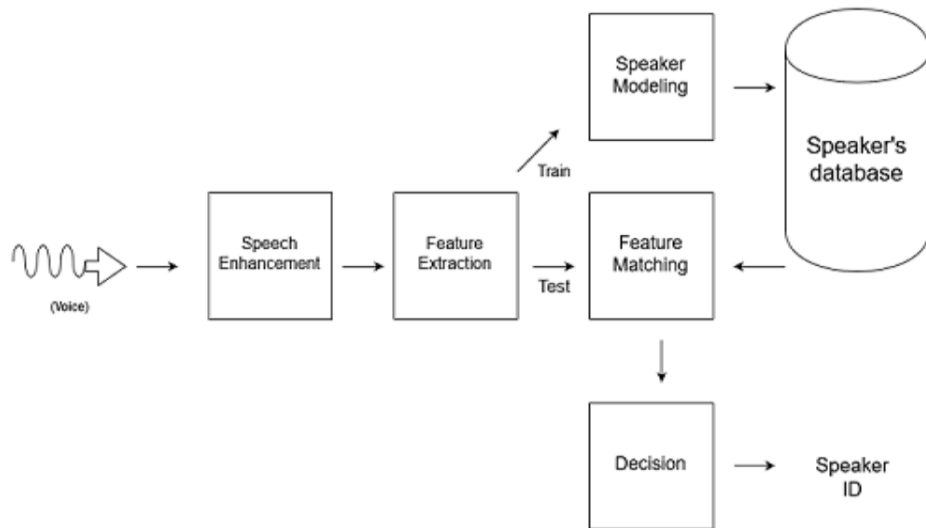


Figure 1.2: Voice recognition process
[Dif]

1.3.2.3 Speech recognition

Speech recognition is a breakthrough that allows a machine to interpret spoken language into text. It entails recording sound information from a person speaking, processing the sound information to isolate relevant speech designs, and converting the speech designs into prepared text.[85]

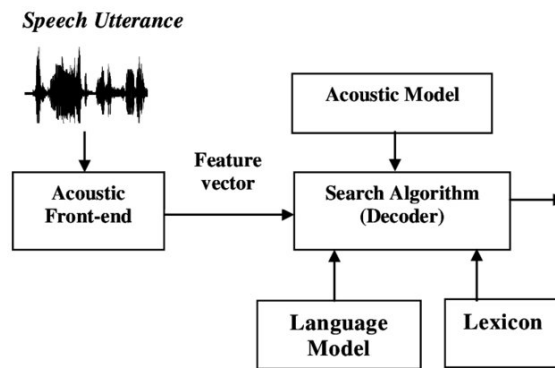


Figure 1.3: Speech Recognition process
[85]

Calculations are used by speech recognition frameworks to study the sound of speech and detect the words being orally spoken. The framework then use language models to determine the most likely record of the speech based on the words that it detects.

1.3.2.4 Fingerprint recognition

Fingerprint recognition is a technology that uses fingerprints to identify persons. It entails taking a photograph of an individual's fingerprint, manipulating the photograph to isolate unique highlights of the fingerprint, then comparing these aspects to a data database of existing fingerprints to see whether there is a match.[12]

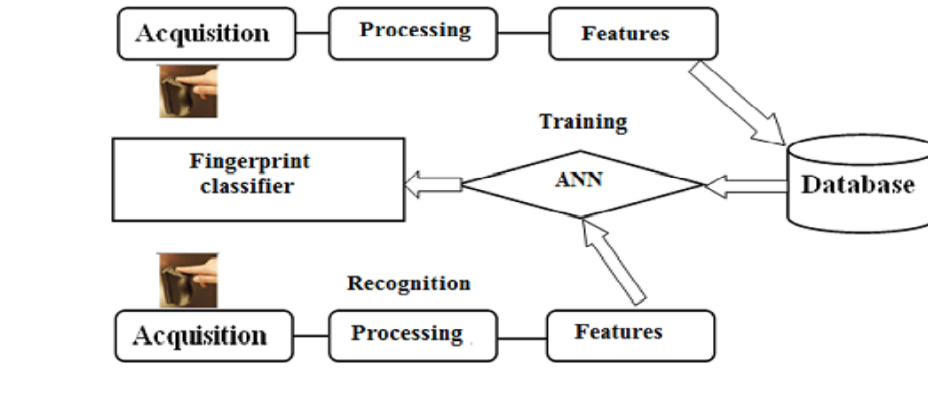


Figure 1.4: Process of fingerprint identifications
[48]

Fingerprint recognition frameworks utilize computations to deconstruct an individual's fingerprint's edge designs, details focuses, and other particular highlights and create a biometric format, which is a numerical representation of the fingerprint. The framework then compares this format to layouts stored in a data collection to see if there is a match.[48]

Security, biometrics, and law enforcement all make use of fingerprint recognition technologies. It may be used to unlock devices, safeguard building access, and identify criminal suspects, among other things.

1.3.2.5 Iris recognition

Iris identification is a breakthrough that allows persons to be identified based on their iris patterns. It entails taking a picture of an individual's iris, manipulating the picture to extract distinctive iris highlights, then comparing these features to a data collection of realized iris instances to see whether there is a match.[26]

Iris recognition frameworks employ computations to assess the surface, diversity, and other noteworthy features of a person's iris and create a biometric layout, which is a numerical representation of the iris. The framework then compares this layout to formats saved in a data collection to determine whether there is a match.[106]

The technique of iris recognition is utilized in a range of applications, including security, biometrics, and law enforcement. It can, for example, be used to safeguard building access, identify criminal

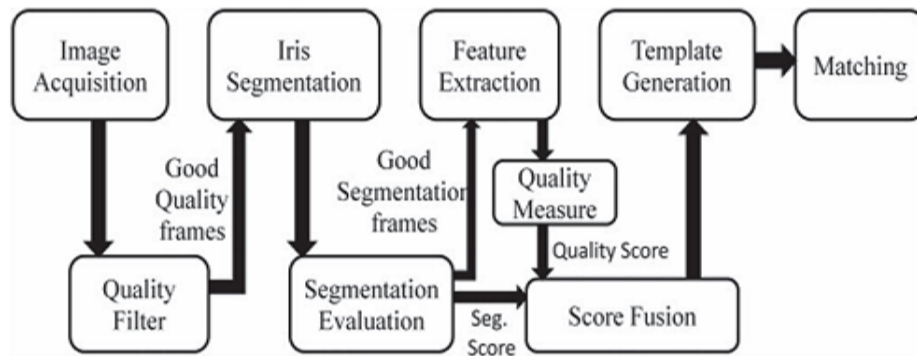


Figure 1.5: Iris recognition system
[106]

suspects, and verify individuals' identities for government benefits.

1.3.2.6 Handwriting recognition

Handwriting recognition is the ability of a framework to recognize human manually written input. Handwriting can come from a variety of sources, including paper archives, photographs, contact screens, and various devices. Handwriting entered by examining is considered unconnected, but handwriting entered with a pen tip is considered on the web.[13]

1.3.2.7 Object recognition

An invention called object recognition is used to identify and categorize items in computer-generated images or recordings. It involves taking a photo or video, processing the data to extract information about the items in the frame, and comparing that information to a database of known things to determine whether there is a match.[43]

Calculations are used by object recognition frameworks to analyze the form, variety, surface, and other distinctive characteristics of objects in a photo or video and create a portrayal of the items. The framework then determines if there is a match by comparing this portrayal to representations stored in a data collection.[83]

1.3.2.8 Signature recognition

With the use of technology, people may be recognized only by looking at their signatures. It entails photographing a person's signature, processing the picture to extract the signature's distinctive traits, then comparing those features to a database of previously-known signatures to see whether there is a match.[31]

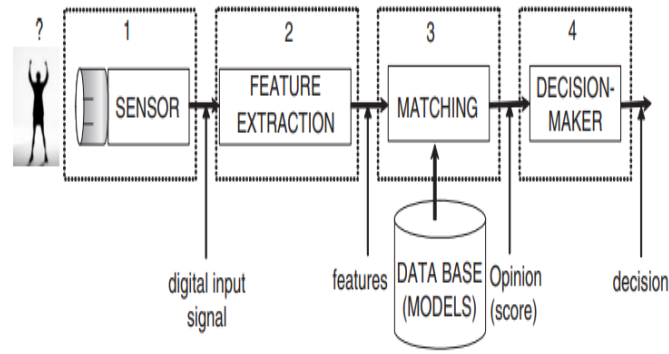


Figure 1.6: General scheme of a biometric recognition system
[31]

In order to produce a biometric template, which is a mathematical representation of the signature, signature recognition systems employ algorithms to examine the size, shape, and style of the letters. The system then determines if there is a match by comparing this template with templates kept in a database.

Signature recognition innovation is utilized in different applications, including security, biometrics, and signature check. For instance, it tends to be utilized to get building access, recognize criminal suspects, and check the validness of signatures on authoritative records.

1.3.2.9 Barcode recognition

Barcode recognition is a technology that uses barcodes to identify products and other objects. A barcode is a visual representation of information, often consisting of a series of high contrast lines and spaces that may be studied and read using barcode recognition systems.

Image examination and example recognition computations are used by barcode recognition frameworks to dissect a picture of a barcode and determine the information it addresses. The framework interprets the barcode by identifying the instances of lines and spaces and then converting the information into a digital design that PCs can store, manage, or analyze.

Each type of recognition system uses different technologies and algorithms to achieve its purpose, and their accuracy and reliability can vary greatly depending on the specific application and environment in which they are used.

1.3.2.10 OCR (Optical Character Recognition)

An innovation known as optical character recognition (OCR) is used to turn scanned pictures, photographs, or PDFs into text that computers can read. It entails inspecting a picture or report to detect the characters and numbers present, then transforming them into complex communication that can be

edited, viewed, or decoded by PCs.[67]

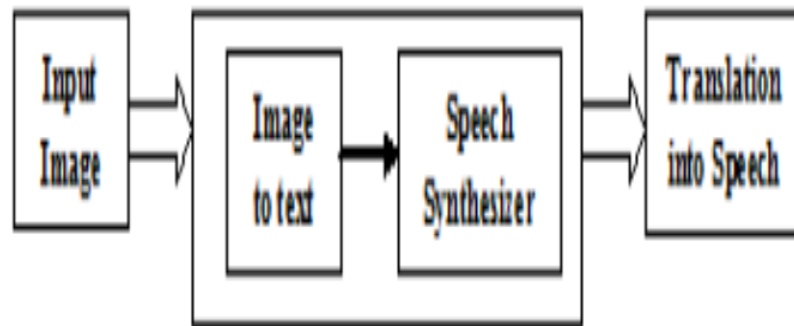


Figure 1.7: Block Diagram of OCR
[67]

OCR frameworks use calculations to examine the structure and form of the characters in an image in order to identify each one. They next compare their findings to a database of known characteristics. The frameworks then transform the characters into complex text.[67]

1.4 Sign language Recognition

1.4.1 Definitions

Sign language recognition refers to the process of using technology to recognize and interpret sign languages used by deaf and hard-of-hearing people. Algorithms and systems that can properly identify sign language motions and translate them into text or voice are being developed to help deaf individuals interact with hearing people and technology.[30]

1.4.2 Motivation

Sign language recognition uses a variety of technologies, including computer vision, machine learning, and natural language processing. The goal is to develop systems capable of reliably identifying the specific qualities of sign languages, such as hand gestures, facial expressions, and body language, and converting these aspects into a form that computers and hearing people can understand.

Sign language recognition is a critical field of study and development because it has the potential to improve deaf people's communication and accessibility, as well as their knowledge and enjoyment of sign languages and deaf culture. The use of sign language recognition technology is expected to become more popular in the future years as technology advances and becomes more widely available.

1.4.3 Problem Area of SLR

In the subject of sign language recognition, there are various obstacles and constraints, such as:

- Segmentation
- Unrestricted environment
- Size of dictionary or Limited training data
- Invariance
- Variety
- Motion gestures
- Feature Selection
- Feature Extraction
- User engagement and usability

1.4.3.1 Segmentation

There are numerous issues with hand segmentation for motion recognition. Two broad methodologies are used for segmentation. The first is based on skin color, while the second makes use of outside assistance[53]

(A) Skin color

Many experts have utilized skin variety based segmentation. Yet, it has numerous intrinsic issues that adversely influence the exactness of the segmentation. These issues are:

- Illumination variation
- Background colors
- Area of Interest segregation
- Wide range of skin colors

It is very difficult to accommodate variety in brightness for precise segmentation. Similar to when there is confusing foundation, there is a risk for false recognition of items near to human skin tone other than hands, arms, or face. Accurately isolating the region of interest is difficult in these situations. It is difficult to fully computerize the problem since a variety of different items might be picked as objects of interest. The fact that there is a wide range of skin tones makes it difficult to choose the right reach for human skin tone, which is another problem with the skin variation based segmentation.[53]

(B) External Aid

Skin variety-based segmentation techniques have a number of disadvantages. Such a big number of professionals have decided to use outside assistance to solve the issues associated with skin variety-based approaches. Two different types of outside guidance are utilized by researchers:

- Data gloves
- Color gloves

This aid has simplified the segmentation process, but many people don't really value it. Because it isn't respected to first train oneself to use a certain framework, and it minimizes its general material.[53]

1.4.3.2 Unrestricted environment

Researchers often impose restrictions on signers environment due to issues with segmentation, feature selection, and feature extraction. These limitations may include

- Signer's Background
- Signer's Clothing
- Preprocessed images

For instance, researchers may restrict the background color to simplify segmentation and feature extraction, as well as impose dressing requirements, such as half sleeves and specific colors. In some cases, researchers may manually crop or alter signer images to address classification challenges.[53]

1.4.3.3 Size of dictionary or Limited training data

The dictionary size used in current research studies is often limited. Most studies have only utilized a small set of static one-handed alphabet signs from a specific language. Only a limited number of eight to ten phrases are used for continuous sign recognition purposes. The practicality and usefulness of the research is limited until a larger sign dictionary is employed.[53]

1.4.3.4 Invariance

Many researchers encounter the issue of feature selection lacking orientation invariance and being affected by variations in illumination. Thus, a robust sign recognition system must exhibit invariance[53] with respect to the following factors:

- Rotation

- Translation
- Scaling
- Illumination

1.4.3.5 Variety

Sign languages use a wide range of gestures, such as one-handed signals, two-handed motions, arm positions, facial expressions, eye gazing, head and body postures, and so on. Nonetheless, incorporating all of these gestures might be difficult for researchers, who frequently work with a restricted number of gestures. One-handed or two-handed signals are typically used because selecting an acceptable feature set gets increasingly challenging as the lexicon of gesture types grows bigger.[53]

1.4.3.6 Motion gestures

The majority of research on sign language recognition has been on static signs. Only a few research on dynamic gestures have been performed. Because of the restricted forms of gestures utilized and the limited dictionaries, research on motion gestures has limited relevance. The following are some of the difficulties related with motion gestures:

- Start/ end identification of gesture sequences
- Motion identification
- Isolated sign recognition

Motion gestures are a series of signs, making it critical and difficult to identify individual signs within the sequence, recognize the signer's movements, and determine the start and end of isolated gestures and complete phrases. It is crucial to focus research on movement segmentation[53]

1.4.3.7 Feature Selection

Researchers encounter obstacles such as :

- reducing the feature set sequences
- picking robust features
- basing features on local information

throughout the feature selection process. These are some of the most pressing difficulties at this critical stage of sign language recognition systems.[53]

1.4.3.8 Feature Extraction

After selecting features, the next step is to extract them from the signer's image. However, this process must overcome challenges such as :

- complex backgrounds
- use of colored or data gloves

Complex backgrounds can make accurate feature extraction difficult. To facilitate the process, some researchers have used gloves, but this is not necessary for the system's general applicability.[53]

1.4.3.9 User engagement and usability

Communication via gestures acknowledgment frameworks should have a clear and easy to use plan and should be fit for adjusting to the singular client's communication via gestures style.[53]

1.4.4 Types of sign language recognition

There are several types of SLR, including:

1.4.4.1 Hand gesture recognition

The major focus of this sort of SLR is the recognition of sign language hand motions. Tracking hand motions and assessing the form and velocity of the hands is required to recognize certain indicators.[88]

1.4.4.2 Face and body posture recognition

In addition to hand motions, sign languages transmit meaning through face expressions and body position. This type of sign language identification includes evaluating how the face and body move and are positioned in order to understand signs.[45]

1.4.4.3 Sign language translation

Real-time sign language to speech or text translation is a feature of this type of SLR. It could be managed with SLR programs or gesture interpreters.[47]

1.4.4.4 Sign language tutoring systems

These SLR-based solutions enable deaf persons to acquire new sign language skills and improve their existing sign competence. Users may learn to sign properly and naturally with the aid of the interactive

activities and feedback they provide.

1.4.4.5 Assistive technologies for deaf individuals

SLR technology may also be used to create sign language enabled gadgets and apps, as well as voice to sign language translation systems and speech to sign language translation systems.

The following is an overview of some of the most common SLR types, which may be used in a variety of ways thanks to this technique and methodology.

1.4.5 Sign Language Recognition System

A SLR (Sign Language Recognition) system is a computer-based system that is designed to recognize and interpret sign language gestures and movements. The goal of a SLR (Sign Language Recognition) system is to translate sign language into text or speech, or to control other devices or machines based on the recognized sign language.[87]

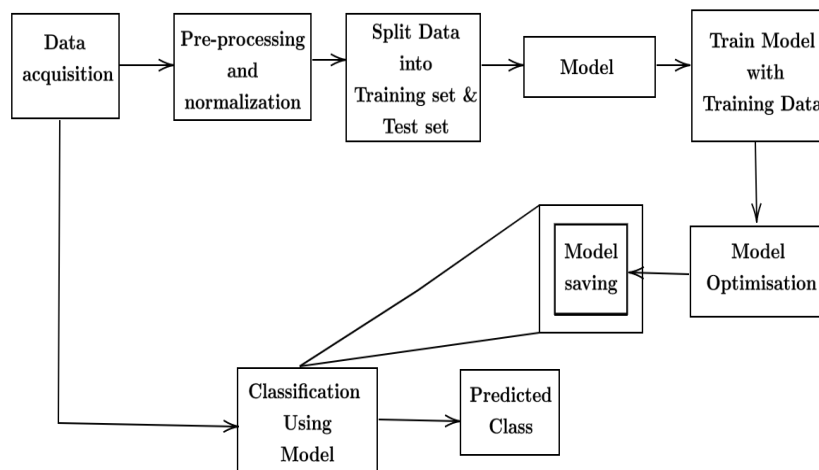


Figure 1.8: Block Diagram of SLR

An overview of a SLR (Sign Language Recognition) system typically includes the following components:

1.4.5.1 Data acquisition

Acquiring sign language data is the first step in creating a SLR (Sign Language Recognition) system. In developing a standard database, various researchers utilize a series of digital cameras/video cameras or sensor with varying positions/places before the item. They also used various lighting illuminations, background choices, and other equipment such as hats/caps, gowns, and spectacles to collect data in the

form of static gestures (photographs) or dynamic gestures (videos)[87], Looking at the area, research can be divided into two main categories depending on the signs recognized in these studies.[30] These categories are as follows:

- Static Signs based recognition
- Dynamic Signs based recognition

Other researchers use the same approaches to obtain their own datasets. Some researchers collect gesture datasets using specifically built input devices. Each of them will be discussed in the next section. Various studies employ digital cameras to capture static movements in general (signs).

(A) Digital still camera [77, 81]

(B) Video camera [27, 50, 66]

(C) Specially designed input devices :

A few of the researchers additionally utilize specifically designed data acquisition devices to gather input signals. The list of input devices is as follows:

- CyberGlove®[54, 62]
- Sensor glove [21]
- Polhemus FASTRAK [MANUAL]

1.4.5.2 Feature Extraction

In computer vision applications, feature extraction is a critical step in processing visual information. It involves identifying and extracting meaningful information from images or videos, which can then be used for tasks such as object detection, tracking, and recognition. There are several methods and techniques for feature extraction, each with its own strengths and limitations.

In this Table 1.1, we will explore some of the most common methods for feature extraction, including those that use digital still cameras, specially designed devices, and video cameras. For each method, we will provide a brief description and reference to relevant literature.

Feature extraction using digital still camera		
Method	Description	ref
Contour	extract relevant information from the edges of an object or shape in an image. This involves identifying the contours and using mathematical algorithms to extract information about the object's shape, size, and orientation.	[82]

Hidden Markov Model	is a statistical model that represents a system with an unobserved state that evolves over time, and produces observations that are dependent on the current state	[69]
Hu moments, Gabor wavelets, Scale Invariant feature Transform	Hu moments are a set of image moments that are used to describe the shape of an object in an image. Gabor wavelets are a type of wavelet transform that is commonly used in image analysis to capture both spatial and frequency information. Scale Invariant Feature Transform (SIFT) is an algorithm used to detect and describe local features in an image that are invariant to scale, rotation, and translation.	[59, 77]
Elliptical Fourier representation	In complicated deformable scenarios, the usage of an Elliptical Fourier representation of a hand might be quite effective. The suggested technique has been validated using both synthetic and actual data.	[81]
Thirty features used	As indicated by the variety layers extended from the portioned variety areas. Thirty highlights are separated and gathered in one vector that addresses a single signal. Every vector comprises of thirty passages; fifteen sections to communicate the points between the actual fingertips and among them and the wrist, furthermore, one more fifteen sections that represent distances among fingertips; and among fingertips and the wrist. Then there was a need to utilize a bunch technique to decide the best district to communicate the variety region. Indeed, even in the instance of missing variety for some explanation, like commotion or one fingertip has been covered by another variety, this case has been settled by picking the default position in the picture	[64]
Haar wavelets transform	is a numerical tool used to analyze signals and extract useful information from them. It entails converting a sign or image into a series of wavelets, which are little wave-like capabilities that address the sign's behaviour at various scales and frequencies.	[89]
Eight features	The size, circumference, length of two axes of the ellipse to match the gesture region, and their derivatives are eight properties that the authors derive from these gestures.	[105]

2D DWT HAAR wavelet	It involves decomposing a 2D signal or image into a set of wavelets, which are small wave-like functions that represent the signal's behavior at different scales and frequencies. The Haar wavelet is a specific type of wavelet used in the decomposition process	[52]
Feature extraction using specially designed devices		
Fourier Analysis	It involves expressing a signal or function as a sum of sine and cosine waves with different frequencies, phases, and amplitudes. This decomposition allows us to understand the frequency content of a signal or function, as well as to identify and remove unwanted noise or interference.	[54]
Feature extraction using video camera		
Contour mapping	Centroids, Finite State Machine, Canny Edge Detection were used for feature extraction.	[56]
Contour Mapping VOP	Video Article Plane (VOP) age was utilized to remove highlights from video outlines. Between outline change location calculation was utilized for removing the VOP utilizing shape planning.	[84]
Points of Interest, Fourier Descriptor	Extracted characteristics were separated into two categories: hand form and hand movement. The Point of Interest (POI) of the hands hand gesture properties were employed. The feature vector has 55 features.	[66]
GFD/GCD	GFD (fractal dimension) stands for "Gradient Feature Descriptor", which is a method used for shape analysis that quantifies the complexity of the shape by estimating its fractal dimension. GCD (global color distribution) refers to a method of analyzing color information in digital images by computing the frequency of occurrence of each color in the image, providing a statistical summary of the overall color distribution.	[73]
Orientation his-togram	The properties of translational invariance were extracted.	[70]
Histogram	The clearest image was identified using histogram analysis.	[64]

PCA-Hand features without distortion	The proposed method of Visual Speaker Alignment (VSA) significantly enhances the performance of appearance-based frame features. While the use of additional training data through Virtual Training Samples (VTS) may not always improve results, combining both methods produces the best reported outcome for this data in literature. The output of multiple recognition systems can be easily combined by "rovering" over the system outputs using word confidences of the recognizer output. Four different systems were combined, including sliding window over PCA-Frames, sliding window over PCA-Frames and hand trajectory (HT) features, sliding window over PCA-Frames and hand velocity (HV) features, and appearance-based PCA-hand patches.	[29]
Dynamic Time Warping (DTW)	A distance measure was defined for sign videos in order to enable the system to identify the most similar matches from the database to a query video.	[16]
Blob extracting	Blob extraction is a technique used in computer vision and image processing to identify and isolate connected regions of an image that share a common set of properties, such as color or texture. The resulting "blobs" can be further analyzed and used for a variety of tasks, such as object recognition, tracking, and segmentation.	[29]
The accumulated prediction errors or image differences (AD)	The proposed feature extraction schemes aim to remove the temporal dimension of video-based gestures by utilizing weighted and directional accumulated prediction errors. The input video is converted into a single absolute AD image that represents the motion in the video sequence. However, different sign gestures may result in very similar AD images. To address this, the authors suggested using weighted directional accumulated image differences, where the weighting determines how prediction errors are accumulated into a single AD image. These image differences are further categorized into positive and negative accumulated prediction errors (AD+ and AD-).	[90]
Four features were used	Data on projection, The number of visible fingers and the multi-frame features, CSS functions that are embedded, Edge orientation histograms embedded	[96]

Normalized Moment of Inertia (NMI)	used to quantify the distribution of intensity in an image. It is calculated by computing the normalized second moments of the image intensity, which reflect the shape and orientation of the image. NMI is invariant to translation, rotation, and scale, making it a useful tool for object recognition and tracking.	[50]
Graph parsing	The study used the Indexed Edge Unambiguous –PSL (IE-PSL) graph set to represent hand position as a formal language generated with the ETPL graph grammar. Compared to a standard (string) grammar, a graphic grammar includes an embedding transformation in its output. This allows the path on the right to be integrated into the transformed path during the derivation step.	[32]
Local Linear Embedding	Local Linear Embedding (LLE) is a non-linear dimensionality reduction technique that maps high-dimensional data into a lower-dimensional space while preserving the underlying structure of the data. LLE does this by modeling the relationship between neighboring points and representing the data as a weighted graph. The lower-dimensional embedding is then computed by finding the set of weights that preserves the geometry of the high-dimensional data.	[103]
Kalman Filter	The extracted features for sign recognition were limited to the hand motion, shape, and position. The video was first processed for hand and face detection and segmentation. Sign features were then extracted based on the manual signs including hand motion, hand shape, and hand position with respect to the face. For hand motion analysis, a Kalman filter was applied to the center of mass (CoM) of each hand. The features included the width, height, and orientation parameters of an ellipse, as well as seven Hu moments calculated on the binary hand image. The hand position features were based on the normalized horizontal and vertical distances between the hand center and the face center. The feature vector had a dimensionality of 32 per frame, with four hand motion, 10 hand shape, and two hand position features for each hand.	[15]

Table 1.1: Feature extraction methods employed by various researchers

1.4.5.3 Recognition

Researchers have developed various categorization strategies to tackle this problem, ranging from appearance-based approaches that extract visual features from sign images, to motion-based approaches that analyze the dynamics of sign motions, to hybrid approaches that combine both appearance and motion features to achieve higher accuracy. Additionally, some approaches rely on linguistic information to recognize signs within the context of a sentence or conversation.

In this Table 1.2, we will discuss some of the most common categorization strategies employed by researchers to detect sign language motions, along with relevant references to the literature.

Method	Description	ref
Neural Network	A verity of neural network classifiers are used	[52, 64, 66, 69, 89, 98]
SVM	A Support Vector Machine classification	[77]
HMM	A HMM classifier with its variants	[15, 18, 27, 44, 56, 62, 105]
Fuzzy sets	Fuzzy Sets with other classifiers used	[21]
Tensor analysis	Tensor based classification proposed.	[78]
FSM and DTW	Finite State Machine and Dynamic Time Wrapping Algorithms.	[84]
ROVER	Recogniser output voting error reduction	[29]
Euclidean distance classifier with GCD	GCD features of hand shapes in key frames and Euclidean distance classifier.	[73]
I, II order Polynomial Classifier	User-independent classification of proposed solution using KNN, 1st order polynomial classifier and 2nd order polynomial classifier.	[90]
CAMSHIFT Algorithm	Continuous Adaptive Mean SHIFT Algorithm	[70]
HSBN	A Handshapes Bayesian Network	[95]
BoostMap	A binary classifier and boosting method Ada Boost is employed for embedding optimization. This approach was adopted in Boost Map method.	[16]
SVR	A Support Vector Regression technique	[96]
VQPCA	A Vector Quantization Principal Component Analysis	[54]

Table 1.2: Classification methods used by researchers

1.4.6 SLR using Deep learning

Deep learning is a branch of machine learning that includes training multiple-layer artificial neural networks to learn from and predict complex input such as pictures, audio, and text. The networks are meant to extract key elements from data automatically, allowing for more accurate and complex predictions than typical machine learning models. Deep learning has resulted in substantial advances in computer vision, natural language processing, and speech recognition.[9]

1.4.6.1 Artificial neural networks (ANN)

ANN (Artificial neural network) are a sort of computing model that mimics how the human brain processes information. They are made up of layers of linked nodes called "neurons" that process and transfer data. The input layer receives data, which is then processed through a succession of hidden layers, each of which generates an output by applying a set of weights to the input data. The prediction or classification result is produced by the final output layer.[9]

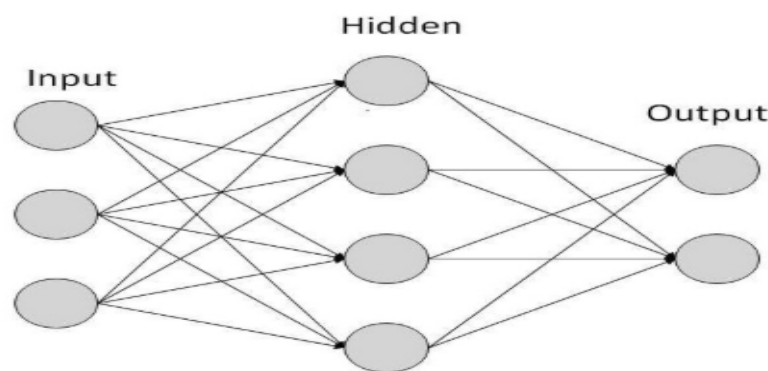


Figure 1.9: A typical artificial neural network (ANN).

[10]

Image and audio recognition, NLP (Natural Language Processing), and anomaly detection are among activities that ANNs are utilized for. Backpropagation is a method that modifies the weights between neurons depending on the difference between the expected and actual output, allowing the network to learn and improve over time.

1.4.6.2 Deep Neural Network (DNN)

DNN (Deep Neural Networks) are ANN (Artificial neural network) that include numerous layers between the input and output layers. These hidden layers enable the network to learn increasingly complicated characteristics and patterns from the incoming data.[9]

DNNs are used in many different applications, such as computer vision, natural language processing,

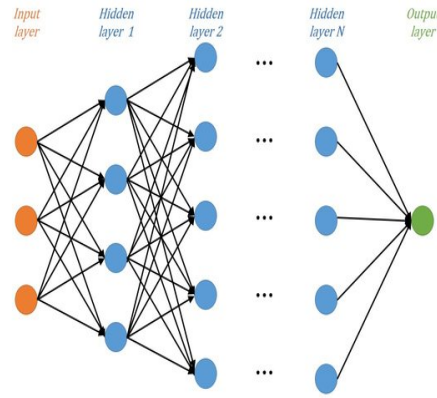


Figure 1.10: Construction of the deep neural network (DNN) model.
[61]

and speech recognition. They are often trained by backpropagation, a method in which the loss in the network's output is transmitted backwards through the layers, allowing the weights of the neurons to be modified to increase prediction accuracy.[61]

DNNs can be difficult to train due to the enormous quantity of data and processing resources required to understand the complex features and patterns in the data. They have, however, been demonstrated to be extremely good at tasks like as picture categorization, speech recognition, and language translation.[61]

The following parts are the popularly used and they represent deep neural networks type

- MLP (Multi-Layer Perceptron)
- RNN (Recurrent Neural Network)
- CNN (Convolutional Neural Network)

1.4.6.3 Multilayer Perceptron (MLP)

A MLP (Multi-Layer Perceptron) is a feedforward artificial neural network with many layers of linked neurons, comprising an input layer, one or more hidden layers, and an output layer. Each neuron in an MLP gets input from the previous layer and adds a set of weights to it before passing it through an activation function to generate an output.[9]

The hidden layers in an MLP allow the network to learn increasingly complex representations of the input data, while the output layer produces the final prediction or classification result. MLPs are trained using backpropagation, where the error between the predicted output and the actual output is propagated backwards through the network, allowing the weights of the neurons to be adjusted to improve the accuracy of the predictions.[9]

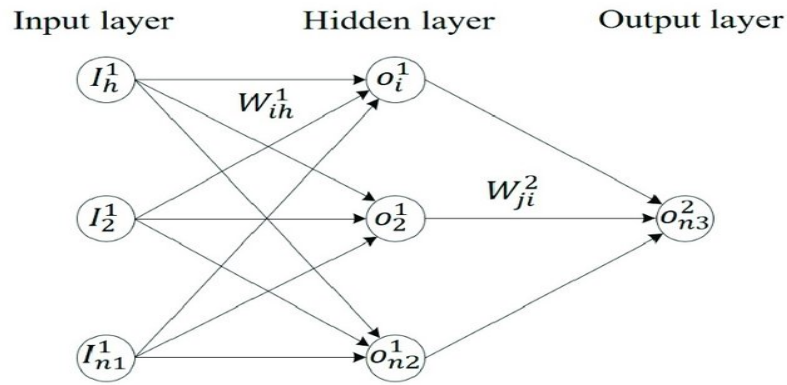


Figure 1.11: Structure of a multilayer perceptron (MLP) algorithm [22]

1.4.6.4 Recurrent Neural Networks (RNN)

RNNs are a sort of ANN (Artificial neural network) that can handle sequential data such as time series data, audio, and text. RNNs, as opposed to feedforward neural networks, feature a loop that allows information to be transmitted from one time step to the next, allowing them to use knowledge from previous inputs to guide the processing of current inputs.[75]

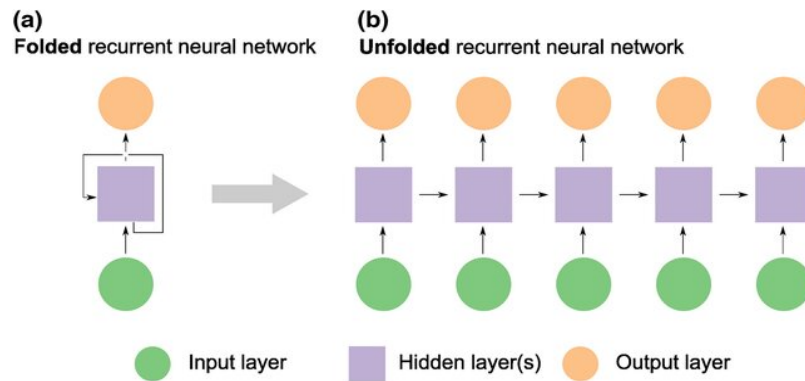


Figure 1.12: Schematic architecture of a recurrent neural network [75]

RNNs are especially effective in applications like speech recognition, natural language processing, and picture captioning, where context and past inputs are critical for interpreting the present input. LSTM (Long short-term memory) and GRU (Gated recurrent unit) RNNs are two prominent forms of RNNs that solve the vanishing gradient problem that can arise while training RNNs with long data sequences.[75]

RNNs are trained by backpropagation over time, which propagates the mistake across the full sequence of inputs, allowing the weights of the neurons to be modified to increase prediction accuracy.[75]

(A) **Long short-term memory (LSTM)**

LSTM (Long short-term memory) is a form of RNN (Recurrent Neural Network) developed to overcome the issue of disappearing gradients during training. LSTMs employ a memory cell, which can hold data for long periods of time and selectively forget or update it as needed. This enables LSTMs to handle long-term dependencies in sequential data well, which is useful in natural language processing and speech recognition.[28]

The architecture of LSTMs is complicated, with many gating mechanisms controlling the flow of information into and out of the memory cell. The input gate regulates the flow of fresh information into the cell, the forget gate regulates which information should be discarded from the cell, and the output gate regulates the flow of information from the cell to the output layer.[28]

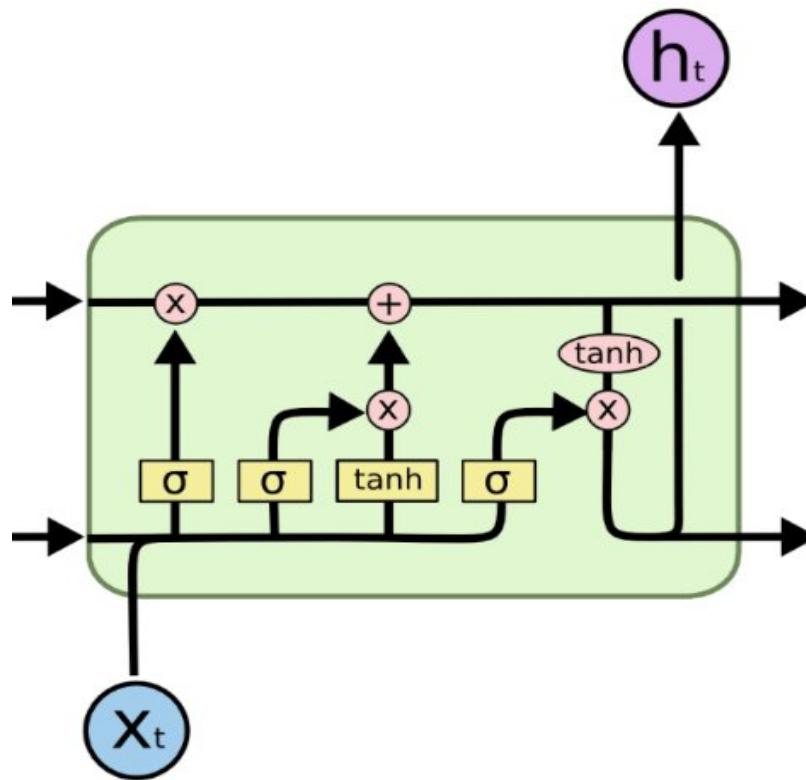


Figure 1.13: Schematic architecture of a LSTM
[28]

LSTMs have been demonstrated to be extremely successful in a variety of applications, including speech recognition, natural language processing, and music production.

(B) **gated recurrent unit (GRU)**

The GRU (Gated recurrent unit) is a form of RNN (Recurrent Neural Network) that, like the LSTM (Long short-term memory) network, is intended to overcome the issue of disappearing gradients during training. The GRU, like the LSTM, features a gating mechanism that enables it to selectively store or discard information from previous time steps.[6]

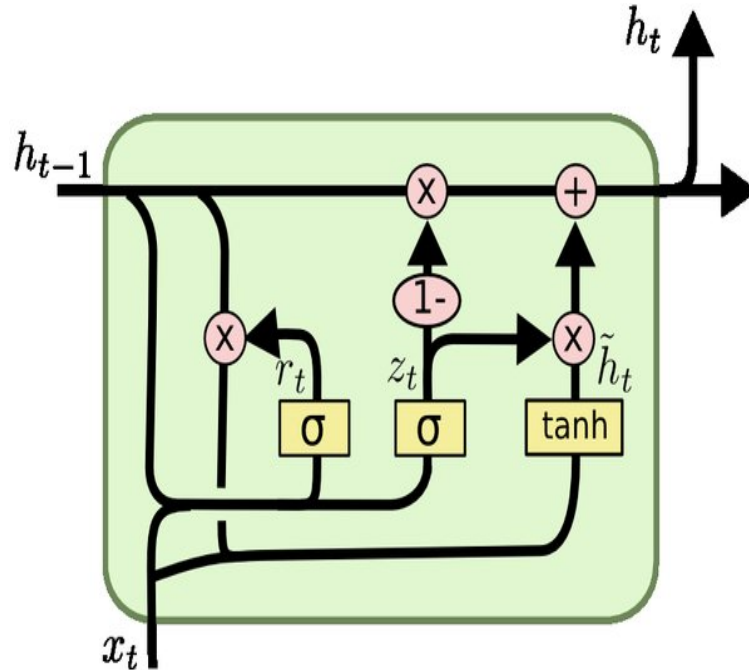


Figure 1.14: Schematic architecture of a GRU
[6]

The GRU has two gates that govern the flow of information into and out of the network: a reset gate and an update gate. The reset gate regulates how much previous data should be destroyed, whereas the update gate specifies how much fresh data should be preserved.[6]

The GRU can be trained more quickly and is less likely to overfit than the LSTM since it has fewer parameters than the latter. In tasks like speech recognition, natural language processing, and machine translation, it has been demonstrated to be quite successful.[6]

Backpropagation over time is used to train GRUs, which allows the network to learn and get better over time by adjusting the weights of the neurons depending on the difference between the expected and actual results.

1.4.6.5 Convolutional Neural Networks (CNN)

CNNs are a sort of ANN (Artificial neural network) that is extensively used for image and video processing. They are intended to handle grid-like inputs, like as pixels in a picture, by employing a sequence of convolutional and pooling layers that gradually learn and extract more complicated characteristics from the input.[11]

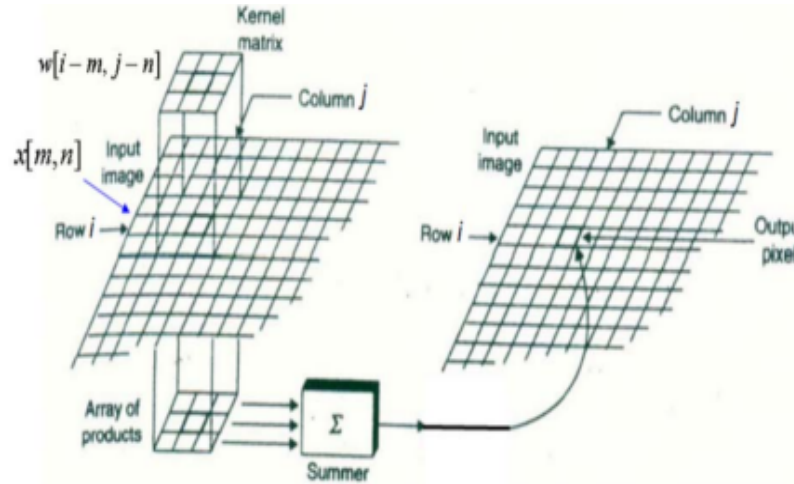


Figure 1.15: Details on Convolution layer
[11]

Conv1D, Conv2D, and Conv3D are types of convolutional layers used in deep learning models, including CNNs.

- **Conv1D:** is a one-dimensional convolutional layer that is used for processing time series data or signals.
- **Conv2D:** is a two-dimensional convolutional layer that is used for processing 2D images.
- **Conv3D:** is a three-dimensional convolutional layer that is used for processing 3D volumetric data, such as medical images or videos.

Each of these convolutional layers has its own specific function and role in deep learning models, and they can be combined with other types of layers such as pooling, dropout, and normalization to form a complete network architecture

The convolutional layers in a CNN apply filters to the input data, which convolve with the input to produce a feature map. This feature map highlights the important spatial patterns in the input.

The pooling layers then downsample the feature map, reducing its size and making it more computationally efficient.

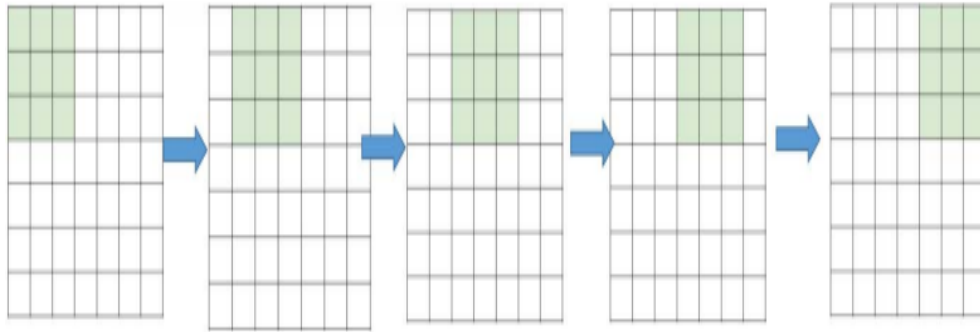


Figure 1.16: Stride 1, the filter window moves only one time for each connection [11]

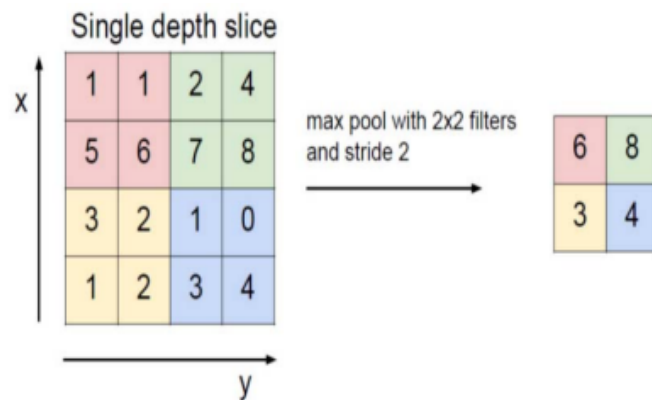


Figure 1.17: The max-pooling with 2x2 filter and stride 2 lead to down-sampling of each 2x2 blocks is mapped to 1 block (pixel).

[11]

CNNs have been demonstrated to be extremely successful in a variety of image and video processing applications, including image classification, object identification, and face recognition. Backpropagation is commonly used to train them, which modifies the weights of the neurons depending on the difference between the expected and actual output, allowing the network to learn and improve over time.[11]

1.4.6.6 Transformers

Transformers are a sort of brain network engineering that is broadly utilized for NLP (Natural Language Processing) undertakings, for example, language interpretation, question responding to, and feeling analysis. Transformers were first presented in the paper "Attention is All You Need" by Vaswani et al. in 2017.[99]

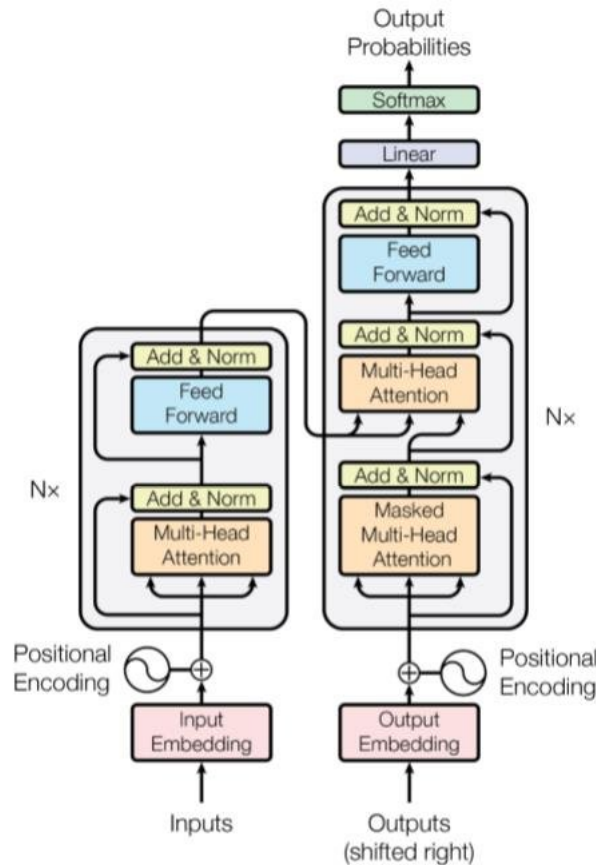


Figure 1.18: The transformer model
[99]

Transformers are based on the idea of self-attention, where the network learns to attend to different parts of the input sequence to compute the output. This allows the network to capture long-range dependencies and perform well on sequential tasks[99]

The architecture of Transformers includes an encoder and decoder network, both comprising multiple layers of self-attention and feedforward neural networks. The encoder processes the input sequence, learning representations of the input suitable for the task through the self-attention mechanism. On the other hand, the decoder generates the output sequence using a combination of self-attention and encoder-decoder attention.[99]

1.5 Related Work

In the field of sign language recognition, there has been a significant amount of research aimed at improving the accuracy and performance of recognition systems. Researchers have explored various feature extraction and classification methods to detect and interpret sign language gestures.

In this Table 1.3, we will summarize some of the related work in the field of sign language recognition, with a focus on the authors, publication years, feature extraction methods, and classification methods used in each study. The goal is to provide an overview of the different approaches that have been used to tackle the sign language recognition problem, and to highlight the strengths and weaknesses of each approach. By understanding the related work in this field, we can identify the gaps and challenges that still need to be addressed, and pave the way for further research in this area.

Author, Publication Year	Feature Extraction Method	Classification Method
Vishwakarma (2017)[100]	Gabor filter	SVM (Support vector machine)
Gao et al. (2017)[35]	CNN	
Bheda and Radpour (2017)[17]	CNN	
Sadeddine et al. (2018)[86]	Hu's invariant moments + LBP + Zernike moments + GF	PNN (Probabilistic neural network)
Zhang et al. (2018)[104]	HOG (Histogram of oriented gradients)+ LBP (local binary pattern)	SVM (Support vector machine)
Gajalakshmi and Sharmila (2019)[34]	CCH (Chain code histogram)	SVM (Support vector machine)
Mittal et al. (2019)[68]	Leap motion sensor	2D-CNN and Modified LSTM
Aparna and Geetha (2019)[14]	CNN and 2layer LSTM	
Liao et al. (2019)[60]	3D- ConvNet with BLSTM	
Ozcan and Basturk (2019)[72]	AlexNet + ABC	
Adithya and Rajesh (2020)[8]	CNN	
Adaloglou et al. (2021)[7]	Inflated 3D ConvNet with BLSTM	
Jiang et al. (2021)[49]	3DCNN with SL-GCN using RGB-D modalities	

Tan et al. (2021)[92]	EDenseNet	
Wang et al. (2021)[101]	MobileNet	Random forest
Sahoo et al. (2022)[88]	AlexNet, VGG16	
Wang et al. (2022)[102]	E-MobileNetv2	
Gadekallu et al. (2022)[33]	CNN + HHO	
Li et al. (2022)[58]	HOG, 9ULBP	SVM (Support vector machine)

Table 1.3: Summary of related works

1.6 Advantages and disadvantages

(A) Advantages

Advantages of sign language recognition (SLR) include:

- **Accessibility:** SLR provides accessibility to the deaf and hard-of-hearing community, enabling them to communicate more easily with hearing individuals.
- **Convenience:** SLR eliminates the need for interpreters, which can be expensive and not always available
- **Real-time communication:** SLR can provide real-time communication, allowing for more efficient and effective conversations.
- **Improved education:** SLR can be used to improve education for deaf and hard-of-hearing students, by providing real-time translation in the classroom.

(B) disadvantages

However, there are also some disadvantages of SLR, including:

- **Complexity:** Sign language is a complex language with a large vocabulary and grammar, which can be challenging to recognize accurately.
- **Limited data:** There is limited data available for training SLR systems, making it challenging to develop accurate models.
- **Variability:** Sign language varies across regions and individuals, making it difficult to develop a universal SLR system.
- **Technological limitations:** Current SLR technology is limited by factors such as camera resolution, lighting, and computational resources.

1.7 Conclusion

The analysis of the literature in this chapter leads to the conclusion that sign language recognition is an increasingly important area of research. The development of effective sign language recognition systems has the potential to significantly enhance communication and accessibility for individuals who are deaf or hard-of-hearing.

However, there are still several challenges that need to be addressed, such as the complexity and variability of sign language, the limited availability of data, and the technological limitations of current systems.

Deep learning has emerged as a promising solution for sign language recognition, with numerous deep learning models proposed to tackle these challenges. Multiple studies have demonstrated that deep learning-based sign language recognition systems outperform traditional machine learning techniques.

Overall, the research presented in this chapter highlights the importance of sign language recognition and the potential of deep learning in addressing the challenges of this field. Further research is needed to develop more accurate and efficient sign language recognition systems that can provide better accessibility for the deaf and hard-of-hearing community.

Conceptual Study

2.1 Introduction

The preceding chapter provided a comprehensive overview of sign language recognition including its most popular systems and the underlying processes involved. Specifically, we delved into the intricacies of sign language recognition systems, detailing how deep learning and neural networks form the backbone of this technology.

This chapter aims to present our proposed design for the sign language recognition system. We will begin by discussing the different steps involved in the sign language recognition process. Following that, we will provide a detailed description of the recognition phase, which utilizes an LSTM model as a key component.

2.2 System design

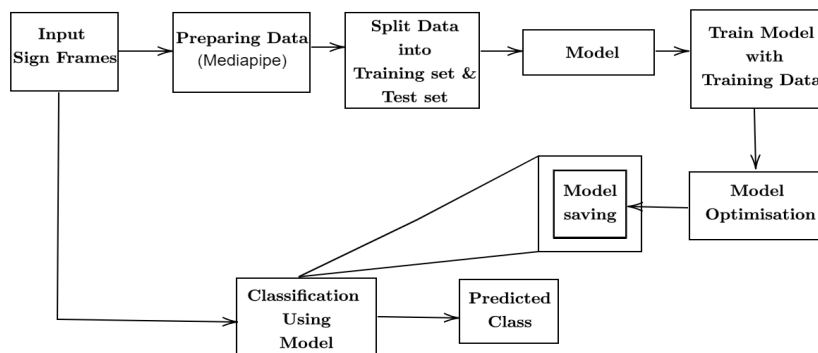


Figure 2.1: Our system architecture

2.2.1 Data Acquisition

For data acquisition, images are captured by webcam using Python and OpenCV

2.2.2 Preparing Data

The use of video data for machine learning tasks often requires preprocessing to extract the relevant information. One common approach is to convert videos into sequences of frames, which can then be used as input for a model. In addition to preprocessing, feature extraction is another crucial step in machine learning. In the context of video data, the extraction of features from each frame can provide useful information about the movement and position of objects within the video.

The MediaPipe library is a popular tool for feature extraction, particularly for tasks involving body and hand tracking. For example, the landmarks of the hands can be extracted from raw videos using the MediaPipe holistic model.[38]

The type of landmark :

(A) Face Mesh :

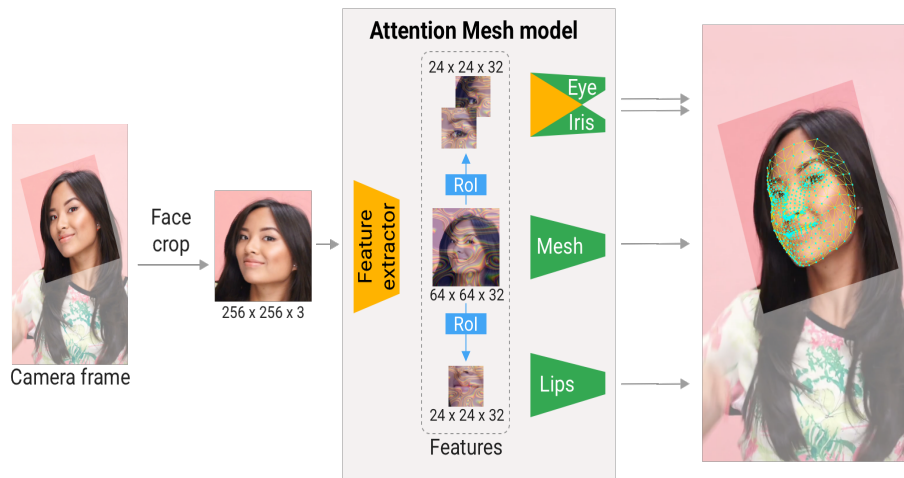


Figure 2.2: Attention Mesh: Overview of model architecture [39]

The MediaPipe Face Mesh solution provides a list of 468 face landmarks, where each landmark is represented by x, y, and z coordinates. The x and y coordinates are normalized to the range of [0.0, 1.0] by the image width and height, respectively. The z coordinate represents the depth of the landmark, where the origin is at the center of the head and the smaller the value, the closer the landmark is to the camera. The magnitude of the z coordinate is scaled roughly the same as the x coordinate.[41]

(B) Pose :

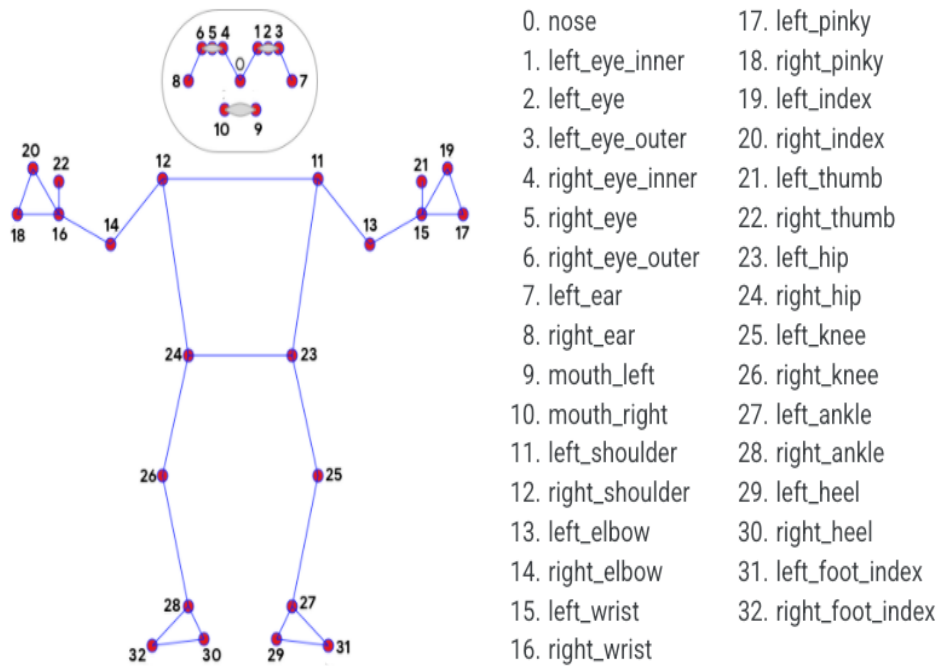


Figure 2.3: 33 pose landmarks.

[42]

The MediaPipe Pose solution provides a list of 33 pose landmarks, where each landmark is defined by its x and y coordinates normalized to the range of [0.0, 1.0] by the image width and height, respectively. The z coordinate is currently not reliable and should be discarded as the model is not yet fully trained to predict depth, although this is something that may be addressed in future updates. Additionally, each landmark has a visibility value in the range of [0.0, 1.0] indicating the probability of the landmark being visible (i.e., present and not occluded) in the image.[41]

(C) **Hand :**

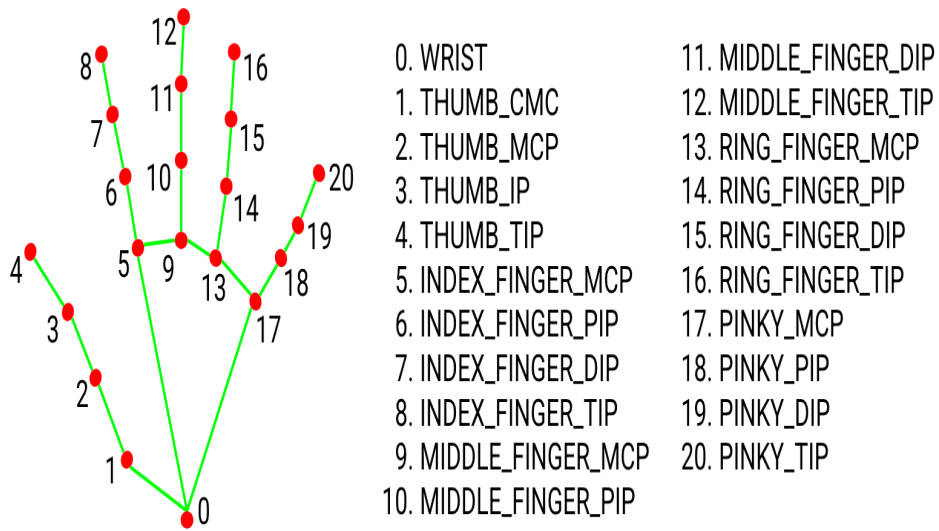


Figure 2.4: 21 hand landmarks.
[40]

The MediaPipe Hands solution provides a list of 21 landmarks on the left hand, where each landmark is represented by its x, y, and z coordinates. The x and y coordinates are normalized to the range of [0.0, 1.0] by the image width and height, respectively. The z coordinate represents the depth of the landmark, where the origin is at the wrist and the smaller the value, the closer the landmark is to the camera. The magnitude of the z coordinate is scaled roughly the same as the x coordinate.[41]

A list of 21 hand landmarks on the right hand, in the same representation as left handlandmarks.[41]

However, it is important to note that not all frames will necessarily contain visible hands or hands that can be detected by the model

2.2.3 Split Data into Train and Test

We use **Holdout method** to split Data ,This involves randomly selecting a portion of the data for training and reserving the rest for testing.

we divide our dataset as 80% for training and 20% for testing.

2.3 Presenting Dataset

Our work involved the creation of the ArSLA28 dataset, which represents Arabic Sign Language alphabets. The dataset contained 28 alphabets, and we utilized different cameras to record the data with the involvement of 13 people, including both men and women. To ensure diversity in our dataset, we used dynamic sign language, which incorporated movements and gestures that are used in natural conversations. For each alphabet, we captured 390 samples in the form of videos, with each video containing 30 frames. We extracted the landmarks from the raw videos using the MediaPipe holistic model. However, not all frames necessarily had visible hands or hands that could be detected by the model. The resulting data was saved in numpy format, with each frame containing 1662 points. These points were divided into 468 face landmarks with 3 dimensions each for x, y, and z, 42 hand landmarks with 3 dimensions each, and 33 pose landmarks including x, y, z, and visibility. This dataset will be a valuable resource for future research in Arabic Sign Language recognition and interpretation.

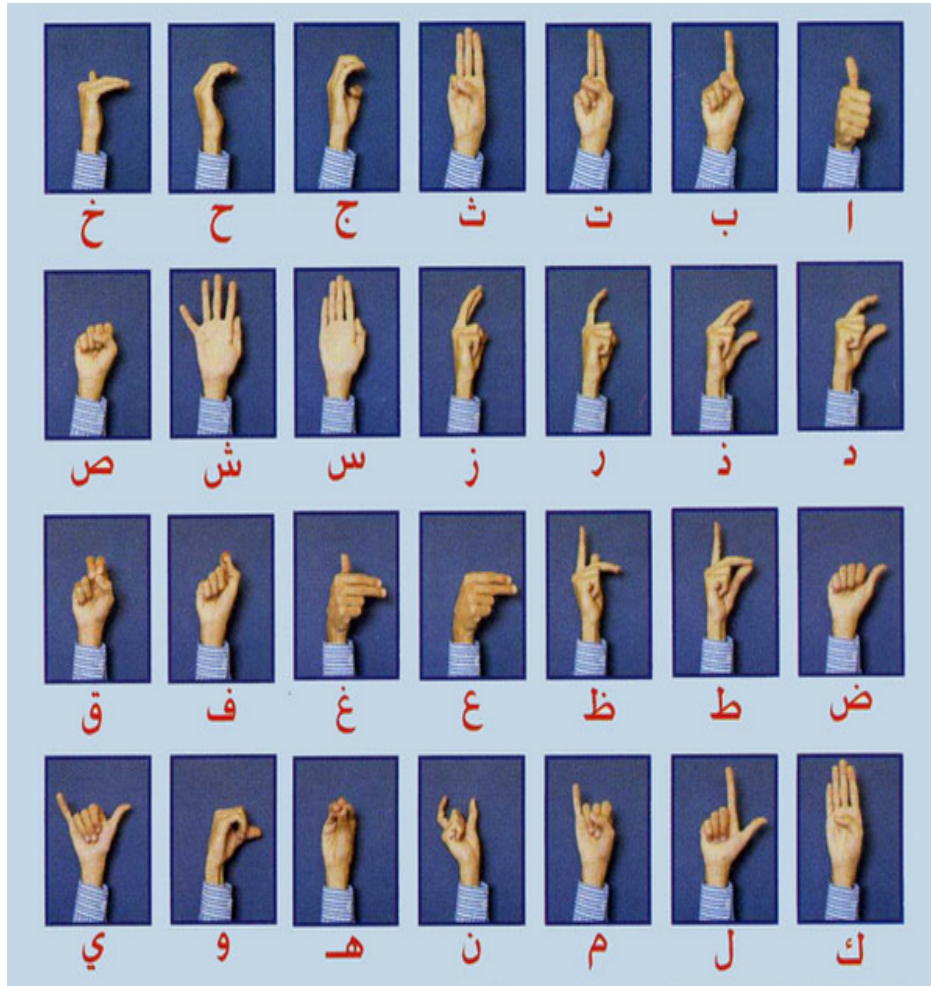


Figure 2.5: Representation of the Arabic sign language alphabets

2.4 Optimizer

In machine learning, an optimizer is an algorithm that is used to adjust the parameters of a model in order to minimize the error or loss function. The loss function represents the difference between the predicted output of the model and the true output, and the optimizer is responsible for finding the set of model parameters that will result in the lowest possible loss.[37]

The choice of optimizer can have a significant impact on the performance of a machine learning model. Different optimizers use different algorithms to search for the optimal set of model parameters, and some may be better suited to certain types of data or model architectures than others.[37]

Some popular optimizers used in machine learning include stochastic gradient descent (SGD), Adam, RMSprop, and Adagrad. These optimizers use various techniques such as adaptive learning rates, momentum, and gradient clipping to improve the efficiency and effectiveness of the optimization process.[37]

1. **SGD** : The stochastic gradient descent (SGD) optimizer update rule[24] can be written as follows:

$$w_t = w_{t-1} - \alpha * g_t$$

In this update rule

- w_t represents the updated weights
- g_t is the gradient of the loss function with respect to the weights
- α is the learning rate

2. **Adadelta** : the Adadelta optimizer uses the following equation to update the parameters:

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon}$$

$$RMS[\Delta x]_t = \sqrt{E[\Delta x^2]_t + \epsilon}$$

$$\Delta x = -\frac{RMS[\Delta x]_{t-1}}{RMS[g]_t} \cdot g_t$$

$$E[g^2]_t = \rho \cdot E[g^2]_{t-1} + (1 - \rho) \cdot g_t^2$$

$$E[\Delta x^2]_t = \rho \cdot E[\Delta x^2]_{t-1} + (1 - \rho) \cdot \Delta x^2$$

$$\theta_t = \theta_{t-1} + \Delta x$$

where:

- $\mathbf{g}$ is the gradient of the objective function with respect to the parameters
- ε is a small constant added for numerical stability
- ρ is a hyperparameter that controls the decay rate of the moving averages
- $E[g^2]_t$ is the exponentially decaying average of the squared gradient
- $E[\Delta x^2]_t$ is the exponentially decaying average of the squared parameter updates
- $RMS[g]_t$ and $RMS[\Delta x]_t$ are the root mean square (RMS) values of $\mathbf{g}$ and Δx respectively at time step t
- θ_t is the vector of parameters at time step t .

The Adadelta optimizer is a variant of the Adagrad optimizer that addresses some of its limitations, such as the need to manually tune the learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients and uses the RMS value of the parameter updates to adaptively adjust the learning rate.[24]

3. **Adafactor** : Adafactor is an optimizer that adapts the learning rate for each parameter, based on the geometry of the gradients. [24]The update rule of Adafactor can be written as:

$$v_t = v_{t-1} + g^2$$

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g$$

$$lr = \min(1.0, (1 - \beta_1) * lr_t / \sqrt{(v_t)})$$

$$w_t = w_{t-1} + lr * m_t$$

where:

- $\mathbf{w}$ is the vector of parameters
- $\mathbf{g}$ is the gradient vector,
- $\mathbf{v}$ is an exponentially weighted moving average of the squared gradients,
- $\mathbf{m}$ is an exponentially weighted moving average of the gradients

- η is the learning rate
- η_t is an initial learning rate
- β_1 is the exponential decay rate for the moving averages step t.
- $\sqrt{(\cdot)}$ function is applied elementwise.

The Adafactor optimizer also includes a few additional features, such as per-parameter scaling and a second-order correction term. However, the above update rule captures the essential idea behind Adafactor.[24]

4. **Adagrad** : the Adagrad optimizer adapts the learning rate for each parameter based on the historical gradient information.[24] The update rule for Adagrad can be written as:

$$g_t = \nabla J(\theta_t)$$

$$r_t = r_{t-1} + g_t^2$$

$$\theta_{t+1} = \theta_t - (\eta / (\sqrt{r_t} + \epsilon)) * g_t$$

where:

- θ is the vector of parameters
- $\nabla J(\theta)$ is the gradient vector of the objective function $J(\cdot)$ with respect to θ
- $\mathbf{r}$ is a vector that accumulates the sum of squared gradients element-wise η is the learning rate, and ϵ is a small constant added for numerical stability to avoid division by zero.

The Adagrad optimizer computes the learning rate for each parameter by dividing the initial learning rate η by the square root of the accumulated squared gradients $\mathbf{r}$. This scaling allows the optimizer to automatically decrease the learning rate for parameters with frequent and large gradients, while increasing it for infrequent and small gradients.[24]

Note that in practice, Adagrad may suffer from the issue of decreasing learning rates over time, which can lead to slow convergence or even premature stopping. To address this, variants of Adagrad such as Adadelta and RMSprop have been developed to dynamically adjust the learning rates.

5. **Adam** : Adam is an optimizer that adapts the learning rate for each parameter based on the first and second moments of the gradients. [24]The update rule for Adam can be written as:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g^2$$

$$m_{t_{hat}} = m_t / (1 - \beta_1^t)$$

$$v_{t_{hat}} = v_t / (1 - \beta_2^t)$$

$$\theta_{t+1} = \theta_t - (\eta / (\sqrt{v_{t_{hat}}} + \epsilon)) * m_{t_{hat}}$$

where:

- θ is the vector of parameters
- g is the gradient vector
- m is the exponentially weighted moving average of the gradient
- v is the exponentially weighted moving average of the squared gradient
- η is the learning rate
- β_1 and β_2 are the exponential decay rates for the moving averages
- t is the current iteration step
- ϵ is a small constant added for numerical stability to avoid division by zero.

The Adam optimizer computes the learning rate for each parameter by dividing the initial learning rate η by the square root of the biased-corrected second moment estimate v_t . The first moment estimate m_t is also bias-corrected to account for its initialization at zero.[24]

Adam is a popular optimizer for deep learning because it can handle sparse gradients, noisy data, and non-stationary objectives. By adapting the learning rates for each parameter based on the first and second moments of the gradients, Adam can converge faster and more reliably than other optimizers like Adagrad and SGD.[24]

6. **Adamax** : Adamax is a variant of the Adam optimizer that replaces the L2 norm of the gradients in the update rule with the L-infinity norm.[24] The update rule for Adamax can be written as:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g$$

$$u_t = \max(\beta_2 * u_{t-1}, |g|)$$

$$\eta = (\eta / (1 - \beta_1'))$$

$$\theta_{t+1} = \theta_t - \eta * m_t / u_t$$

where:

- θ is the vector of parameters
- g is the gradient vector
- m is the exponentially weighted moving average of the gradient
- u is the exponentially weighted moving average of the L-infinity norm of the gradient
- η is the learning rate
- β_1 and β_2 are the exponential decay rates for the moving averages
- t is the current iteration step

The main difference between Adamax and Adam is in the computation of u_t . In Adamax, u_t is the maximum of $\beta_2 u_{t-1}$ and the element-wise absolute value of the gradient $|g|$, whereas in Adam, u_t is the L2 norm of the gradient $\|g\|_2$. By using the L-infinity norm, Adamax is less sensitive to outliers and can better handle situations where some components of the gradient are much larger than others.[24]

7. **AdamW** : AdamW is a variant of the Adam optimizer that introduces weight decay regularization into the update rule.[24] The AdamW update rule can be written as follows:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$

$$w_t = w_{t-1} - \alpha * (m_t / (\sqrt{v_t} + \epsilon) + \text{weightdecay} * w_{t-1})$$

In this update rule

- w_t represents the updated weights
- $\mathbf{g}$ is the gradient vector
- g_t is the gradient of the loss function with respect to the weights
- m_t and v_t are the first and second moments of the gradient
- α is the learning rate
- β_1 and β_2 are exponential decay rates for the moment estimates
- ϵ is a small constant to prevent division by zero
- *weightdecay* is the strength of the weight decay regularization.

8. Nadam :

NAdam, or Nesterov-accelerated Adam, is a variant of the Adam optimizer that includes Nesterov momentum.[24] The NAdam update rule can be written as follows:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$

$$m_{t_{hat}} = m_t / (1 - \beta_1^t)$$

$$v_{t_{hat}} = v_t / (1 - \beta_2^t)$$

$$w_t = w_{t-1} - \alpha \left(\frac{\mu \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + (1 - \mu) \left(g_t + \frac{\beta_1 (\mu \hat{m}_t - g_t)}{1 - \beta_1^t} \right) \right)$$

In this update rule

- w_t represents the updated weights
- g_t is the gradient of the loss function with respect to the weights
- m_t and v_t are the first and second moments of the gradient
- $m_{t_{hat}}$ and $v_{t_{hat}}$ are the bias-corrected moments
- α is the learning rate
- β_1 and β_2 are exponential decay rates for the moment estimates
- ϵ is a small constant to prevent division by zero

- μ is the Nesterov momentum parameter.

9. **RMSprop :**

The RMSprop optimizer update rule[24] can be written as follows:

$$cache_t = decayrate * cache_{t-1} + (1 - decayrate) * g_t^2$$

$$w_t = w_{t-1} - \alpha * g_t / (\sqrt{(cache_t)} + \epsilon)$$

In this update rule

- w_t represents the updated weights
- g_t is the gradient of the loss function with respect to the weights
- $cache_t$ is the moving average of the squared gradients
- $decayrate$ is the decay rate for the moving average
- α is the learning rate
- ϵ is a small constant to prevent division by zero

10. **Ftrl :**

The FTRL optimizer update rule[24] can be written as follows:

$$z_t = z_{t-1} + g_t - (\sqrt{(n_t + g_t^2)} - \sqrt{(n_t)}) / \alpha_t * w_t$$

$$n_t = n_{t-1} + g_t^2$$

$$v_t = (\sqrt{(n_t + g_t^2)} - \sqrt{(n_t)}) / \alpha_t + v_{t-1}$$

$$w_t = -v_t / (\lambda + \sqrt{(n_t)}) * z_t$$

In this update rule

- w_t represents the updated weights
- g_t is the gradient of the loss function with respect to the weights
- z_t and n_t are accumulators for the weight updates
- v_t is an accumulator for the scaling of the weight updates

- α is a time varying learning rate
- λ is a regularization parameter

2.5 Model architecture

2.5.1 LSTM model

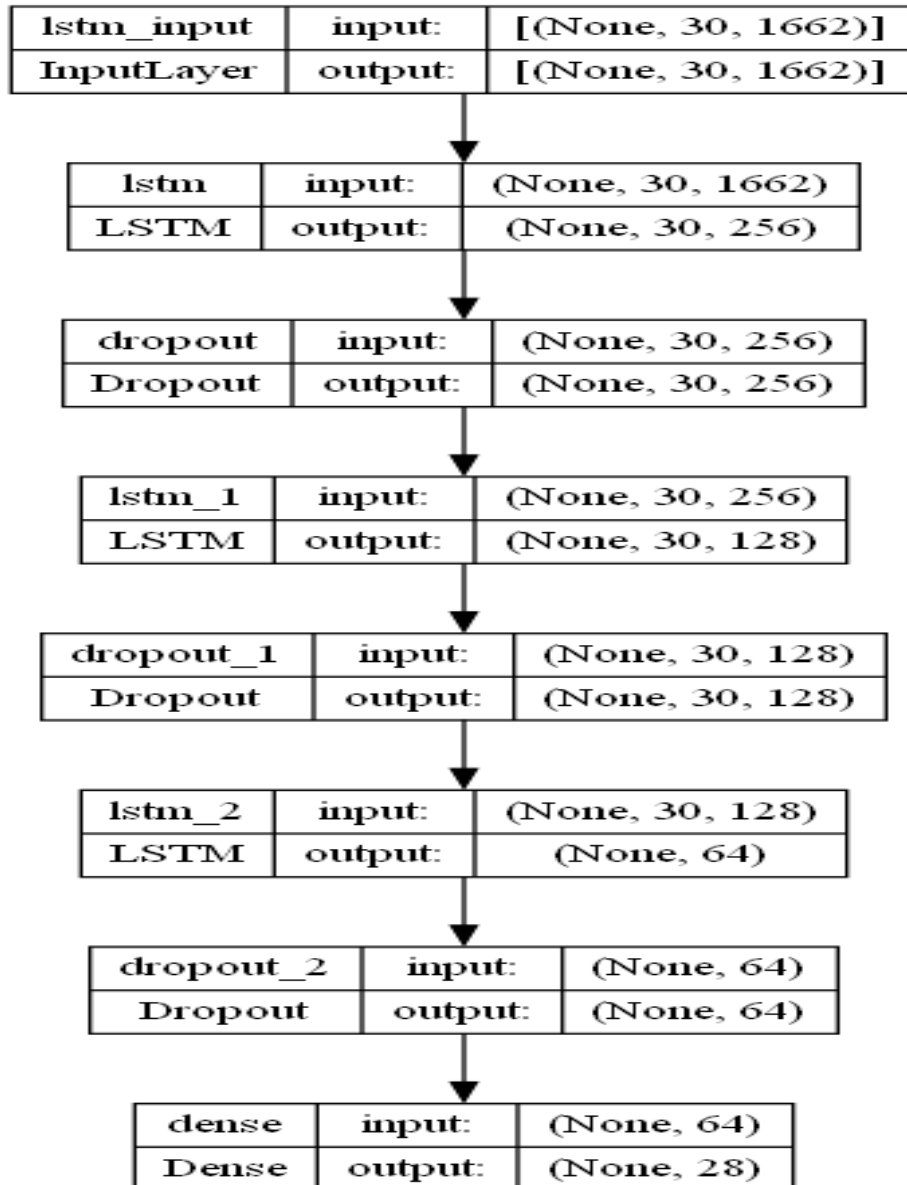


Figure 2.6: our Model architecture

The model architecture consists of three LSTM (Long short-term memory) layers and a dense output layer:

The input to the model is a tensor with shape (30, 1662), which means that the model can take in sequences of length 30, where each element in the sequence has a feature dimension of 1662.

The number of features may represent any numerical or categorical features of the input data. The first LSTM layer has 256 units, with the input shape being the shape of the input tensor. The return sequences parameter is set to True, which means that the output of this layer is also a sequence.

The Dropout layer is added after this layer, which helps prevent overfitting by randomly dropping out a fraction of the inputs to the layer during training. The second LSTM layer has 128 units, also with return sequences set to True. Another Dropout layer is added after this layer.

The third LSTM layer has 64 units and doesn't have return sequences parameter set to True, so it outputs only the final output of the sequence. Another Dropout layer is added after this layer.

Finally, a Dense layer with softmax activation function is added, which produces a probability distribution over the 28 possible output classes.

The model is trained using a categorical cross-entropy as a loss function and an NAdam optimizer

2.5.2 Model configuration and Number of parameters

Model: "sequential"		
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 30, 256)	1965056
dropout (Dropout)	(None, 30, 256)	0
lstm_1 (LSTM)	(None, 30, 128)	197120
dropout_1 (Dropout)	(None, 30, 128)	0
lstm_2 (LSTM)	(None, 64)	49408
dropout_2 (Dropout)	(None, 64)	0
dense (Dense)	(None, 28)	1820
=====		
Total params: 2,213,404		
Trainable params: 2,213,404		
Non-trainable params: 0		

Figure 2.7: The configuration of our model

2.6 Model Evaluation

2.6.1 Metrics

We used the categorical cross-entropy loss and accuracy metrics for the model evaluation. For the prediction results, we always generate a Confusion Matrix and Classification report to understand the Type I and Type II errors.

Cross Entropy Loss - The categorical cross-entropy loss function estimates the loss of an example by calculating the following sum:

$$L = - \sum_{c=1}^C y_c \log(p_c)$$

where y_c is the true label for class c (a one-hot encoded vector) and p_c is the predicted probability of class c

Accuracy - This metric describes how the model performs across all classes. It is helpful when all classes are equally important. It is evaluated as the ratio between the number of correct predictions to the total number of predictions,

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

where:

- TP = true positives (number of correctly predicted positive examples)
- TN = true negatives (number of correctly predicted negative examples)
- FP = false positives (number of incorrectly predicted positive examples)
- FN = false negatives (number of incorrectly predicted negative examples)

The equation for accuracy in multi-class classification is:

$$Accuracy = \frac{\sum_{i=1}^N \mathbb{1}_{y_i = \hat{y}_i}}{N}$$

where: argument is true, and 0 otherwise.

- N = total number of examples
- y_i = true label for example i
- $\hat{y}_i$ = predicted label for example i

- $\mathbb{1} \cdot$ = indicator function that returns 1 if the condition is true, and 0 otherwise.

Precision - It is used to get the number of positively classified predictions that actually belong to the class. Precision is obtained by analyzing the confusion matrix, and is given by,

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

Recall - It is used to obtain the number of positively classified predictions out of the total number of predictions. We infer the Recall value from the confusion matrix, and is given by

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

f1 score - is a common metric used in classification tasks that balances both precision and recall. It is defined as the harmonic mean of precision and recall, and is given by the following equation:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In the case of multi-class classification, the F1 score can be computed separately for each class, and then averaged across all classes using a suitable method such as macro-average or micro-average.

Confusion Matrix - is a table that summarizes the performance of a classification model on a set of test data for which the true labels are known. It is a useful tool for evaluating the performance of a classification model, and provides insights into the types of errors that the model is making.

	True diagnosis		Total
	Positive	Negative	
Positive	TP	$FN(TypeI\ error)$	$TP + FN$
Negative	$FP(TypeI\ error)$	TN	$FP + TN$
Total	$TP + FP$	$FN + TN$	N

Table 2.1: Confusion Matrix

2.7 Conclusion

Based on the information provided, the "Conceptual Study" chapter appears to cover three main aspects of the project:

- **The design of the system:**

This likely includes a discussion of the overall purpose of the system, the intended user base, and the key features and functionality of the system.

- **The data set used in the project:**

This may include details about the size and composition of the data set, how it was collected, any preprocessing or cleaning that was done, and any challenges or limitations encountered while working with the data.

- **Optimizer:**

This includes a detailed description of the Optimazer used in the project

- **The model architecture and configuration:**

This likely includes a detailed description of the specific neural network architecture used in the project, along with information about the number of parameters and any hyperparameters that were tuned during model training.

- **Model Evaluation:**

the Model Evaluation section presents an analysis of how well the model performed in terms of accuracy, recall, precision, F1 score, and confusion matrix. This section provides critical insights into the effectiveness of the model and can inform any necessary adjustments or improvements to the system.

Overall, the "Conceptual Study" chapter provides a high-level overview of the key elements of the project, including the problem being addressed, the data used to train the model, the architecture and configuration of the model itself, Model Evaluation. This information is important for understanding the approach taken in the project and the potential strengths and limitations of the final model.

Results and Implementation work

3.1 Introduction

This chapter is divided into three main sections

- The first section outlines the chosen tools and technologies for the system development. It provides an overview of the programming languages, libraries, and frameworks used in the implementation of the system. The tools chosen for the project are discussed in detail, along with their advantages and limitations.
- The second section discusses and compares the results obtained from the system testing.
The evaluation metrics used to assess the performance of the system are presented, and the results are analyzed and compared against the expectations and requirements of the system.
- The third and final section of the chapter presents the system interface and displays some screenshots of the test phase.

Overall, this chapter provides a comprehensive overview of the development and testing of the computer vision system, from the tools and technologies used to the final user interface and test results.

3.2 Representation of the development tools

3.2.1 Physical environment

we utilize this hardware in training and testing processes.

Component	Brand/Model	Specification
CPU	Ryzen 5 5600g	6 cores, 12 threads, 3.9 GHz base, 4.4 GHz
GPU	Amd rx 580	2304 Stream processors, 8GB GDDR5 memory, 1257 MHz boost
RAM	XPG Gammix D30	16GB DDR4-3200MHz, CL16, dual-channel
Motherboard	ASUS B450M-K II	mATX, AM4, AMD B450 chipset
Storage	hyper ssd	128 GO SSD
Power Supply	Raidmax XTB Series	550W, 80+ Bronze certified

Table 3.1: Desktop Hardware

3.2.2 Software and libraries used in the implementation

Operating system : 64-bit Windows 11.

Python : Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. It is designed to be easy to read, write, and maintain, with a syntax that emphasizes code readability and simplicity. Python is a popular choice for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, and automation. It features a large standard library, as well as numerous third-party packages and modules, making it a versatile language that can be used for a variety of tasks. Python is also known for its strong community support, with numerous resources, documentation, and tutorials available online, making it an accessible language for developers of all skill levels. Overall, Python is a powerful and flexible language that has become one of the most popular programming languages in the world.[Pyt]

VS Code:is a free, open-source code editor developed by Microsoft. It provides a powerful and flexible environment for coding, debugging, and building applications across a wide range of languages and platforms. VS Code offers a wide range of features, including support for syntax highlighting, code completion, debugging, version control, and extensions. [25]

TensorFlow: is a popular open-source machine learning framework developed by Google. It allows

developers to create and train various machine learning models, including neural networks, and deploy them for various applications. TensorFlow provides a wide range of tools and libraries for data manipulation, model training, and deployment, making it a powerful and flexible tool for machine learning. It supports various programming languages, including Python, C++, and Java, and can be used on a variety of platforms, including CPUs, GPUs, and mobile devices. TensorFlow has been widely adopted in the industry and academia for a wide range of applications, including computer vision, natural language processing, and speech recognition[4]

Keras: is a popular open-source deep learning framework developed in Python. It provides a high-level interface to build and train neural networks, allowing developers to quickly create and prototype deep learning models with minimal code. Keras supports multiple backends, including TensorFlow, CNTK, and Theano, and can be used on a variety of platforms, including CPUs, GPUs, and distributed systems. Keras provides a wide range of pre-built layers and models, making it easy to create complex architectures for various applications, including computer vision, natural language processing, and time-series analysis. With its simplicity, modularity, and flexibility, Keras has been widely adopted by researchers and developers for building and training deep learning models.[23]

NumPy: (short for Numerical Python) is a popular open-source library for scientific computing in Python. It provides a multidimensional array object, as well as a wide range of mathematical functions, and tools for working with arrays. NumPy is widely used in various fields of scientific computing, such as physics, finance, engineering, and machine learning, among others.[71]

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It was initially developed by Intel, and now it is maintained by the OpenCV community. The library provides a variety of tools and algorithms for computer vision and image processing applications.[20]

Mediapipe: is an open-source framework developed by Google for building real-time multi-modal (audio, video, and sensor) applications. It provides a customizable pipeline architecture for processing media streams, including computer vision and machine learning models, and offers a collection of pre-built components for common tasks such as face detection, pose estimation, and object tracking. [38]

Scikit-learn: , commonly abbreviated as sklearn, is a popular open-source machine learning library for Python. It provides a wide range of supervised and unsupervised learning algorithms, including classification, regression, clustering, and dimensionality reduction, among others. Scikit-learn is built on top of other popular Python libraries, such as NumPy, SciPy, and Matplotlib, and provides a simple and consistent API that makes it easy to use and integrate into data science workflows.[74]

3.3 Experiments, comparison and discussion the obtained results

In this section, we will discuss and compare the results obtained from a large number of models developed for our project. Each model will be analyzed individually, with a summary name given to each one based on its architecture and number of units. We will then present the results, including accuracy, loss, validation accuracy, and validation loss, and provide a detailed discussion of the findings. Finally, we will summarize the results in a table to provide a comprehensive overview of the performance of each model.

3.3.1 Experiments and Results

The models are numbered from 1 to 7, and some of them have sub-models (e.g., Model 4 has sub-models labeled as 4(a), 4(b), etc.).

For each model/sub-model, the following performance metrics are provided:

- **Accuracy:** a measure of how many predictions the model got right (number of correct predictions / total number of predictions)
- **Loss:** a measure of the error of the model's predictions (usually measured as a difference between predicted and actual values)
- **Validation Accuracy:** the accuracy of the model's predictions on a validation dataset (a subset of the dataset that was not used during training, used to evaluate the model's performance on unseen data)
- **Validation Loss:** the loss of the model's predictions on the validation dataset
- **Epoch:** the number of epochs (complete passes through the dataset) that the model was trained for.

3.3.1.1 In terms of models

In this section, we will show A huge part of our experiments with different Models With Adam optimizer:

1. Model 1: Conv1D-32-5-Relu-MaxPool1D-3-Conv1D-64-5-Relu-MaxPool1D-3-Flatten-Reshape-Dropout-0.2-LSTM-64-Dense-Sigmoid

The model uses 32 and 64 filters and kernel size of 5. in the first two convolutional layers and ReLU activation. The MaxPooling1D layer is used to reduce the input tensor's spatial dimensions. The dropout layer is added to prevent overfitting. The LSTM layer has 64 units, followed by a dense layer with sigmoid activation.

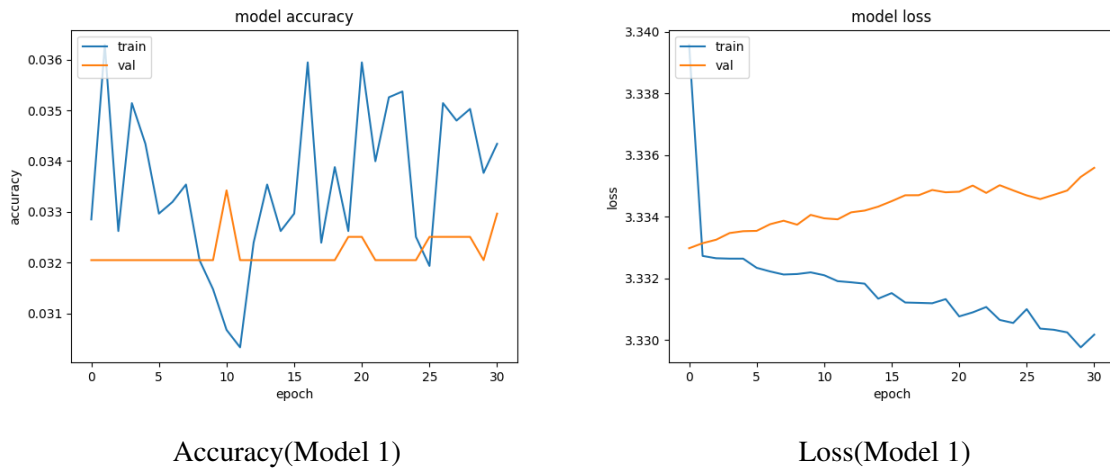


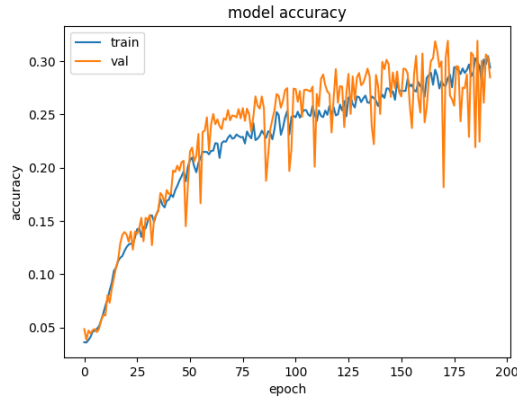
Figure 3.1: Chart of accuracy and loss of Model 1

The model trained for 10 epochs and stopped at 30 epochs using early stopping. However, the model's accuracy is very low, only 3%, which means that it does not perform well on the given task. The loss values for both the training and validation sets are also high, indicating that the model is not effectively learning from the data.

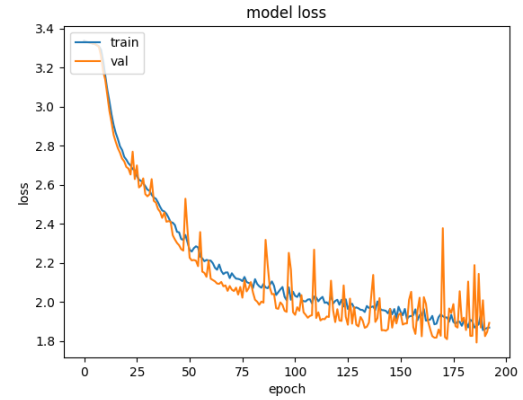
2. Model 2: Conv1D-64-3-Relu-MaxPool1D-Conv1D-64-3-Relu-MaxPool1D-LSTM-64-Dense-Sigmoid

The model uses 64 filters in the first two convolutional layers and ReLU activation. The LSTM layer has 64 units, followed by a dense layer with sigmoid activation.

The model achieved a training accuracy of 0.284 and a training loss of 1.92 after 171 epochs, and was stopped after 192 epochs using early stopping. The validation accuracy of 0.319 and validation loss of 1.81 indicate that the model's performance on the validation set is not very good.



Accuracy(Model 2)

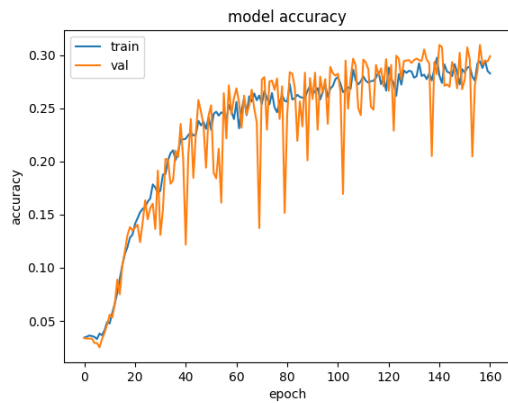


Loss(Model 2)

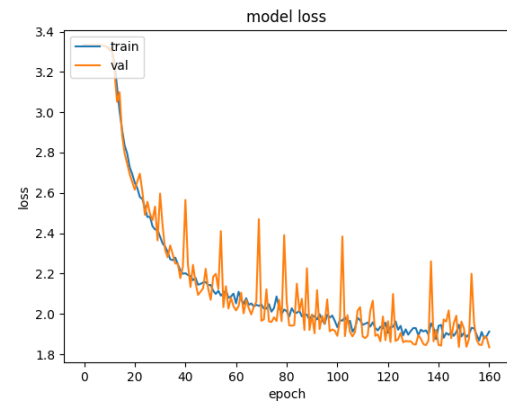
Figure 3.2: Chart of accuracy and loss of Model 2

3. Model 3: Conv1D-32-3-Relu-MaxPool1D-Conv1D-64-3-Relu-MaxPool1D-LSTM-128-LSTM-64-Dense-Sigmoid

th's model consisting of a TWO 1D convolutional layer with 32 and 64 filters and a kernel size of 3, followed by a max pooling layer with a pool size of 2. Two LSTM layers with 128 and 64 units respectively are added, with the first layer returning sequences. The final layer is a dense layer with a sigmoid



Accuracy(Model 3)



Loss(Model 3)

Figure 3.3: Chart of accuracy and loss of Model 3

The model did not perform well in terms of accuracy and loss, achieving a training accuracy of only 27% and a high training loss of 1.95 after 140 epochs. The early stopping mechanism kicked in after 160 epochs. The validation accuracy of 30% and validation loss of 1.84 indicate that the model did not generalize well to the validation set either.

4. Model 4: LSTM-64-128-64-Dense-64-32-Softmax

Where "LSTM" stands for the Long Short-Term Memory recurrent neural network layer, "3x64,128,64" indicates that the model has three LSTM layers with 64,128 and 64, units respectively, and "Dense-64-32" indicates that a two Dense layer

- **Model 4(a):No Dropout**

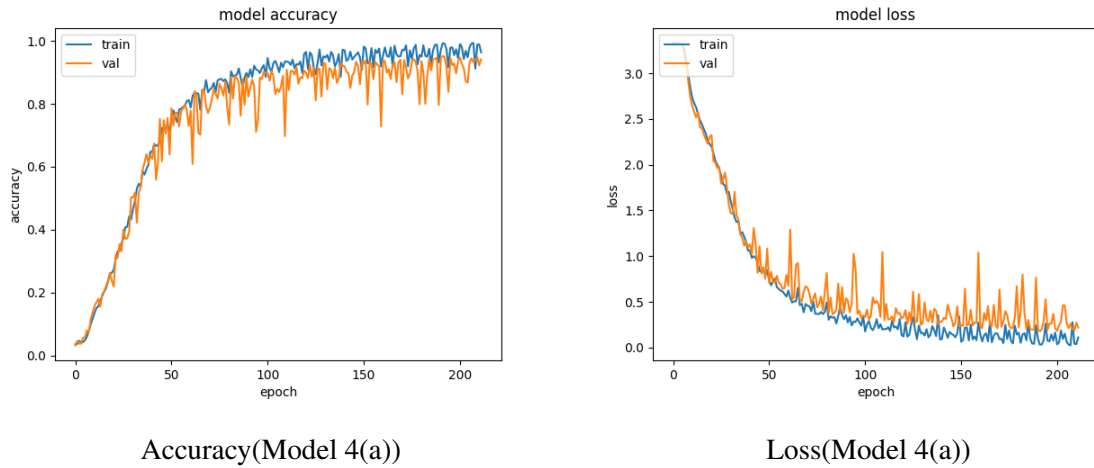


Figure 3.4: Chart of accuracy and loss of Model 4(a)

the model achieved a high training accuracy of 0.99 and a low training loss of 0.03 after 192 epochs. The model was stopped after 211 epochs with early stopping.

The validation accuracy of 0.948 and validation loss of 0.19 indicate that the model's performance is slightly worse on the validation set than on the training set.

- **Model 4(b):Dropout0.1**

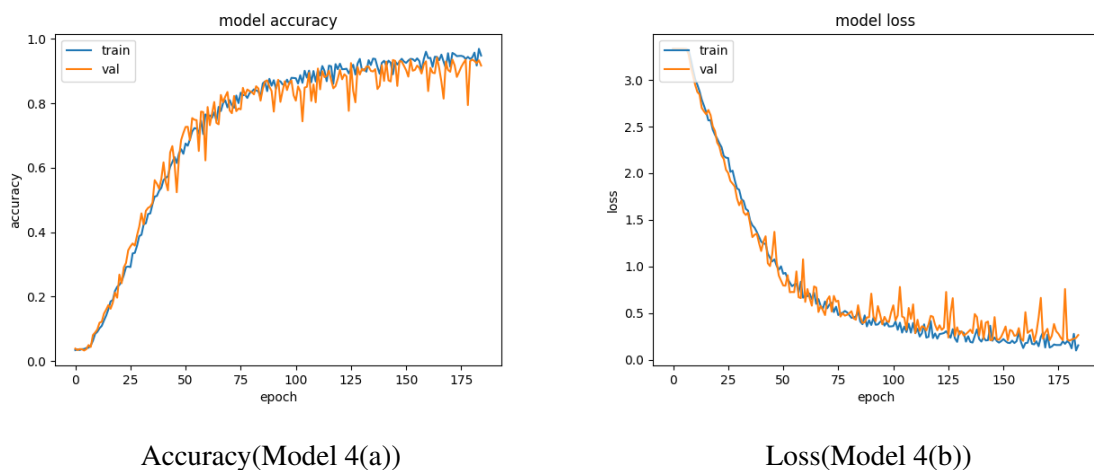


Figure 3.5: Chart of accuracy and loss of Model 4(b)

The reported result of achieving an accuracy of 0.944 and validation accuracy of 0.942

with a loss of 0.165 on the best performing model is quite impressive. It is interesting to note that the model was stopped early at epoch 184, which indicates that the model was already performing well and had converged on a good solution.

- **Model 4(c):Dropout0.2**

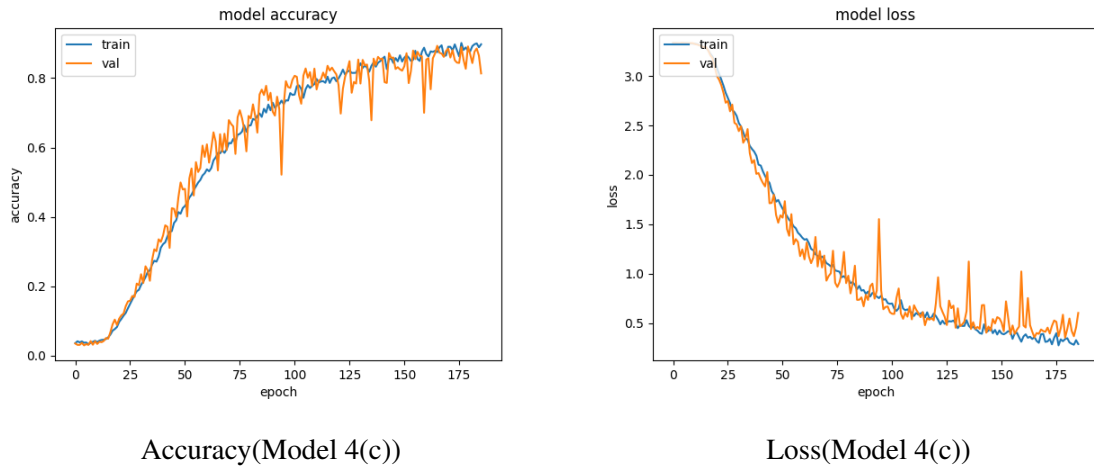


Figure 3.6: Chart of accuracy and loss of Model 4(c)

the model achieved a high training accuracy of 0.87 and a training loss of 0.35 after 165 epochs. The model was stopped after 187 epochs with early stopping.

The validation accuracy of 0.89 and validation loss of 0.35

- **Model 4(d):Dropout0.3**

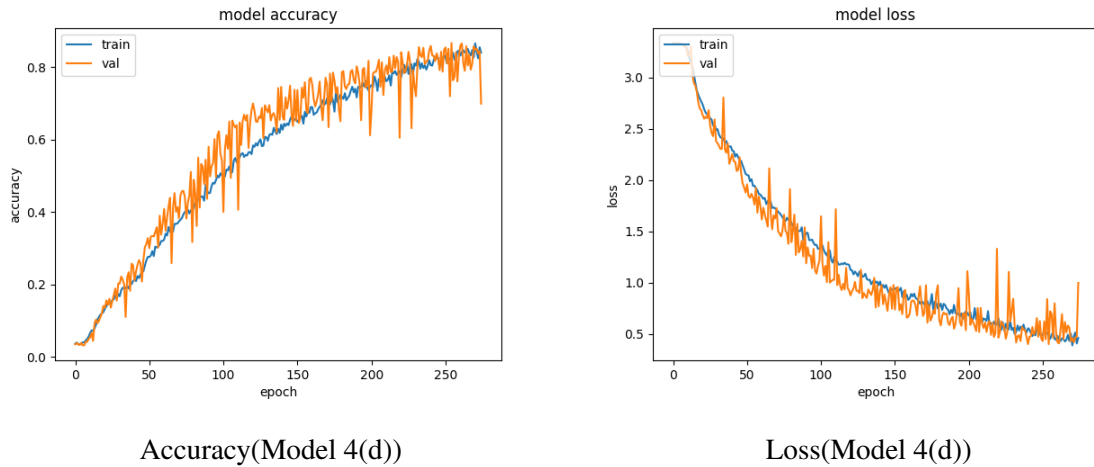


Figure 3.7: Chart of accuracy and loss of Model 4(d)

The trained model achieved an accuracy of 83.31% in epoch 254 and stopped in epoch 274 using early stopping. The loss value is 0.473404318, which is relatively high. However, the validation accuracy is also high at 86.68%, indicating that the model is performing well and is not overfitting. The validation loss is also relatively low at 0.399460256.

- **Model 4(e): Dropout0.4**

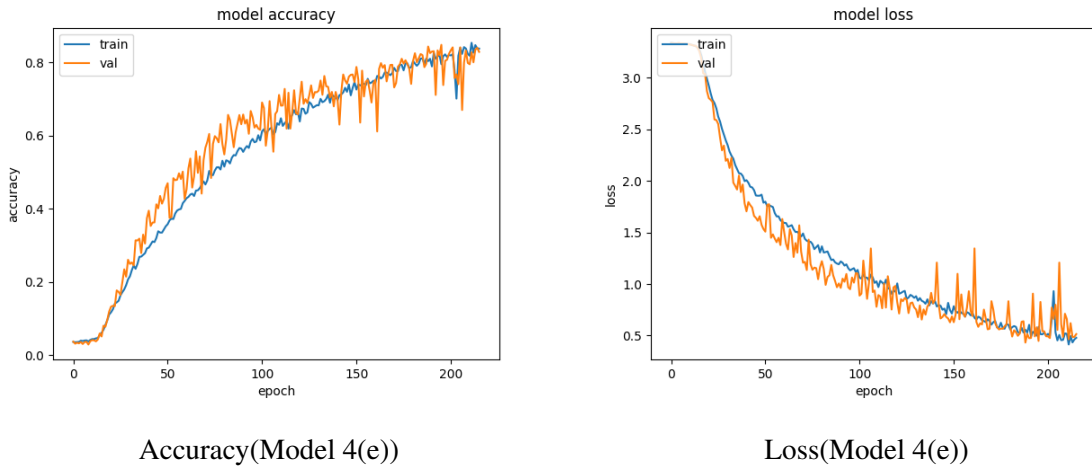


Figure 3.8: Chart of accuracy and loss of Model 4(e)

The trained model achieved an accuracy of 79.50% in epoch 195 and stopped in epoch 215 using early stopping. The loss value is relatively high at 0.584443033. However, the validation accuracy is also relatively high at 84.80%, indicating that the model is performing well and is not overfitting. The validation loss is also relatively low at 0.448069185.

- **Model 4(f): Dropout0.5**

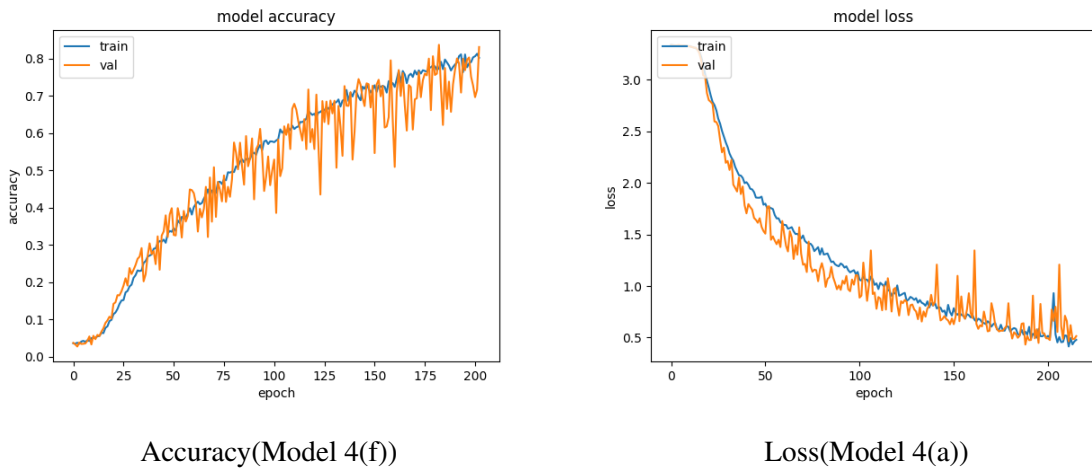


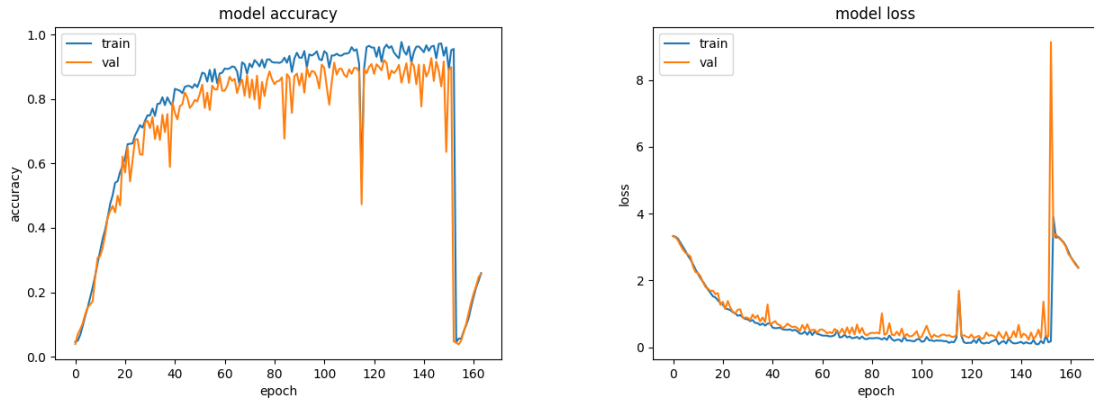
Figure 3.9: Chart of accuracy and loss of Model 4(f)

The trained model achieved an accuracy of 77.37% in epoch 182 and stopped in epoch 202 using early stopping. The loss value is relatively high at 0.646953642. However, the validation accuracy is also relatively high at 83.70%, indicating that the model is performing well and is not overfitting. The validation loss is also relatively low at 0.479840517.

5. Model 5: LSTM-64-128-64-Softmax

the model's use of three LSTM layers with 64, 128, and 64 units, respectively, followed by a softmax output layer.

- **Model 5(a):No Dropout**



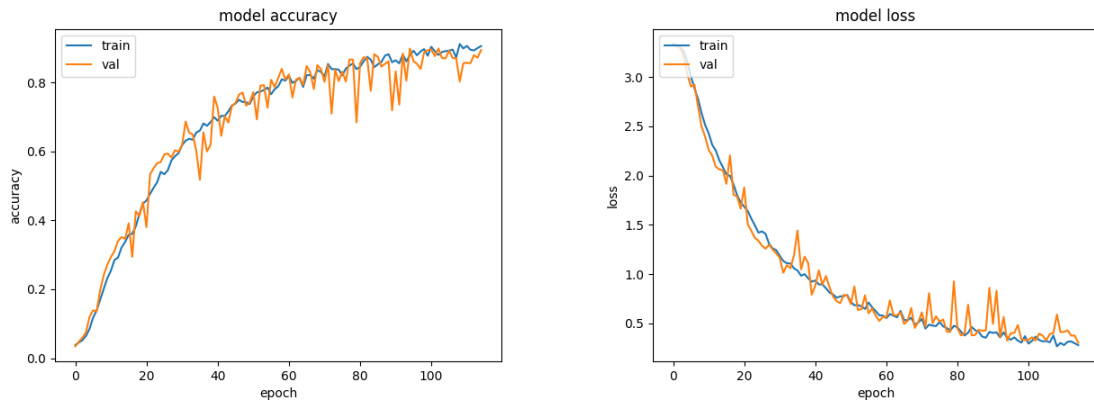
Accuracy(Model 5(a))

Loss(Model 5(a))

Figure 3.10: Chart of accuracy and loss of Model 5(a)

the model achieved high accuracy on the training set (96.29%) but lower accuracy on the validation set (92.72%).

- **Model 5(b): Dropout 0.1**



Accuracy(Model 5(b))

Loss(Model 5(b))

Figure 3.11: Chart of accuracy and loss of Model 5(b)

we can see that the model has achieved an accuracy of 0.880 with a validation accuracy of 0.898, with early stopping applied at the 114th epoch. The loss values for both training and validation are 0.352 and 0.324, respectively.

- **Model 5(c): Dropout 0.2**

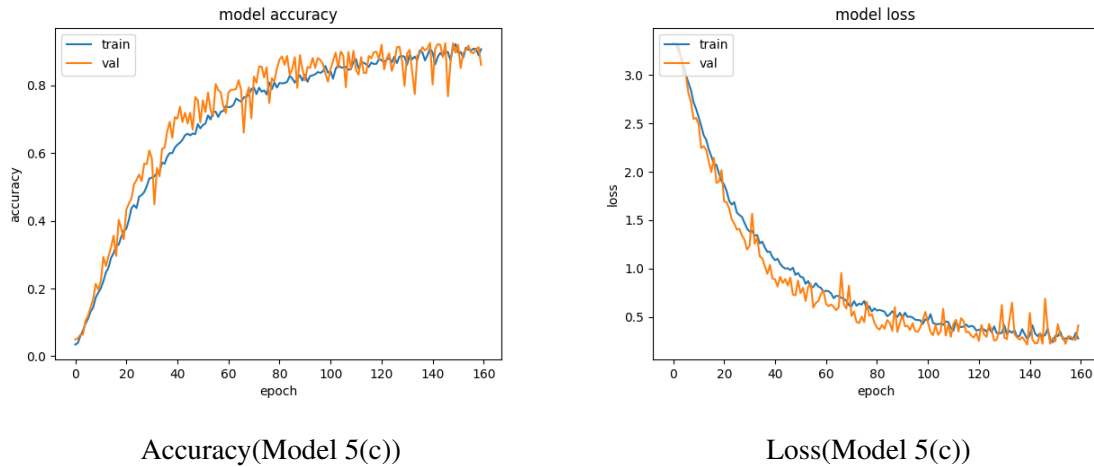


Figure 3.12: Chart of accuracy and loss of Model 5(c)

This is a good result, with a validation accuracy of 0.926 and a validation loss of 0.215. The model achieved an accuracy of 0.909 on the training data, which is close to the validation accuracy, indicating that the model did not overfit the data.

- **Model 5(d):Dropout 0.3**

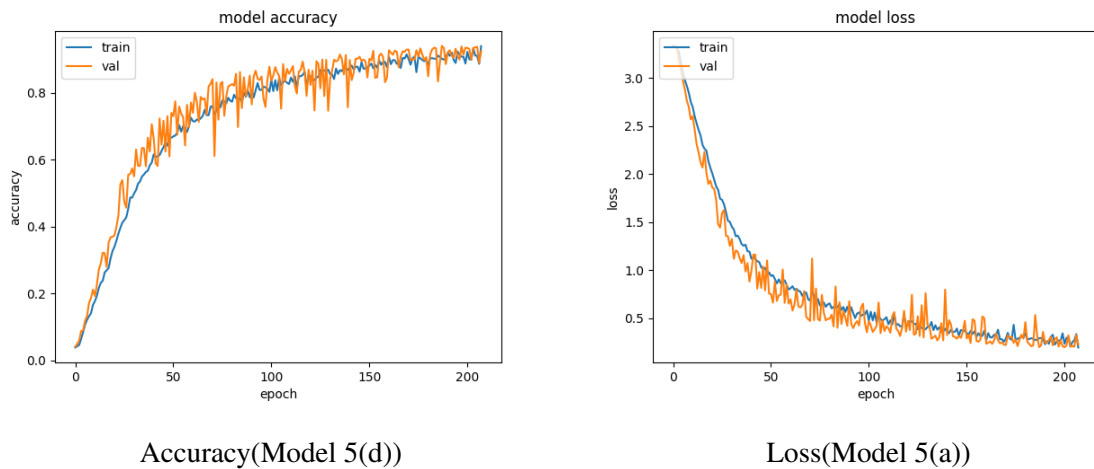


Figure 3.13: Chart of accuracy and loss of Model 5(d)

The trained model achieved an accuracy of 90.22% in epoch 187 and stopped in epoch 207 using early stopping. The loss value is relatively low at 0.290392697. The validation accuracy is also relatively high at 94.13%, indicating that the model is performing well and is not overfitting. The validation loss is also relatively low at 0.197899193.

- **Model 5(e): Dropout 0.4**

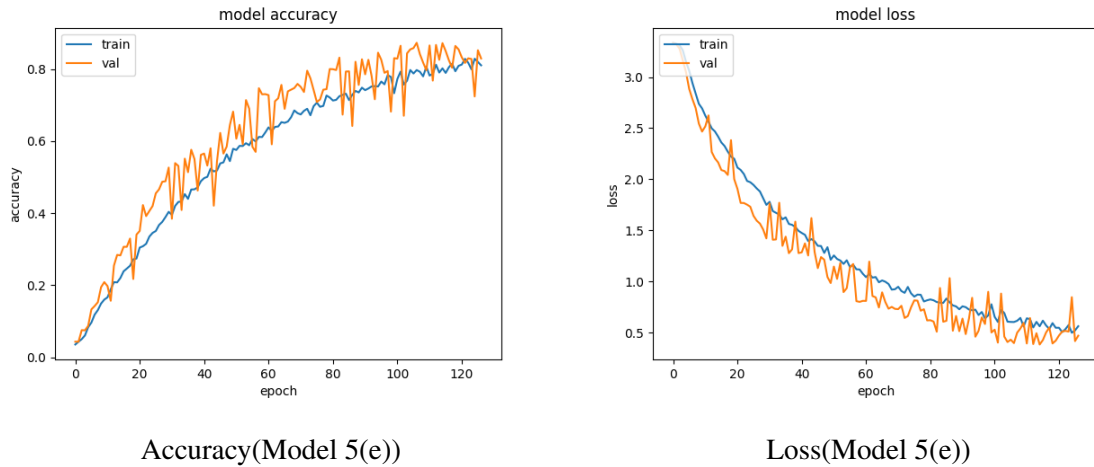


Figure 3.14: Chart of accuracy and loss of Model 5(e)

The trained model achieved an accuracy of 79.72% in epoch 106 and stopped in epoch 126 using early stopping. The loss value is relatively high at 0.603025734. However, the validation accuracy is also relatively high at 87.27%, indicating that the model is performing well and is not overfitting. The validation loss is also relatively low at 0.396841824.

- **Model 5(f):Dropout 0.5**

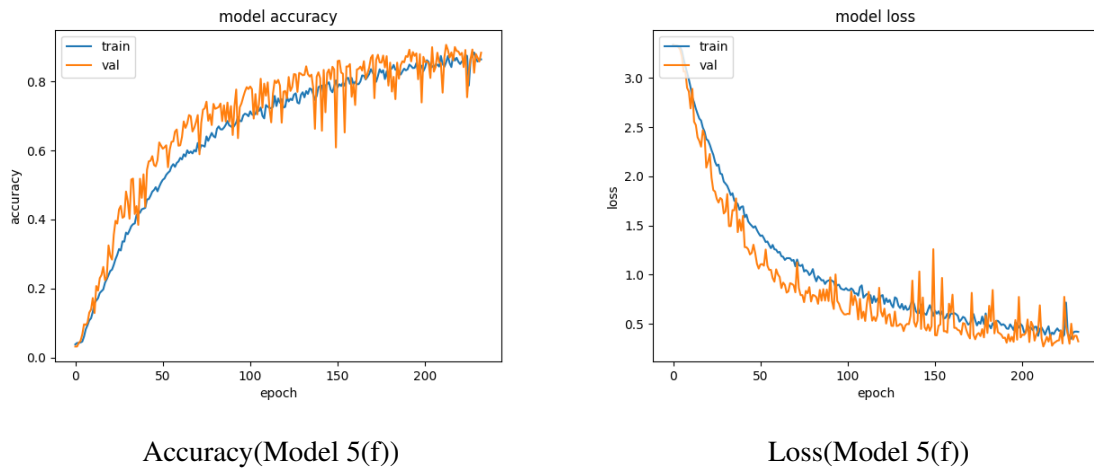


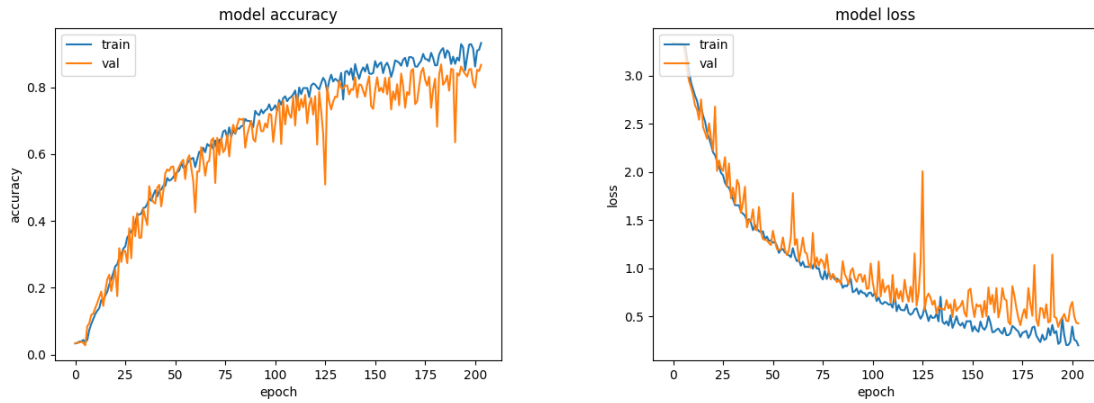
Figure 3.15: Chart of accuracy and loss of Model 5(f)

The trained model achieved an accuracy of 85.57% in epoch 212 and stopped in epoch 232 using early stopping. The loss value is 0.430245161 which is relatively high, but the validation accuracy is also relatively high at 90.61%. This indicates that the model is performing well on the validation set and is not overfitting. The validation loss value of 0.270342797 is also relatively low.

6. Model 6 : LSTM-128-64-32-Softmax

the model's use of three LSTM layers with 128, 64, and 32 units, respectively, followed by a softmax output layer.

- **Model 6(a): No Dropout**



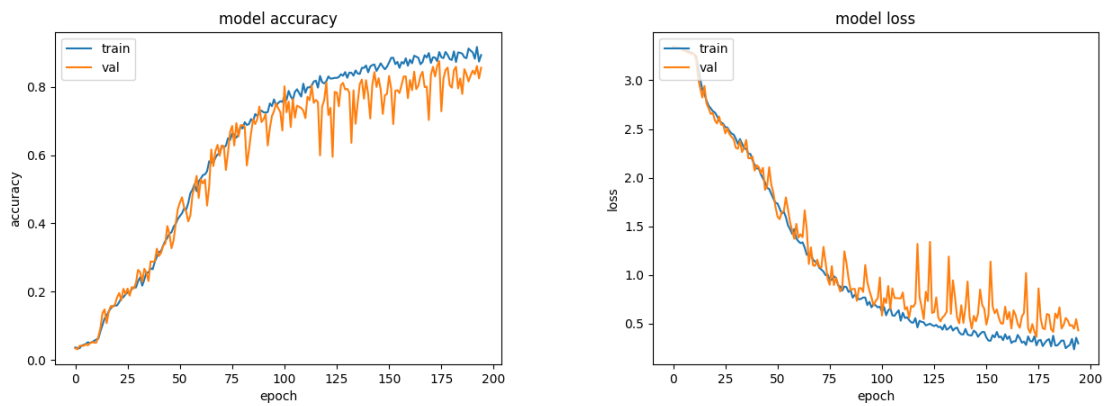
Accuracy(Model 6(a))

Loss(Model 6(a))

Figure 3.16: Chart of accuracy and loss of Model 6(a)

It seems that the trained model achieved a good accuracy of 91.11% in epoch 182 and stopped at epoch 203 using early stopping. The validation accuracy was 86.90% and the validation loss was 0.40.

- **Model 6(b):Dropout0.1**



Accuracy(Model 6(b))

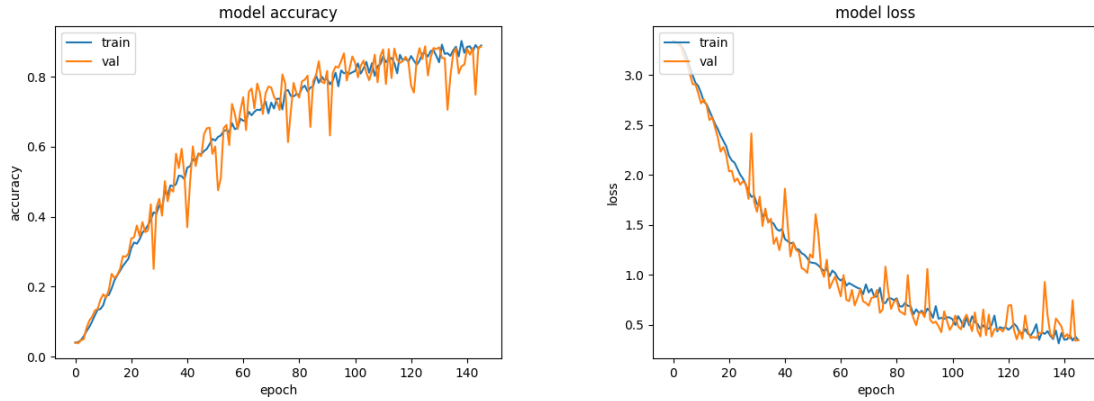
Loss(Model 6(b))

Figure 3.17: Chart of accuracy and loss of Model 6(b)

The trained model achieved a good accuracy of 90.28% in epoch 174 and stopped in epoch 194 using early stopping. The loss value is 0.278117508 which is also good. The validation

accuracy is 87.59% and the validation loss is 0.365888804. These results suggest that the model is performing well and is not overfitting.

- **Model 6(c): Dropout 0.2**



Accuracy(Model 6(c))

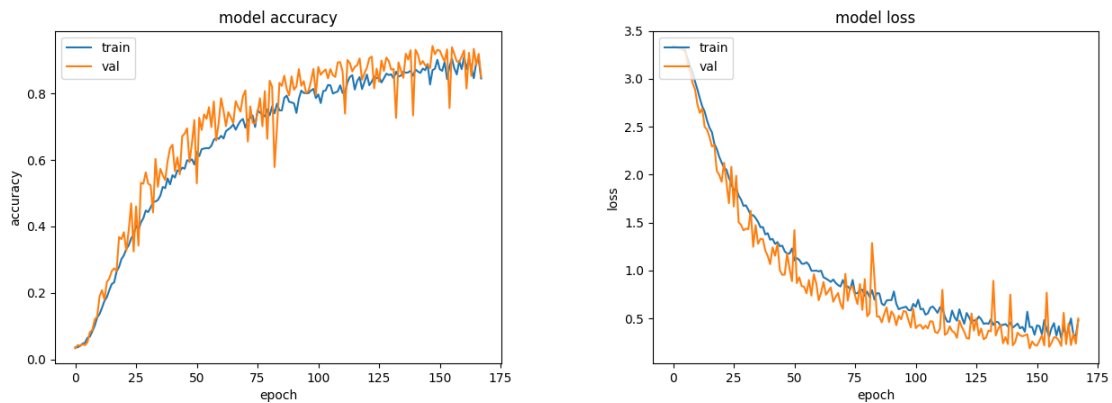
Loss(Model 6(c))

Figure 3.18: Chart of accuracy and loss of Model 6(c)

the model achieved a training accuracy of 0.875 and a training loss of 0.393 after 128 epochs. The model was stopped after 145 epochs with early stopping.

The validation accuracy of 0.881 and validation loss of 0.369 indicate that the model's performance is slightly better on the validation set than on the training set.

- **Model 6(d):Dropout 0.3**



Accuracy(Model 6(d))

Loss(Model 6(d))

Figure 3.19: Chart of accuracy and loss of Model 6(d)

The trained model achieved an accuracy of 86.97% in epoch 147 and stopped in epoch 167 using early stopping. The loss value is 0.411395133 which is relatively high, but the validation accuracy is relatively high at 94.23%. This indicates that the model is

performing well on the validation set and is not overfitting. The validation loss value of 0.188575163 is relatively low.

- **Model 6(e):Dropout 0.4**

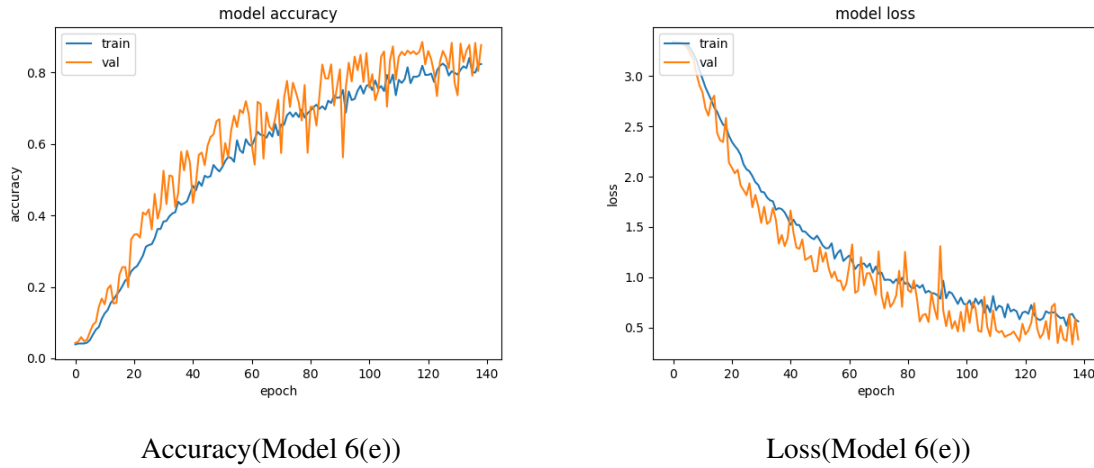


Figure 3.20: Chart of accuracy and loss of Model 6(e)

It seems that the model's accuracy on the training set is not very high, only around 81.9%. The loss value is also quite high, indicating that the model may be overfitting to the training data. However, the validation accuracy is quite good, around 88.5%, and the validation loss is not too high. This suggests that the model is performing well on the unseen data in the validation set.

- **Model 6(f):Dropout 0.5**

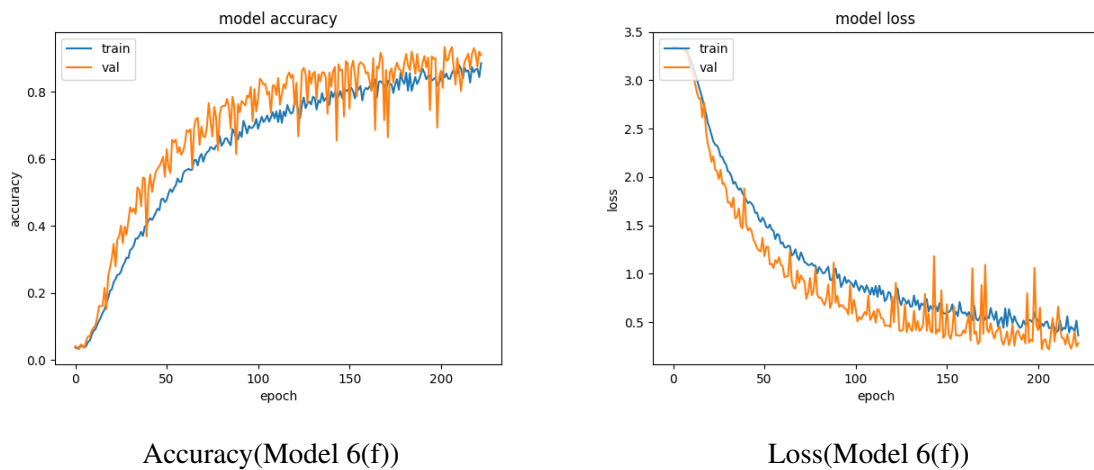


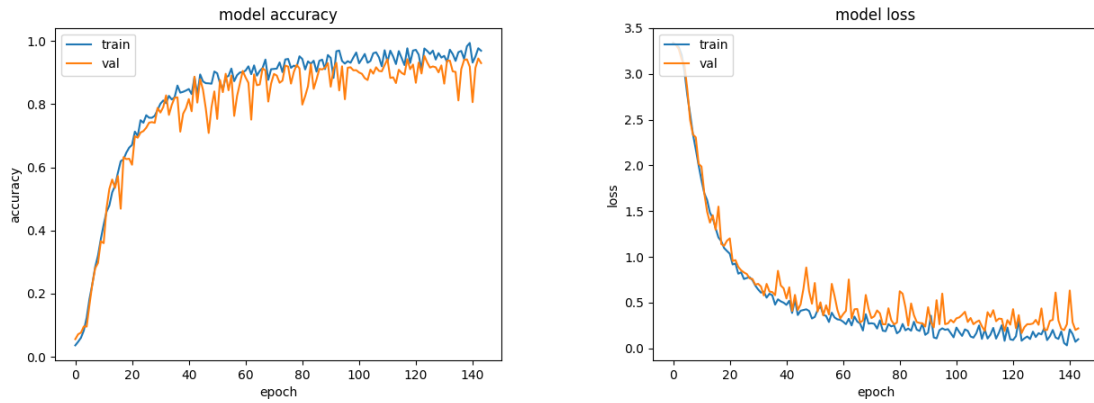
Figure 3.21: Chart of accuracy and loss of Model 6(f)

the model achieved a training accuracy of 0.843 and a validation accuracy of 0.935 after training for 202 epochs and stopping early at epoch 222. The training loss was 0.486, and the validation loss was 0.220

7. Model 7: LSTM-256-128-64-Softmax

the model's use of three LSTM layers with 256, 128, and 64 units, respectively, followed by a softmax output layer.

- **Model 7(a): No Dropout**



Accuracy(Model 7(a))

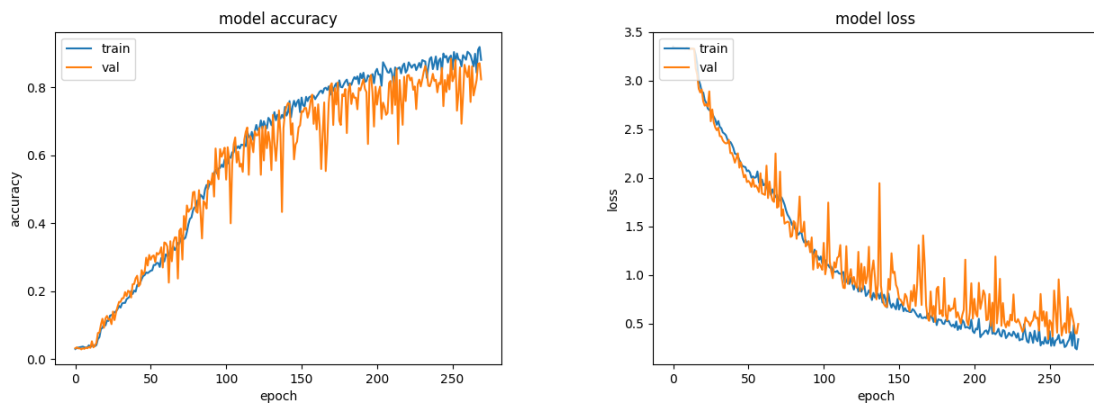
Loss(Model 7(a))

Figure 3.22: Chart of accuracy and loss of Model 7(a)

the model achieved a training accuracy of 0.976 and a training loss of 0.082 after 123 epochs. The model was stopped after 143 epochs with early stopping.

The validation accuracy of 0.952 and validation loss of 0.166 indicate that the model's performance is quite good on the validation set. it seems that this model is performing well, with a high training accuracy and validation accuracy. This model may be a strong candidate for use in my project.

- **Model 7(b):Dropout0.1**



Accuracy(Model 7(b))

Loss(Model 7(b))

Figure 3.23: Chart of accuracy and loss of Model 7(b)

The model has achieved an accuracy of 89.39% on the training set and 88.50% on the validation set after training for 250 epochs, and early stopping was triggered at epoch 269. The model has a loss of 0.297 on the training set and 0.340 on the validation set. Overall, the model seems to be performing well, as the accuracy on the validation set is close to the accuracy on the training set, indicating that the model is not overfitting to the training data.

- **Model 7(c): Dropout0.2**

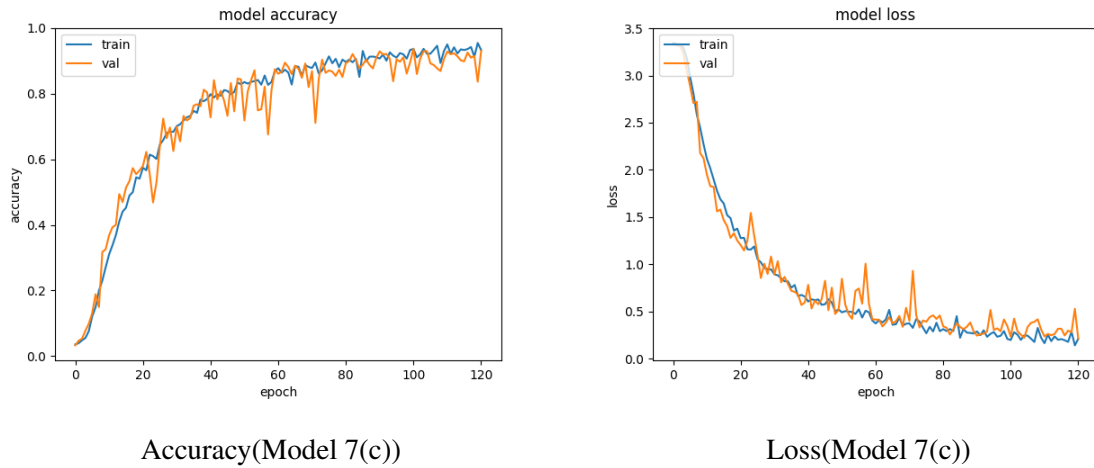


Figure 3.24: Chart of accuracy and loss of Model 7(c)

The model achieved a training accuracy of 0.936 and a training loss of 0.197 after 100 epochs. The model was stopped after 120 epochs with early stopping.

The validation accuracy of 0.937 and validation loss of 0.232 indicate that the model's performance is quite good on the validation set.

- **Model 7(d):Dropout0.3**

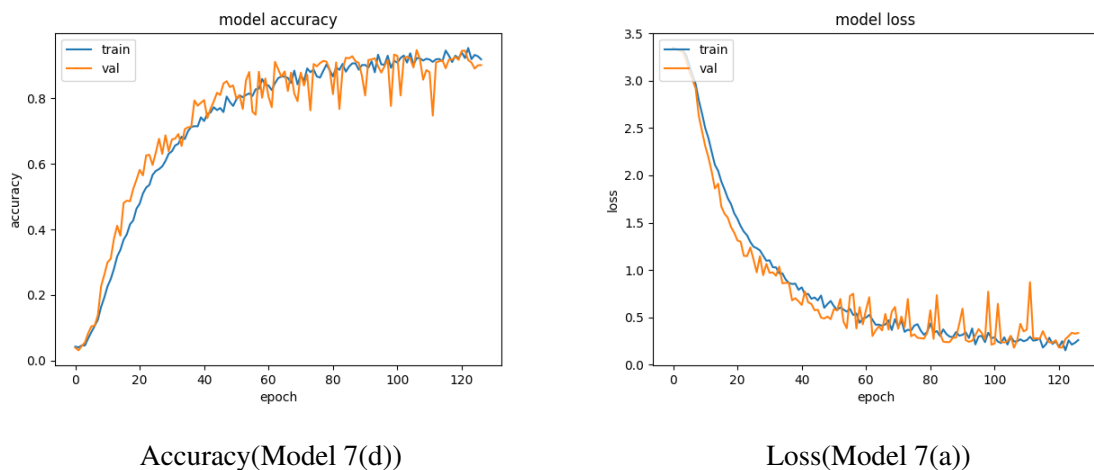


Figure 3.25: Chart of accuracy and loss of Model 7(d)

The trained model achieved a high accuracy of 92.19% in epoch 106 and stopped in epoch 126 using early stopping.

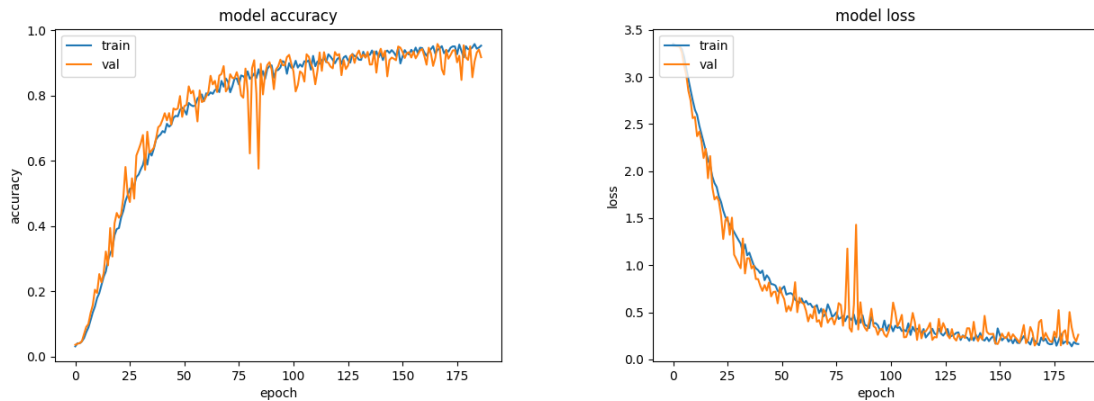
The loss value is low at 0.246491149.

The validation accuracy is very high at 94.64%, indicating that the model is performing very well and is not overfitting.

The validation loss is also low at 0.180725157.

These results suggest that the model is performing exceptionally well and is a good fit for the data.

- **Model 7(e):Dropout0.4**



Accuracy(Model 7(e))

Loss(Model 7(e))

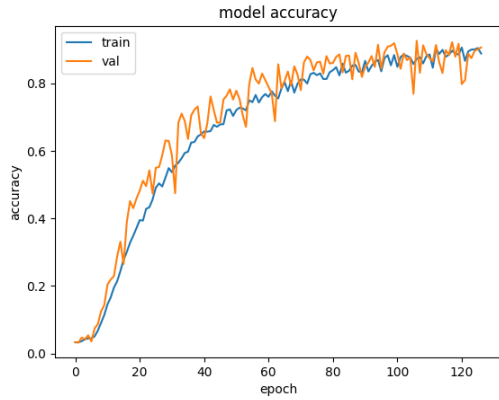
Figure 3.26: Chart of accuracy and loss of Model 7(e)

The trained model achieved a high accuracy of 93.96% in epoch 166 and stopped in epoch 186 using early stopping, The loss value is low at 0.197881401, indicating good performance, The validation accuracy is very high at 95.83%, indicating that the model is performing very well and is not overfitting, The validation loss is also low at 0.146998927. These results suggest that the model is performing exceptionally well and is a good fit for the data.

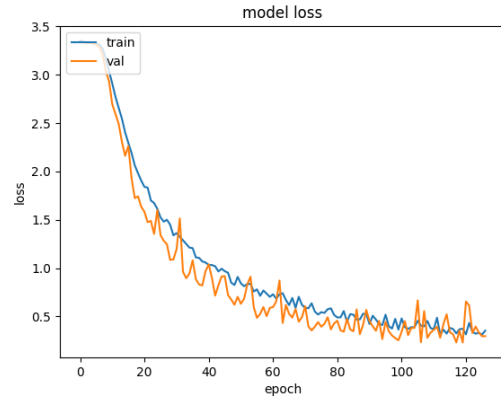
Overall, this model seems to be highly accurate and performing very well.

- **Model 7(f):Dropout0.5**

The trained model achieved an accuracy of 87.26% in epoch 106 and stopped in epoch 126 using early stopping, The loss value is 0.404824913, which is relatively high, However, the validation accuracy is high at 92.63%, indicating that the model is performing well and is not overfitting, The validation loss is relatively low at 0.233692884. Overall, this model seems to be performing well but there may be room for improvement in the training process to decrease the loss value.



Accuracy(Model 7(f))



Loss(Model 7(a))

Figure 3.27: Chart of accuracy and loss of Model 7(f)

This table shows the performance metrics of several models, including accuracy, loss, validation accuracy, validation loss, and number of epochs. The best performing model in each group is highlighted in green, while the worst performing model is highlighted in red.

Model	Accuracy	Loss	Val. Accuracy	Val. Loss	Epoch
1	0.0307	3.3321	0.0334	3.3340	30
2	0.2841	1.9203	0.3191	1.8089	172
3	0.2808	1.9410	0.3095	1.8460	140
4(a)	0.9927	0.0309	0.9524	0.1770	192
4(b)	0.9441	0.1649	0.9423	0.1953	184
4(c)	0.8749	0.3574	0.8933	0.3548	165
4(d)	0.8331	0.4734	0.8668	0.3995	254
4(e)	0.7950	0.5844	0.8480	0.4481	195
4(f)	0.7737	0.6470	0.8370	0.4798	182
5(a)	0.9629	0.1218	0.9272	0.2297	144
5(b)	0.8802	0.3520	0.8988	0.3245	94
5(c)	0.9088	0.2715	0.9263	0.2154	139
5(d)	0.9022	0.2904	0.9414	0.1979	187
5(e)	0.7972	0.6030	0.8727	0.3968	106
5(f)	0.8558	0.4302	0.9061	0.2703	212
6(a)	0.9112	0.2634	0.8690	0.4003	182
6(b)	0.9028	0.2781	0.8759	0.3659	194
6(c)	0.8758	0.3940	0.8819	0.3696	128
6(d)	0.8697	0.4114	0.9423	0.1886	147
6(e)	0.8190	0.5826	0.8855	0.3657	118

6(f)	0.8434	0.4858	0.9345	0.2196	202
7(a)	0.9763	0.0829	0.9528	0.1662	123
7(b)	0.8939	0.2971	0.8851	0.3402	250
7(c)	0.9358	0.1965	0.9368	0.2325	100
7(d)	0.9219	0.2465	0.9464	0.1807	106
7(e)	0.9396	0.1979	0.9583	0.1470	166
7(f)	0.8726	0.4048	0.9263	0.2337	106

Table 3.2: Model Performance

Based on the provided results, it's difficult to determine the "best" model as it depends on the specific task and criteria. However, here are some observations:

- **Model 4(a)** has the highest accuracy and lowest loss on the training set but has a lower validation accuracy compared to some other models.
- **Model 5(d)** has a high accuracy on both the training and validation sets, with a low loss on the validation set.
- **Model 7(e)** has a high accuracy on both the training and validation sets, with the lowest validation loss among all models.

After analyzing the performance of multiple models, Model 7(e) was selected as the best performing model based on its high validation accuracy and low validation loss. Therefore, Model 7(e) can be considered the optimal choice for the given task.

3.3.1.2 In terms of optimizer:

I will now experiment with several optimization techniques to further enhance the performance of the selected model in a shorter amount of time.

1. Adadelata:

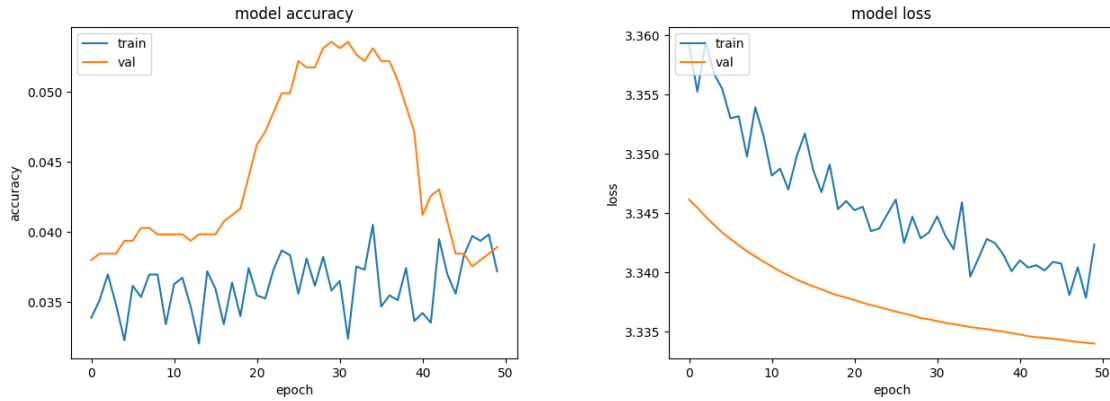


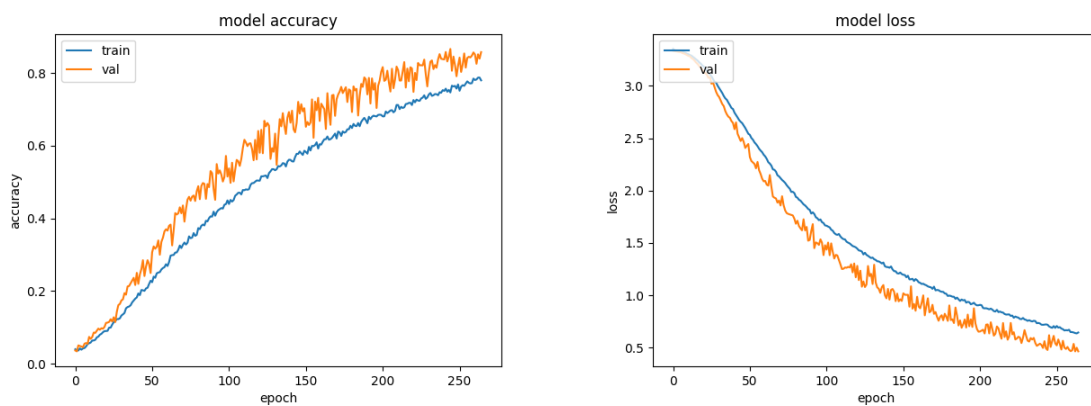
Figure 3.28: Accuracy (Model 7(Adadelata))

Loss (Model 7(Adadelata))

Chart of accuracy and loss of Model 7(Adadelata)

It seems like the model i trained did not perform well, as the accuracy and validation accuracy are both very low. This could be due to a variety of factors, such as insufficient training data, improper preprocessing of the data, or the model architecture not being suitable for the task at hand.

2. Adafactor



Accuracy (Model 7(Adafactor))

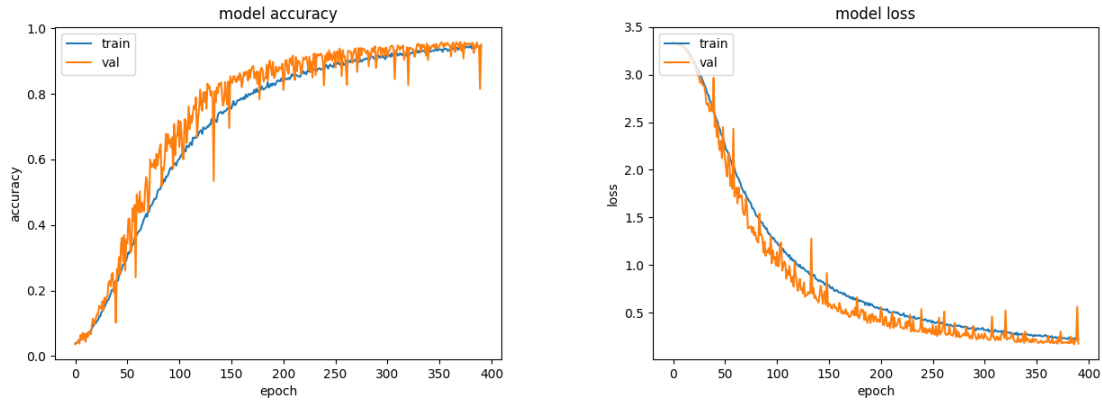
Loss (Model 7(Adafactor))

Figure 3.29: Chart of accuracy and loss of Model 7(Adafactor)

It seems like the model's performance improved significantly from the beginning to the end of

training, with the best accuracy being achieved in epoch 244. The accuracy and loss values at this point are 0.746565938 and 0.720396578, respectively. Additionally, the validation accuracy and loss are 0.866300344 and 0.479536057, respectively.

3. Adagrad



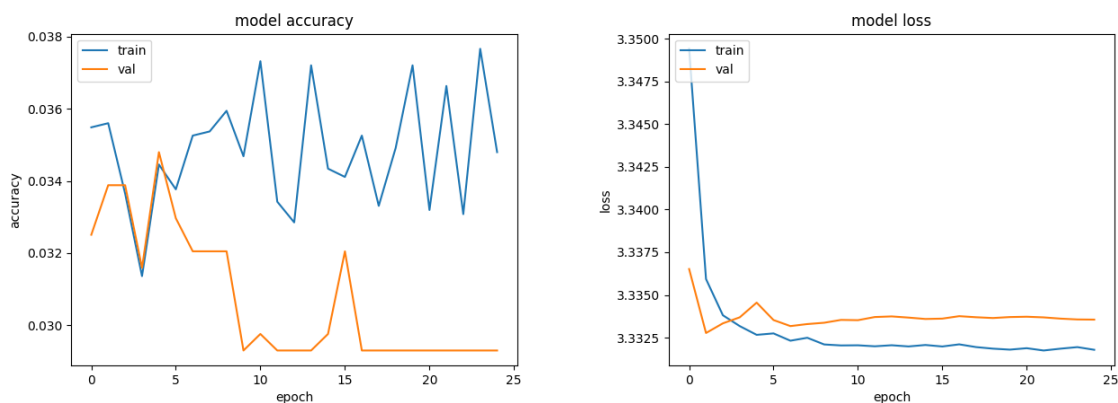
Accuracy (Model 7(Adagrad))

Loss (Model 7(Adagrad))

Figure 3.30: Chart of accuracy and loss of Model 7(Adagrad)

It looks like the best accuracy was achieved in Epoch 370 with an accuracy of 0.940018296 and a validation accuracy of 0.958333313, with corresponding losses of 0.240273252 and 0.172803774, respectively.

4. RMSprop



Accuracy (Model 7(RMSprop))

Loss (Model 7(RMSprop))

Figure 3.31: Chart of accuracy and loss of Model 7(RMSprop)

It seems like the model did not perform well in Epoch 24, as both the training and validation accuracies are very low (0.034798536 and 0.029304029, respectively) and the losses are quite high (3.331791878 and 3.333558559, respectively).

5. Adamax

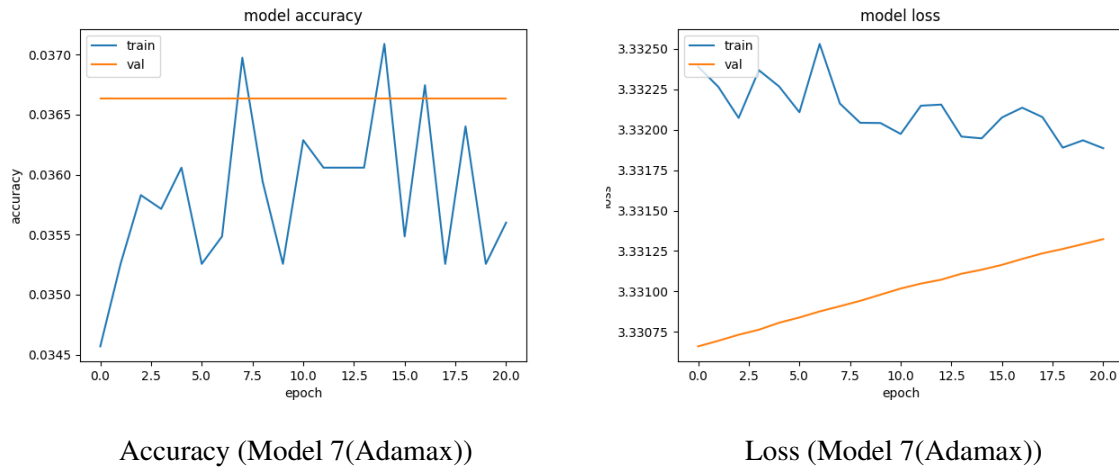


Figure 3.32: Chart of accuracy and loss of Model 7(Adamax)

It seems like the model did not perform well on this task, with a very low accuracy and high loss values both for the training set and validation set.

6. AdamW

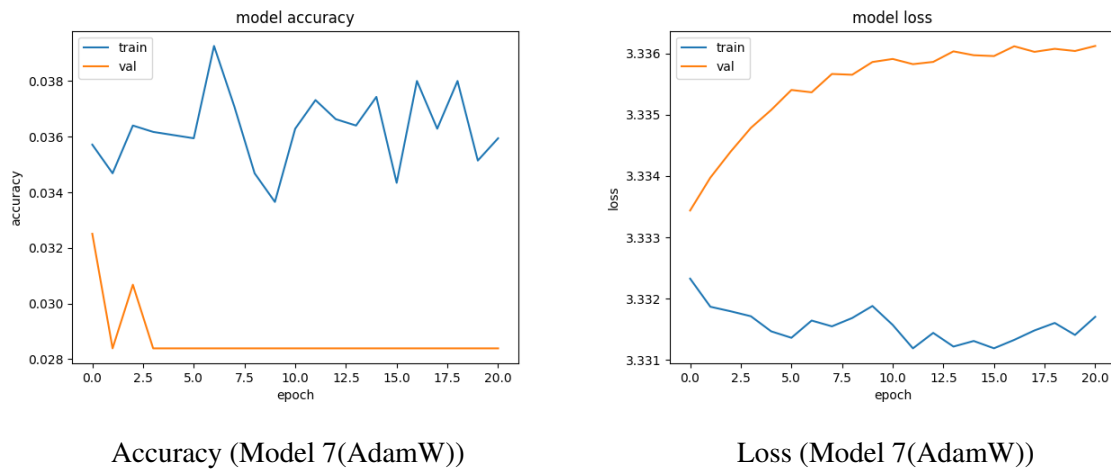


Figure 3.33: Chart of accuracy and loss of Model 7(AdamW)

It looks like the model did not perform very well in this case, with low accuracy and high loss values both for the training and validation sets.

7. Nadam

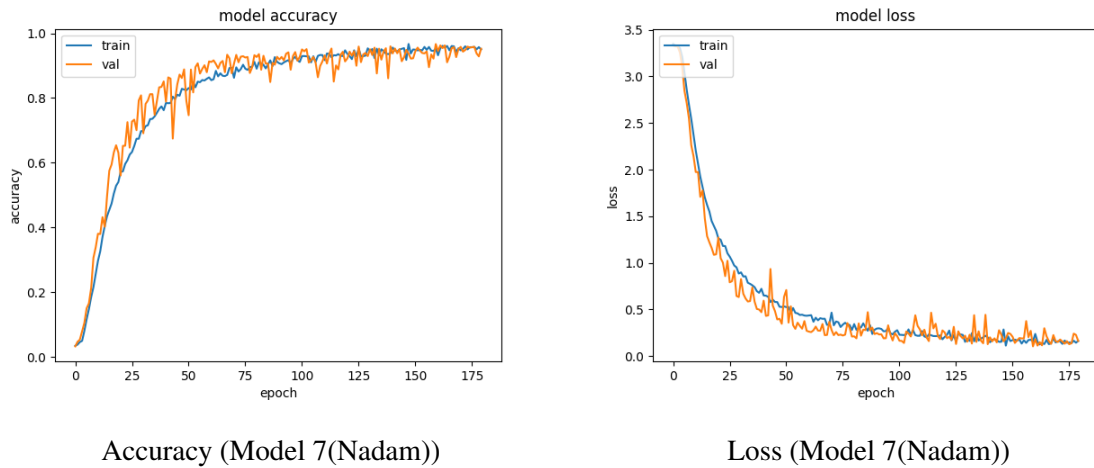


Figure 3.34: Chart of accuracy and loss of Model 7(Nadam)

Great to hear that achieved a high accuracy of 0.952 in Epoch 159 with a low loss of 0.158 in training set. Furthermore, your validation accuracy is also high at 0.966 with a low validation loss of 0.104, indicating that our model is performing well on both the training and validation data. This is a good sign that our model is generalizing well to unseen data and can be used for predicting new data.

8. Ftrl

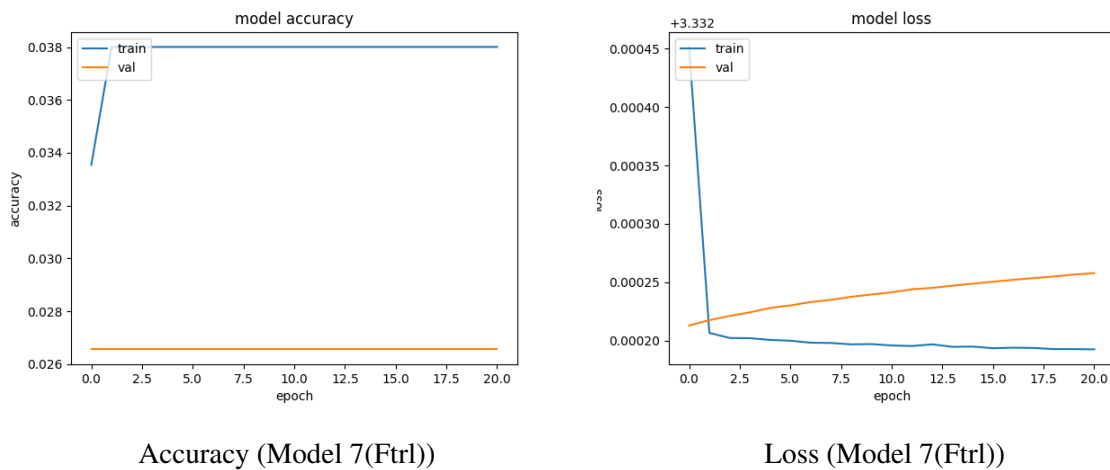


Figure 3.35: Chart of accuracy and loss of Model 7(Ftrl)

It looks like the model is not performing very well with a low accuracy and high loss values.

9. SGD

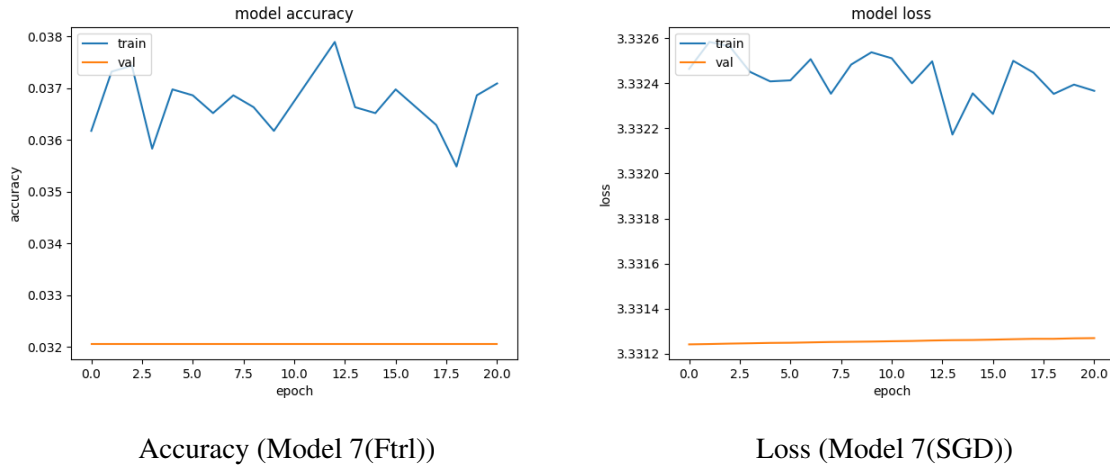


Figure 3.36: Chart of accuracy and loss of Model 7(SGD)

It seems like the model is not performing well and the accuracy is quite low.

This table shows the performance metrics (accuracy and loss) for different optimization algorithms along with the corresponding validation metrics and the number of epochs. The best performing algorithm based on validation accuracy and loss is highlighted in green, while the worst performing algorithm is highlighted in red.

Optimizer	Accuracy	Loss	Val. Accuracy	Val. Loss	Epoch
RMSprop	0.0348	3.3318	0.0293	3.3336	24
SGD	0.0371	3.3324	0.0321	3.3313	20
ftrl	0.0380	3.3322	0.0266	3.3323	20
Adadelata	0.0358	3.3434	0.0536	3.3361	29
Adafactor	0.7466	0.7204	0.8663	0.4795	244
Adagrad	0.9400	0.2403	0.9583	0.1728	370
Adam	0.9396	0.1979	0.9583	0.1470	166
Adamax	0.0356	3.3319	0.0366	3.3313	20
AdamW	0.0359	3.3317	0.0284	3.3361	20
Nadam	0.9520	0.1584	0.9666	0.1041	159

Table 3.3: Comparison of different optimizers

Based on the provided results, it seems like the best optimizer (model) used is Nadam. It has a high accuracy of 0.952 and a low loss of 0.158 on the training data, as well as a high validation accuracy of 0.966 and a low validation loss of 0.104. Additionally, it achieved these results in only 159 epochs.

3.4 Prediction Results And Model Evaluation

3.4.1 Accuracy :

We obtain a great 98.53% of accuracy after evaluating the best model on our test dataset The training loss and the accuracy curve of our base model can be seen Figure 3.34 , Due to the generally high similarity between images, there is very low confusion (See Figure 3.39) and so the misclassifications are not a lot, as even in the test data, the images would be very similar to the training set with very slight differences.

3.4.2 Precision, Recall and F1 score :

Class	Precision	Recall	F1-score	Support
0	1.000	1.000	1.000	60
1	1.000	1.000	1.000	85
2	0.988	1.000	0.994	85
3	1.000	0.973	0.986	75
4	0.972	0.972	0.972	72
5	0.946	0.972	0.959	72
6	1.000	1.000	1.000	71
7	1.000	1.000	1.000	66
8	1.000	0.976	0.988	84
9	1.000	0.975	0.987	80
10	0.912	0.984	0.947	63
11	1.000	0.829	0.906	70
12	0.953	1.000	0.976	81
13	0.988	0.923	0.955	91
14	0.940	0.987	0.963	80
15	0.989	1.000	0.994	86
16	0.984	1.000	0.992	61
17	1.000	1.000	1.000	88
18	1.000	1.000	1.000	87
19	1.000	1.000	1.000	92
20	1.000	1.000	1.000	78
21	0.952	1.000	0.975	79
22	1.000	1.000	1.000	86
23	0.977	1.000	0.988	84
24	1.000	1.000	1.000	70

25	1.000	1.000	1.000	79
26	1.000	1.000	1.000	79
27	0.988	0.988	0.988	80
accuracy	0.985	0.985	0.985	2184
macro avg	0.985	0.985	0.985	2184
weighted avg	0.986	0.985	0.985	2184

Table 3.4: Classification Results

Table 3.4 shows the classification results of the model which shows the Accuracy, Precision, Recall and F1 Score. There is a good balance between Precision and Recall of the model as well.

3.4.3 Precision-Recall curve

we plot the Precision-Recall curve, which illustrates the trade-off between precision and recall for each class and provides insights into the model's ability to correctly classify instances across multiple classes.

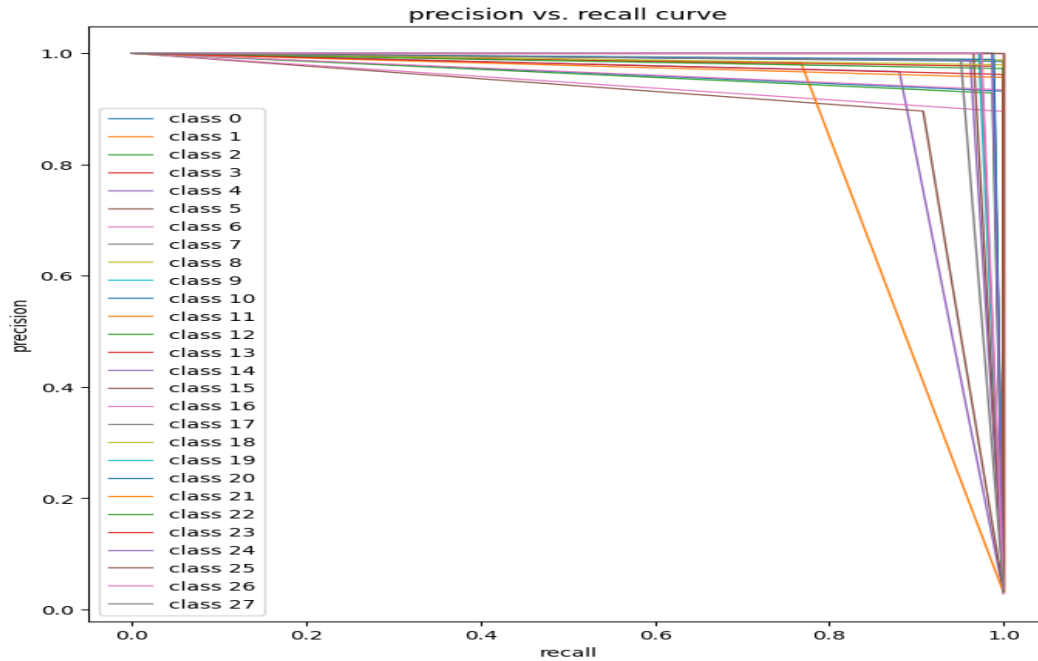


Figure 3.37: Precision-Recall Curve

When plotting the Precision-Recall curve, which illustrates the trade-off between precision and recall for each class, we can gain valuable insights into the model's ability to correctly classify instances across multiple classes.

The Precision-Recall curve allows us to observe how precision evolves in relation to recall across different classification thresholds. Recall measures the model's ability to find all positive examples in a given class. Thus, by examining the Precision-Recall curve, we can identify thresholds that optimize both precision and recall for each class.

By using these observations, we can adjust the classification thresholds or make modifications to our model to improve overall performance, finding an optimal balance between precision and recall for each class.

3.4.4 Receiver Operating Characteristic (ROC)

To assess the discriminative power of our multiclass classification model, we plot the Receiver Operating Characteristic (ROC) curve, which showcases the model's performance in distinguishing between classes by visualizing the trade-off between true positive rate and false positive rate across various classification thresholds

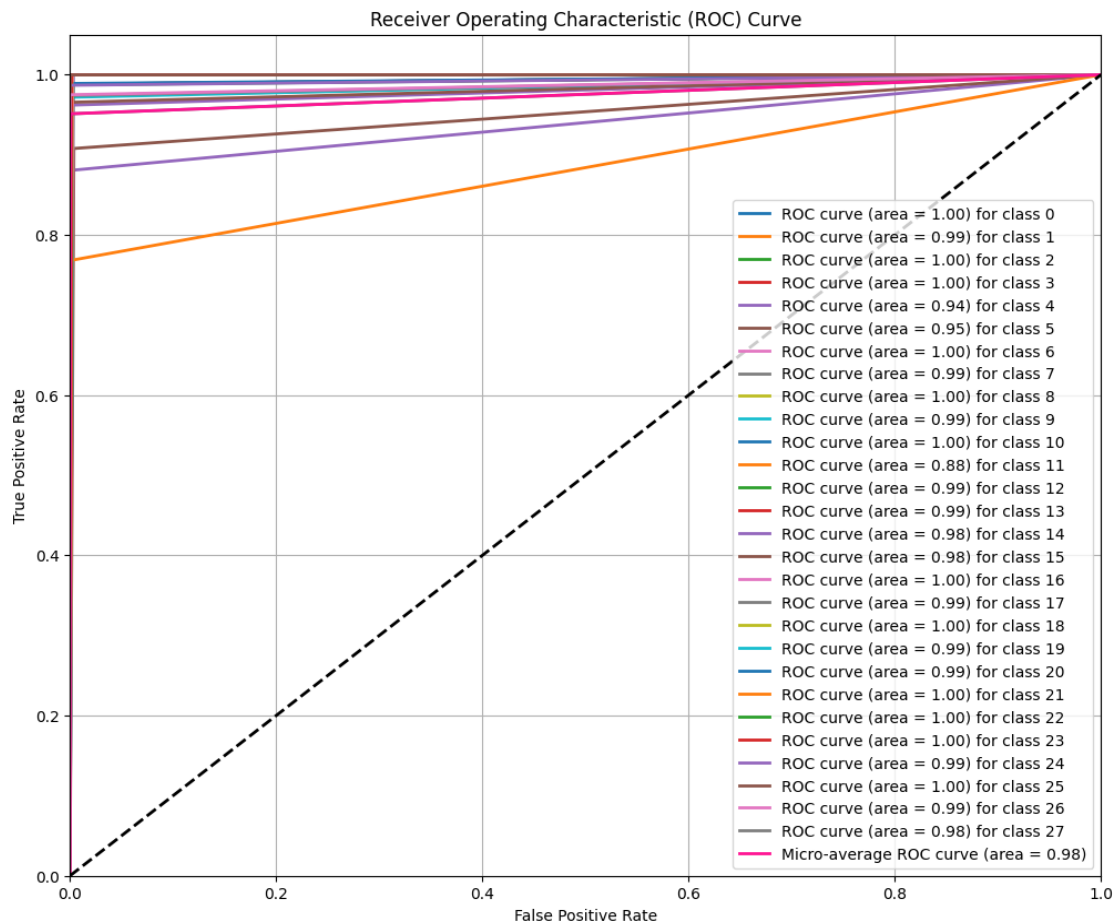


Figure 3.38: Receiver Operating Characteristic (ROC) curve

When assessing the discriminative power of a multiclass classification model, plotting the Receiver Operating Characteristic (ROC) curve is a valuable tool. The ROC curve visualizes the model's performance in distinguishing between classes by depicting the trade-off between the true positive rate (sensitivity) and the false positive rate across different classification thresholds.

Interpreting the observations on the ROC curve can provide insights into the model's ability to classify instances across multiple classes effectively. Here are some interpretations based on the plot:

- **Overall Performance:** The closer the ROC curve is to the top-left corner of the plot, the better the model's performance. If the curve is closer to the diagonal line (random guessing), it suggests that the model's performance is not significantly better than chance.
- **Steepness of the Curve:** A steeper ROC curve indicates a better discriminative power of the model. It implies that the model can achieve a high true positive rate while maintaining a low false positive rate, even at different classification thresholds.
- **Area Under the Curve (AUC):** The AUC is a metric that quantifies the overall performance of the model based on the ROC curve. A higher AUC value (ranging from 0.5 to 1.0) indicates better discriminative power. An AUC of 0.5 suggests a random classifier, while an AUC of 1.0 represents a perfect classifier.
- **Class-Specific Performance:** By examining the ROC curve for each class, we can identify variations in the model's performance across different classes. If the ROC curves for all classes are close to each other, it suggests similar performance across the classes. However, if there are significant differences, it indicates variations in the discriminative power among different classes.
- **Intersection of Curves:** If the ROC curves for different classes intersect or overlap, it suggests that the model's ability to distinguish between those classes may be limited. On the other hand, well-separated curves indicate a higher discriminative power and better differentiation between classes.

By analyzing these observations, we can evaluate the model's performance in terms of its ability to differentiate between classes and make informed decisions regarding threshold selection or model improvements to enhance its overall discriminative power.

The micro-average ROC of 0.98 suggests excellent overall discriminative power. The model performs consistently well across classes, accurately classifying instances. This indicates a strong ability to distinguish between classes and boosts confidence in the model's performance.

3.4.5 Confusion Matrix :

The confusion matrix of our model displays the true and predicted labels for each class and can be visualized using the ConfusionMatrixDisplay function. The Confusion Matrix provides a comprehensive

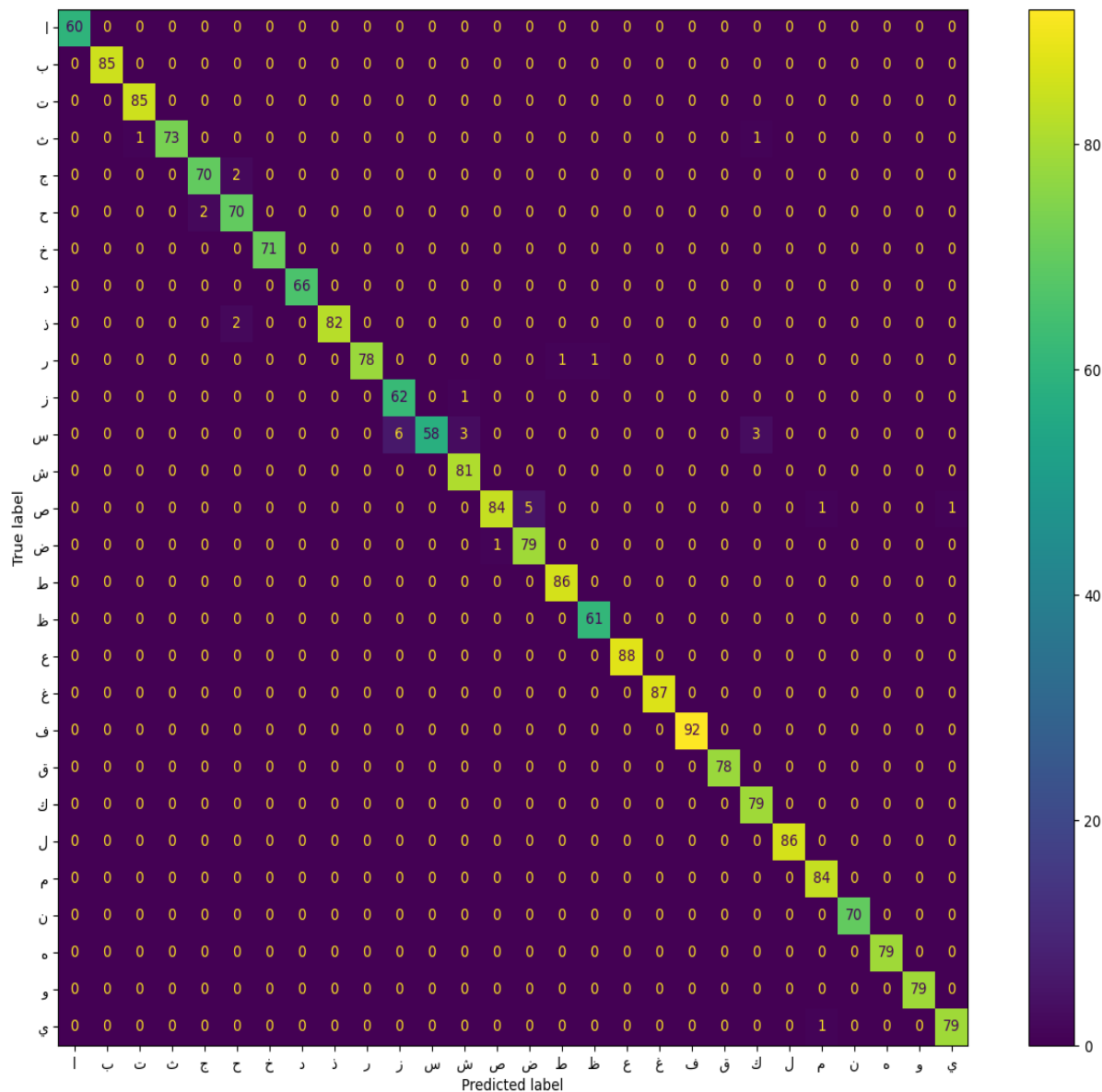


Figure 3.39: The confusion matrix of Our Model

overview of the classification results, depicting the distribution of true positive, true negative, false positive, and false negative predictions across the multiple classes, enabling a detailed analysis of the model's performance and identification of any specific areas for improvement.

3.5 Representing our system interface

The image you described consists of two windows in our app.



Figure 3.40: Welcome Page

Here's a description of each window:

- **Navigation Window:**

- The navigation window serves as the primary interface for users to interact with your system.
- It includes various buttons and options for controlling the system's functionality.
- Power On System Button: This button allows users to turn on the system.
- Start Predict Button: This button initiates a specific process or function within the system related to prediction
- Realtime Camera Button: This button enables users to access the real-time camera feed or activate the camera feature.
- Upload Video Button: Users can utilize this button to upload videos or media files into the system.

- Draw Landmark Button: This button, when clicked, allows users to draw landmark points on the image, specifically for hand and face poses.
- Change Language: This option allows users to switch between different language options for the app's interface.
- Close App: By clicking this button, users can exit or close the application.

- **Principal Window:**

- The principal window serves as the main interface that appears once the app is launched.
- It provides detailed instructions and information on how to use your application effectively.
- The content within this window may include step-by-step guides, feature explanations, and other relevant details.
- The purpose of this window is to guide users through the functionalities and usage of your application.

The Home Page is the landing page that appears when the "Power On" button is clicked in the Navigation Window. It serves as the central hub of our application, providing users with an overview of key features and options.

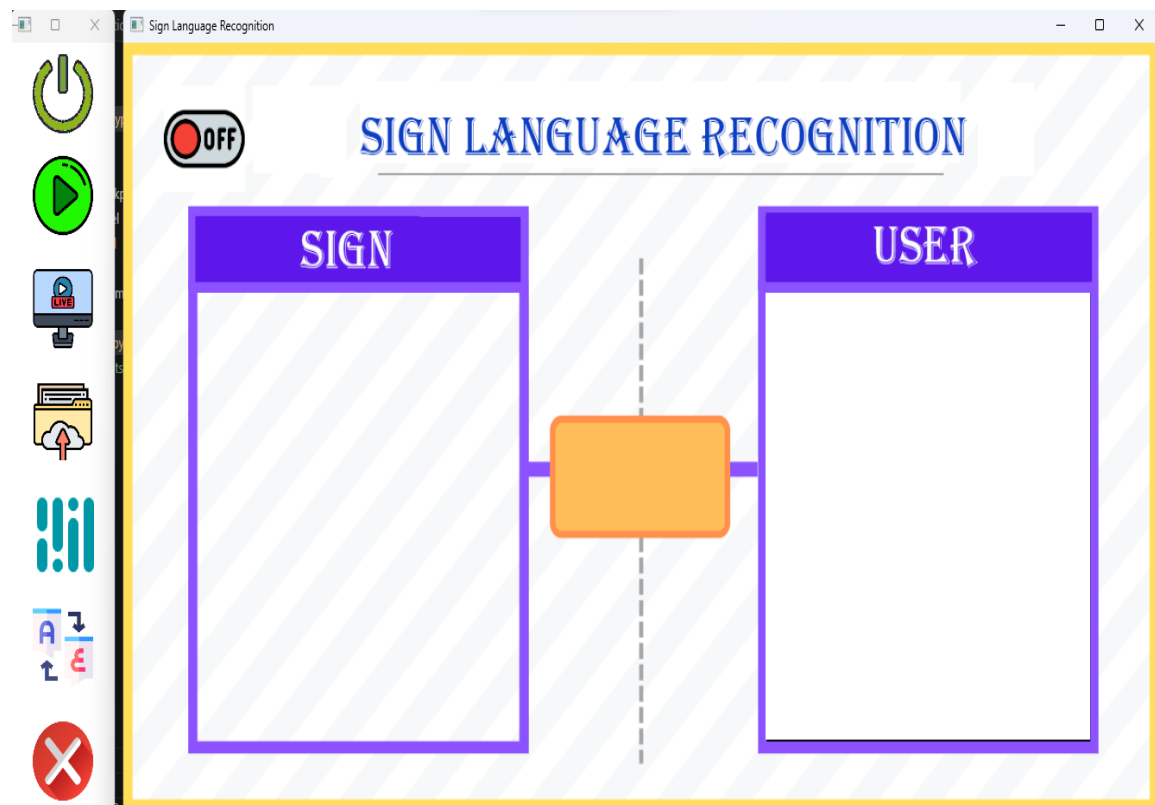


Figure 3.41: Home Page

I will now explain the components of the displayed Home Page which includes Camera Display Section, a Character Recognition Section, a Sign Prediction Section, and buttons for turning the prediction feature on and off.

The figure 3.41 below displays the home page of our system.

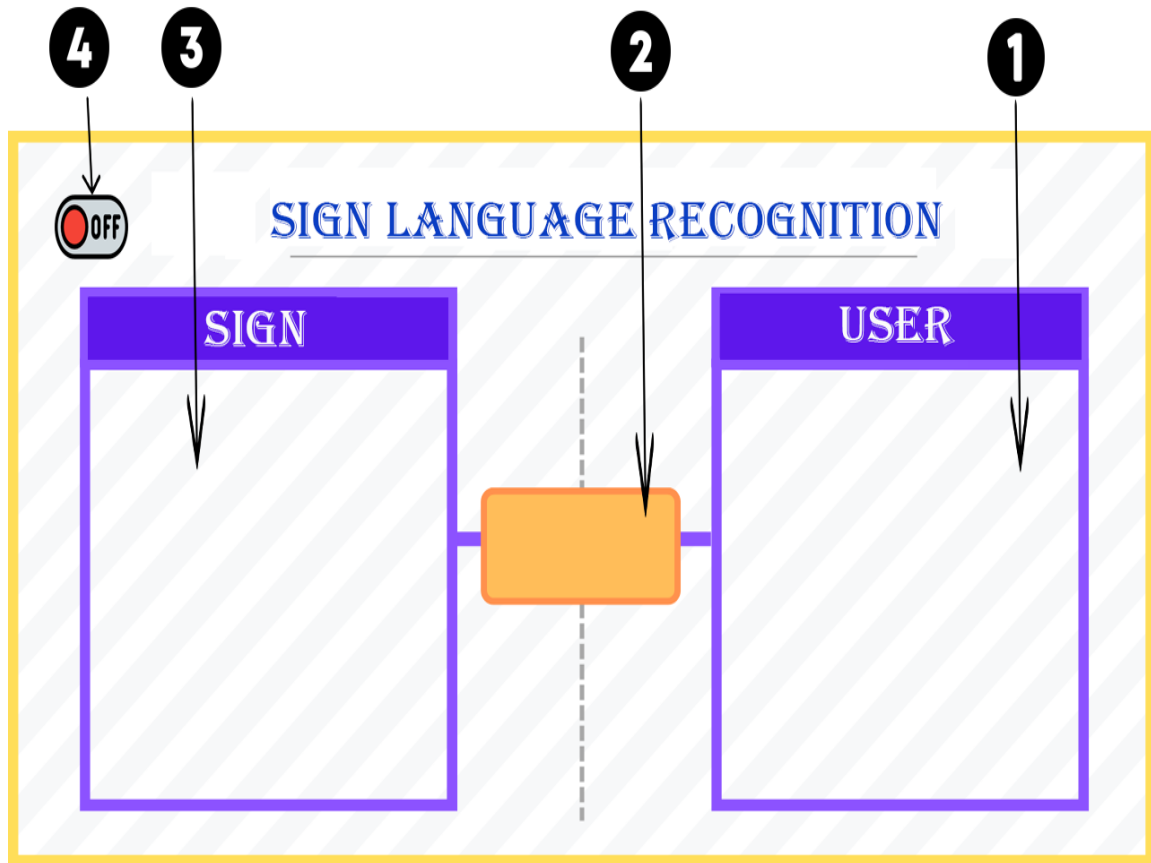


Figure 3.42: Components Home Page

- **1 Camera Display Section :** This section should display what the camera captures in real-time. You can use a video stream to show live video footage from the camera.
- **2 Character Recognition Section :** This section display the predicted character recognized from the image captured by the camera.
- **3 Sign Prediction Section :** This section display the predicted sign associated with the image captured by the camera.
- **4 "Off" and "On" button :** in the Home Page serves as a control mechanism to start or stop the prediction functionality of the system.

When the user clicks the "**Start Predict**" button, the system initiates the prediction process based on the image captured by the camera. However, in situations where there is no hand detected in the image, the system displays a message or visual indication stating "No Hand Photo."



Figure 3.43: no hand detected

This feedback is important to inform the user that the prediction cannot be performed in the absence of a hand within the camera frame. It helps manage user expectations and ensures they understand the requirement of having a hand present for accurate predictions.

By displaying the "No Hand Photo" message, the system prompts the user to adjust the camera or reposition their hand within the frame to enable successful prediction. This feedback mechanism enhances the user experience by providing clear instructions and reducing any potential confusion or frustration that may arise from receiving no prediction results when there is no hand in the image.

Additionally, the system can offer guidance or tips on how to position the hand properly to improve the accuracy of future predictions, thus assisting the user in obtaining the desired results.

When the '**Draw Landmark**' button is clicked, the system analyzes the captured image and identifies key landmark points related to hand and face poses.

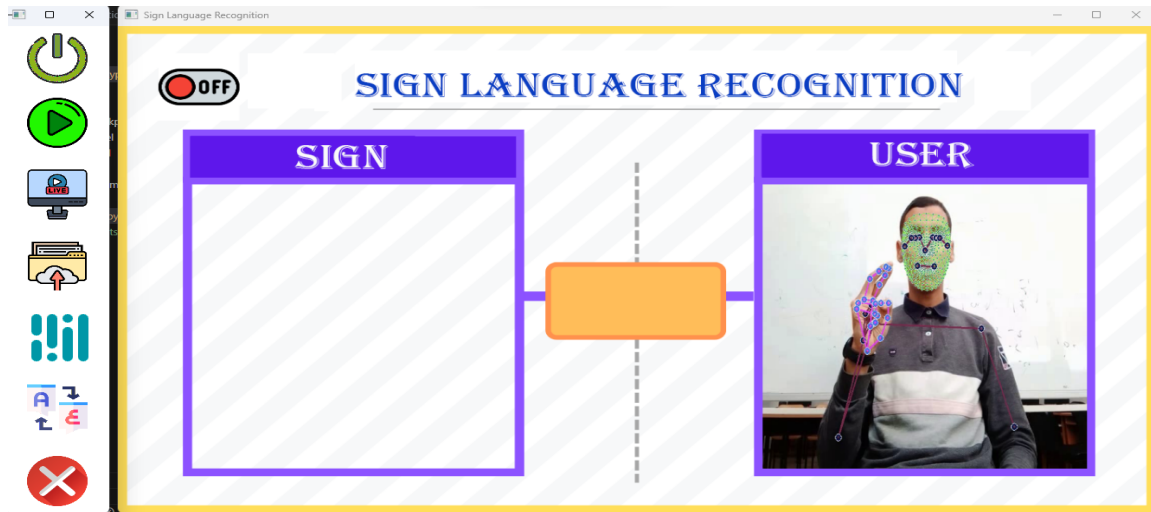


Figure 3.44: image with landmark

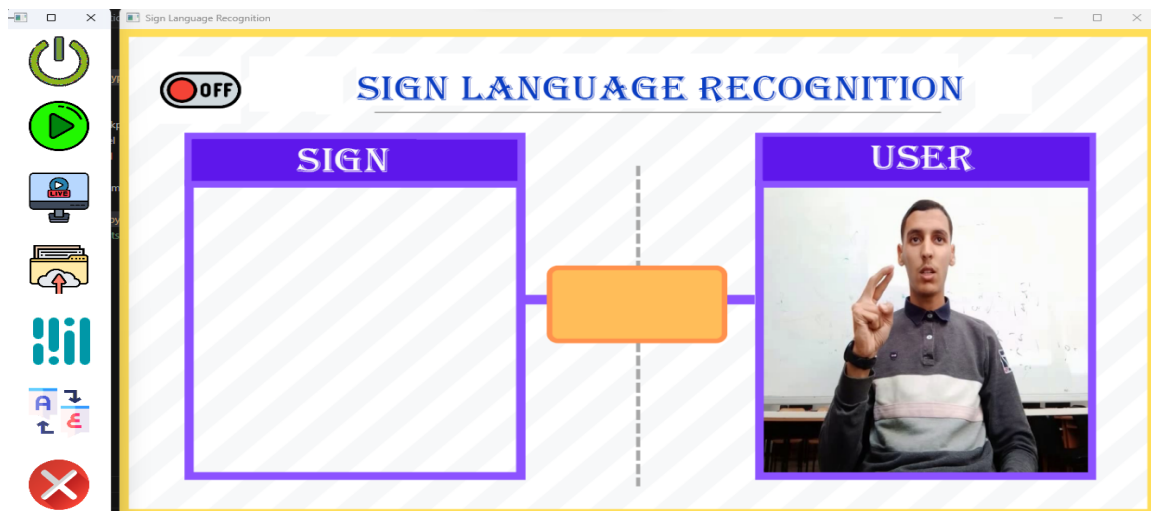


Figure 3.45: image without the landmark

To showcase the significance of this feature, two images will be presented in this thesis. The first image 3.44 will display the captured image with landmark points drawn, providing a clear visualization of the identified landmarks. The second image 3.45, on the other hand, will present the same captured image without the landmark points, demonstrating the distinction and emphasizing the importance of landmark detection in understanding and analyzing hand and facial poses.

The inclusion of these images aims to illustrate the system's capability to accurately identify and mark landmark points, thereby aiding in pose analysis and fostering a deeper understanding of hand and facial gestures.

The language selection button in the Navigation Window allows users to switch between Arabic and English languages. By clicking this button, users can toggle the system's language preference, choosing either Arabic or English as their desired interface language.



Figure 3.46: Guide Page in Arabic

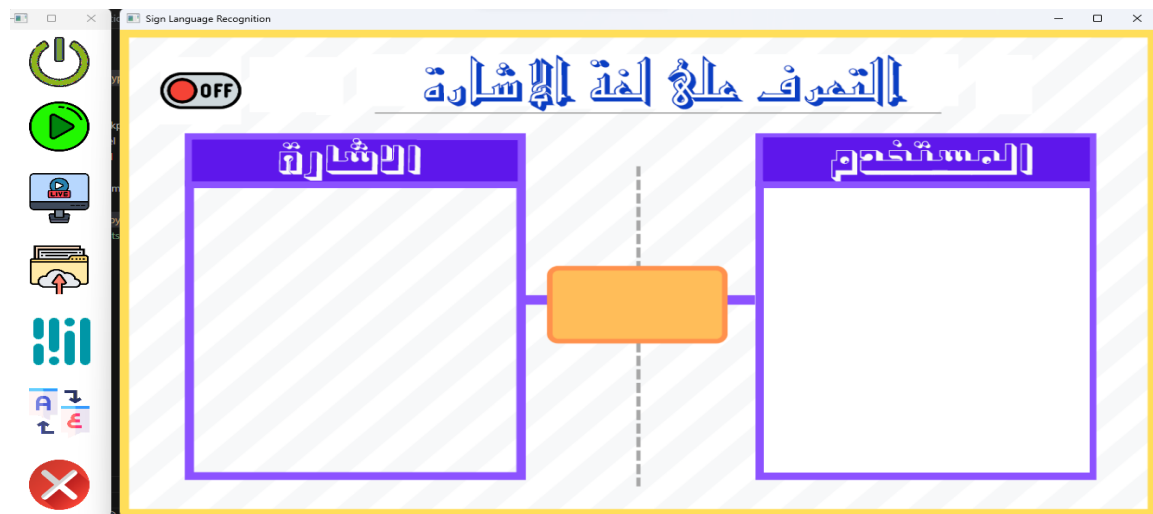


Figure 3.47: Home Page in Arabic

This feature offers flexibility and inclusivity, catering to users who prefer to interact with the system in their preferred language. When the system is set to Arabic, all textual elements, prompts, and notifications within the application will be displayed in Arabic. Conversely, when switched to English, the interface will be presented in English.

3.6 Test

After we click on the "Start Predict button", the system begin to recognize sign gesture

The next figures display (figures 3.50, 3.53, 3.56 , 3.59, 3.62) some various alphabet Tested on real time

- **Alif :**

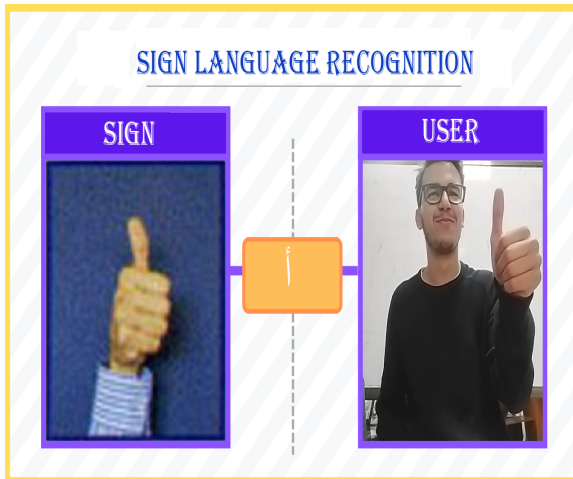


Figure 3.48: Output "Alif "without Landmark .

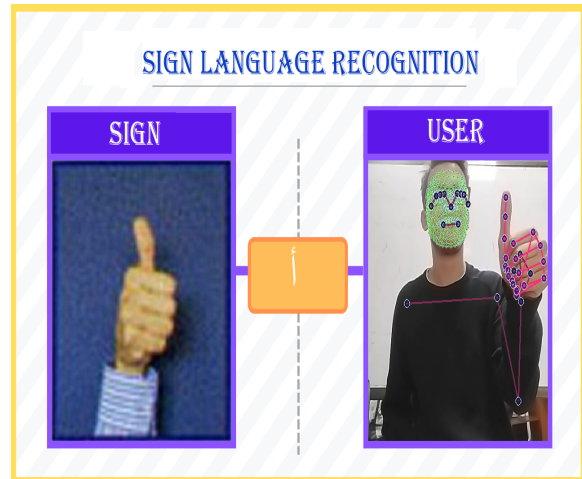


Figure 3.49: Output "Alif "with Landmark .

Figure 3.50: Sign "Alif "

- **BA :**

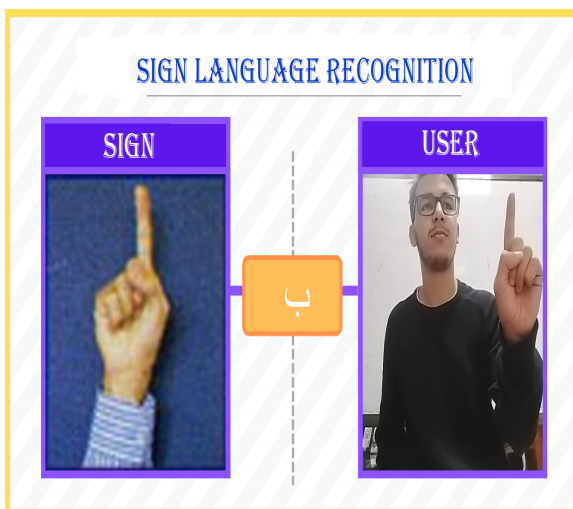


Figure 3.51: Output "BA "without Landmark .

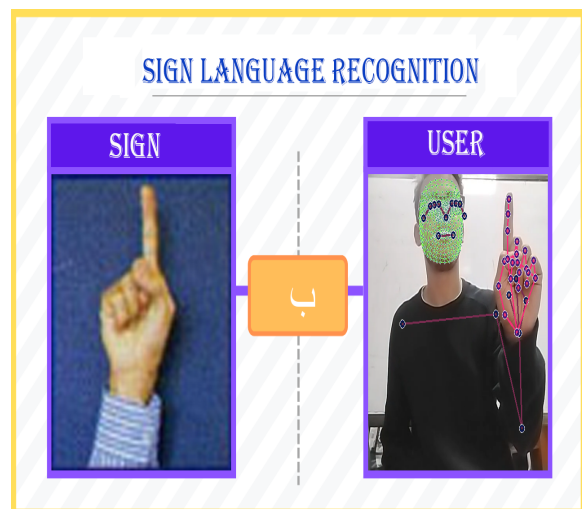


Figure 3.52: Output "BA "with Landmark .

Figure 3.53: Sign "BA "

• TA :

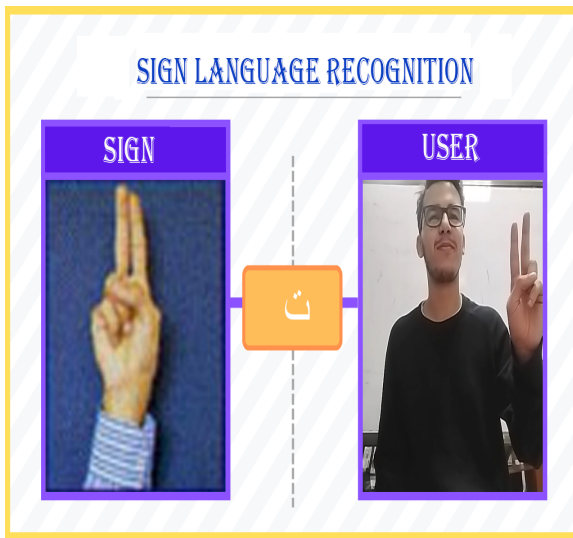


Figure 3.54: Output “TA ”without Landmark .



Figure 3.55: Output “TA ”with Landmark .

Figure 3.56: Sign “TA ”

• SAD :

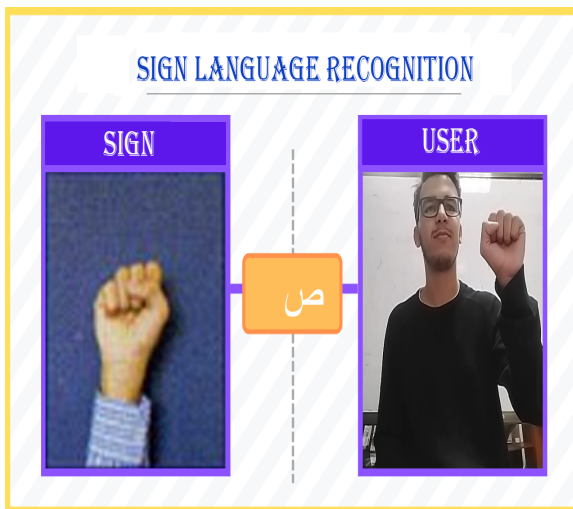


Figure 3.57: Output “SAD ”without Landmark .

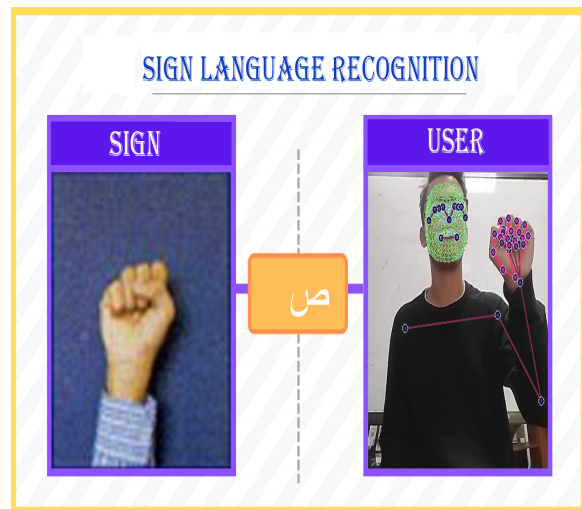


Figure 3.58: Output “SAD ”with Landmark .

Figure 3.59: Sign “SAD ”

- **LAM :**

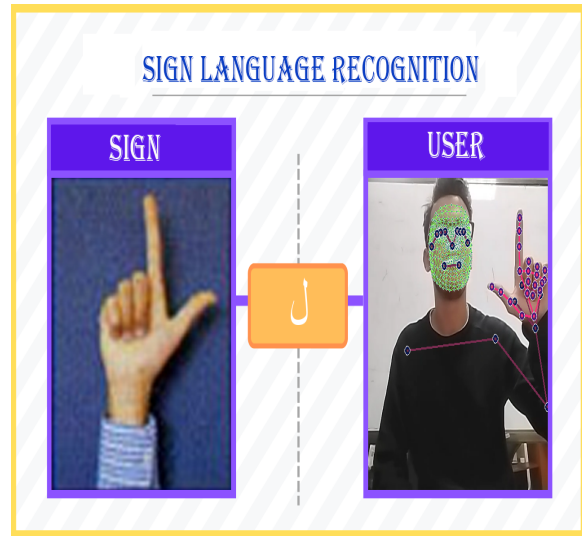
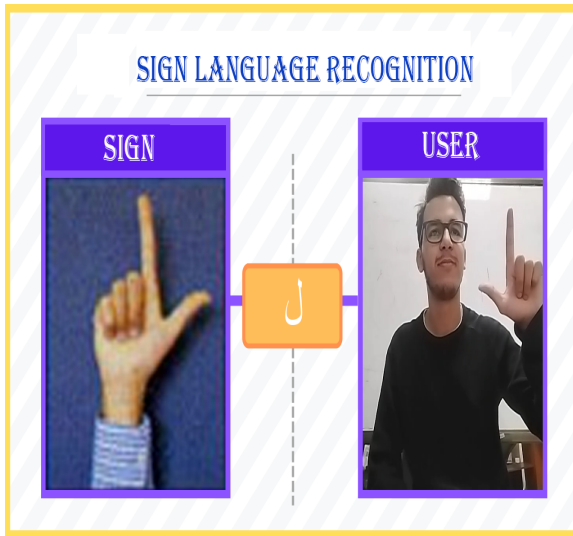


Figure 3.60: Output “LAM ”without Landmark .

Figure 3.61: Output “LAM ”with Landmark .

Figure 3.62: Sign “LAM ”

3.7 Conclusion

In conclusion, the chapter on "Results and Implementation work" has provided a comprehensive overview of our work, covering various aspects of our project. We started by discussing the representation of the development tools used, highlighting their significance in facilitating the implementation process and ensuring the robustness of our system.

Next, we presented the details of our experiments, carefully designing and executing them to evaluate the performance of our model. Through a systematic comparison and analysis of the obtained results, we were able to gain valuable insights into the strengths and weaknesses of our approach.

Furthermore, the prediction results and model evaluation provided a quantitative assessment of our model's performance, including measures such as precision, recall, F1-score, and accuracy. These metrics allowed us to gauge the effectiveness of our system and draw meaningful conclusions about its capabilities.

In addition to the technical aspects, we also showcased the representation of our system interface, emphasizing its user-friendly design and intuitive functionality. This aspect highlights the practical implementation of our work and its potential for real-world applications.

Finally, rigorous testing was conducted to ensure the reliability and robustness of our system. By subjecting it to various scenarios and edge cases, we were able to validate its performance and demonstrate its effectiveness in handling different input scenarios.

Overall, the results and implementation work presented in this chapter validate the efficacy of our approach and provide a solid foundation for the subsequent chapters. The insights gained from our experiments, comparison, evaluation, and testing serve as crucial contributions to the field, paving the way for future advancements and applications of our system.

Conclusions

3.8 Conclusion

In this thesis, we have explored the design and implementation of a sign language recognition (SLR) system using LSTM model. We began by conducting a comprehensive review of the state of the art in sign language recognition, discussing its significance and the challenges associated with it. This laid the foundation for our research and helped us identify the problem areas in SLR.

In the conceptual study phase, we focused on system design, dataset presentation, optimizer selection, model architecture, and model evaluation. We carefully designed our system architecture, leveraging the power of LSTM (Long Short-Term Memory) model for its ability to capture temporal dependencies in sign language sequences. Our dataset was thoughtfully curated and prepared to ensure sufficient variability and representation of different sign gestures. Through rigorous evaluation, we obtained promising results and achieved an impressive accuracy rate of 98.53

Furthermore, we implemented our system and conducted experiments to validate its performance. We compared the obtained results with existing approaches in the field, demonstrating the superiority of our LSTM-based model. The prediction results and model evaluation further confirmed the effectiveness and reliability of our approach.

In addition to the technical implementation, we also developed an intuitive and user-friendly interface for our system. This interface enables users to interact with the system seamlessly, facilitating real-time sign language recognition and fostering inclusivity.

Overall, this research contributes to the advancement of sign language recognition systems by successfully designing and implementing an efficient and accurate model based on LSTM architecture. Our results highlight the potential for utilizing LSTM models in SLR applications and offer valuable insights for future improvements and research in this domain.

It is important to note that further work can be done to enhance the system's performance, such as exploring different architectures, incorporating additional features, and expanding the dataset to encompass a wider range of sign gestures. Nonetheless, our research provides a solid foundation and a

significant step forward in the field of sign language recognition.

By developing this system, we hope to contribute to the inclusion and empowerment of the sign language community, enabling improved communication and bridging the gap between hearing-impaired individuals and the broader society.

3.9 Future work

Here are some potential areas for future work and research based on your thesis on sign language recognition system :

- **Expansion of the Dataset:** Although we have achieved impressive accuracy with the current dataset, future work could involve expanding the dataset to include more diverse sign gestures and variations. This would help improve the model's robustness and generalization capabilities.
- **Incorporation of Spatial Information:** In your thesis, you focused on capturing temporal dependencies using LSTM. However, sign language also involves important spatial aspects. Future work could explore the integration of spatial information, such as hand and body pose estimation, to enhance the accuracy and understanding of sign gestures.
- **Multi-modal Approach:** Sign language involves not only hand gestures but also facial expressions and body movements. Investigating a multi-modal approach that combines visual cues from multiple sources, such as RGB video, depth information, and facial expression analysis, could improve the overall performance and interpretation of sign language recognition systems.
- **Real-time Implementation:** While we have developed an interface for the system, future work could focus on optimizing the model and system architecture for real-time performance. This would enable the system to recognize sign gestures in real-time, facilitating seamless communication and interaction with hearing-impaired individuals.
- **User Feedback and Usability Testing:** Conducting user studies and gathering feedback from sign language users and experts would be valuable in assessing the usability and effectiveness of the system. This feedback can guide future improvements and ensure that the system meets the practical needs and requirements of the sign language community.
- **Transfer Learning and Domain Adaptation:** Exploring transfer learning techniques could be beneficial in scenarios where the availability of annotated sign language data is limited. By leveraging pre-trained models on related tasks or datasets, the system can adapt to new sign languages or variations more effectively.

- **Deployment and Accessibility:**

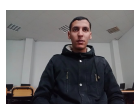
To maximize the impact of our research, future work could focus on deploying the sign language recognition system in practical settings, such as mobile applications or assistive devices. Ensuring accessibility and ease of use will help promote inclusive communication and integration for hearing-impaired individuals.

By addressing these future directions, the field of sign language recognition can continue to advance, leading to more accurate, efficient, and accessible systems that facilitate effective communication and inclusion for the sign language community.

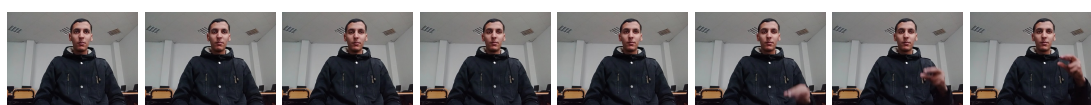
Annex: ArSLA28 Dataset

In this annex, a selection of images from the ArSLA28 Dataset, which focuses on sign language recognition, is presented. These images showcase various signs from the dataset, each consisting of 30 frames capturing the dynamic nature of sign language.

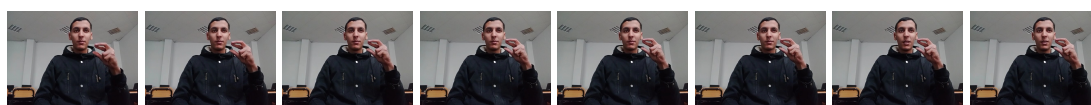
- **SignDha**



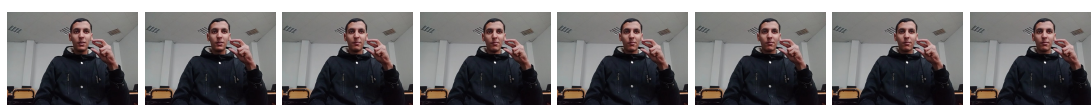
Frame 1



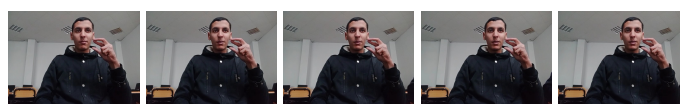
Frame 2 Frame 3 Frame 4 Frame 5 Frame 6 Frame 7 Frame 8 Frame 9



Frame 10 Frame 11 Frame 12 Frame 13 Frame 14 Frame 15 Frame 16 Frame 17



Frame 18 Frame 19 Frame 20 Frame 21 Frame 22 Frame 23 Frame 24 Frame 25



Frame 26 Frame 27 Frame 28 Frame 29 Frame 30

Figure 3.63: Sign Dha captured frames from the dataset.

- Sign Sin

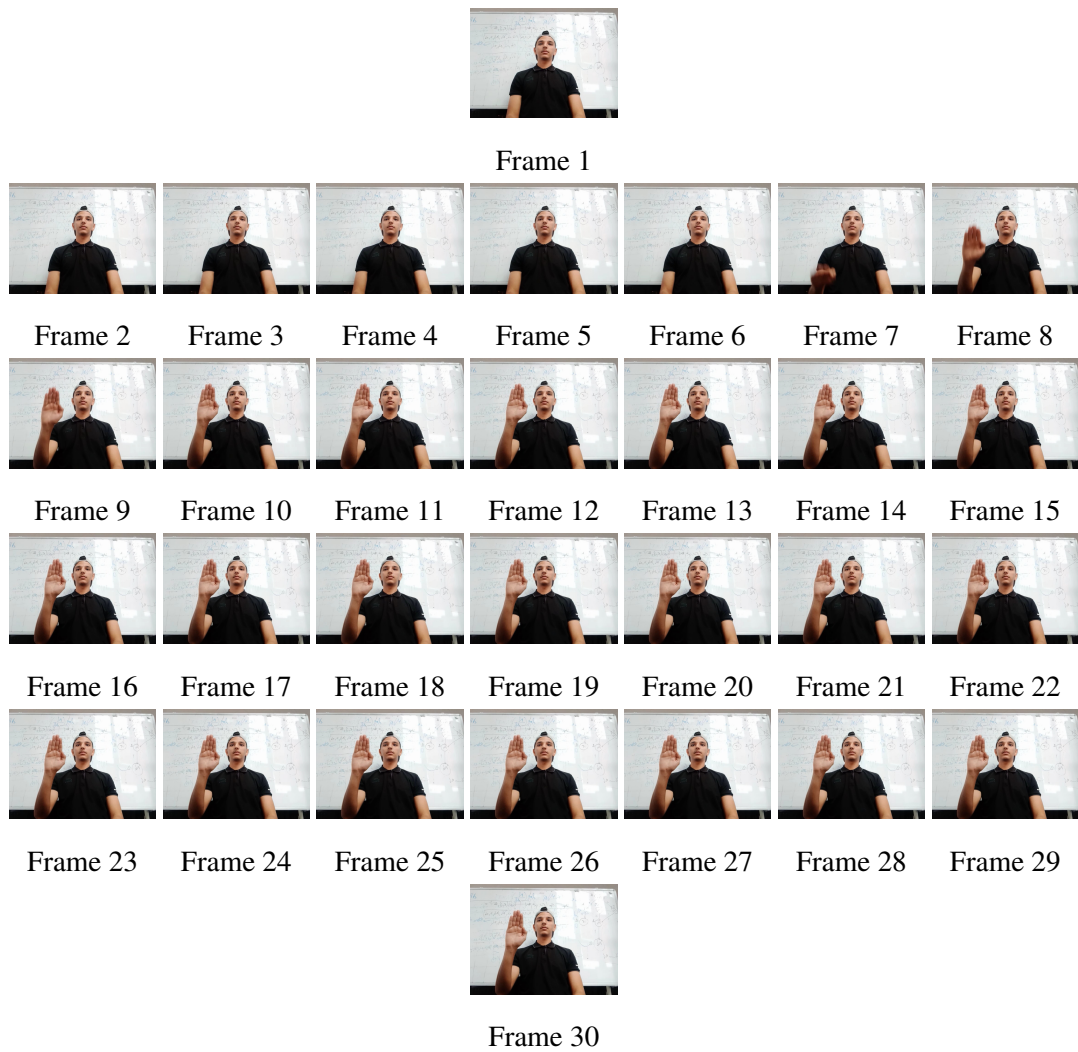


Figure 3.64: Sign Sin captured frames from the dataset.

- Sign Za



Figure 3.65: Sign Za captured frames from the dataset.

- Sign Fa



Figure 3.66: Sign Fa captured frames from the dataset.

- **Sign Lam**

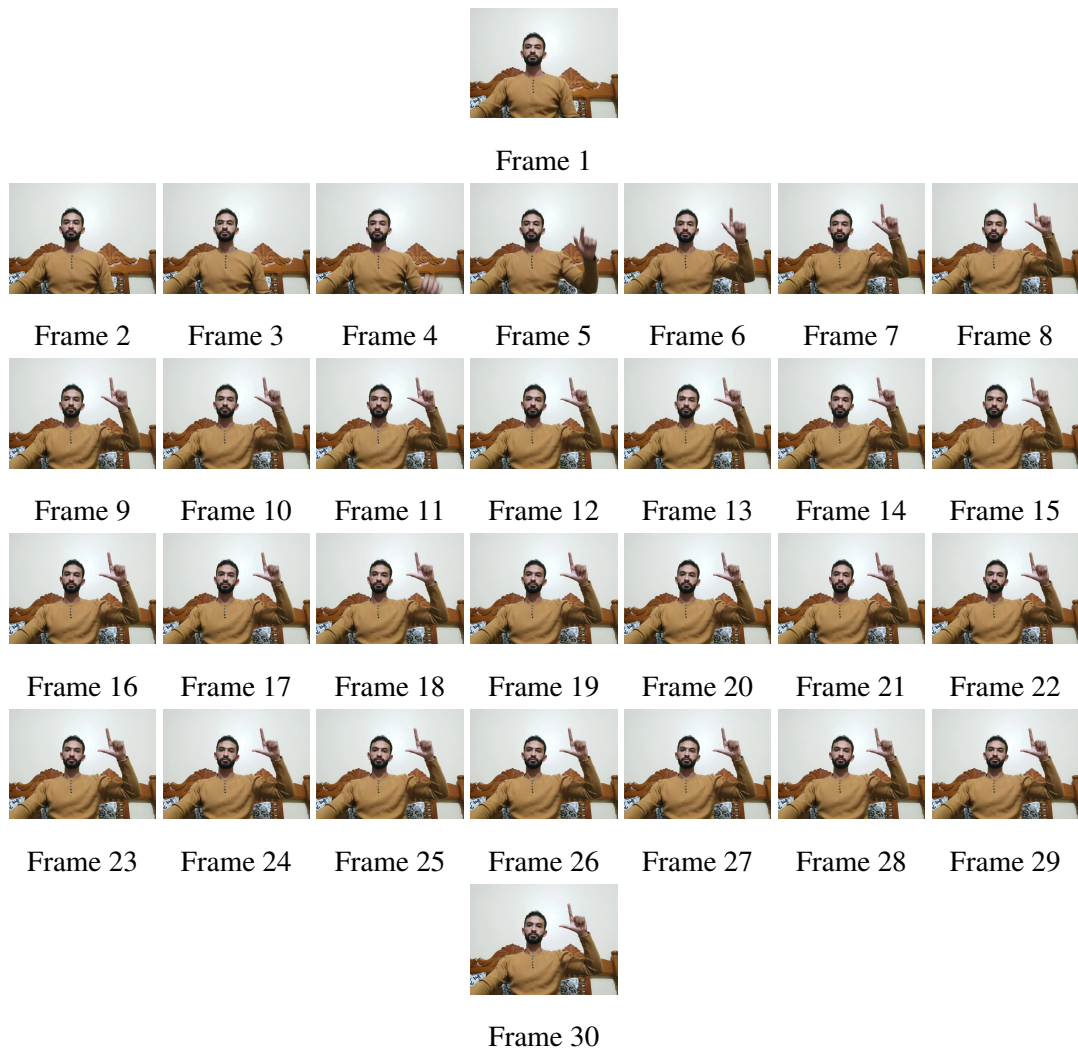


Figure 3.67: Sign Lam captured frames from the dataset.

Bibliography

- [Pyt] <https://www.python.org/doc/essays/blurb/> Accessed.03.2023.
- [Dif] The Difference Between Speech and Voice Recognition. <https://www.kardome.com/blog-posts/difference-speech-and-voice-recognition>. [Online; accessed 2023-02-03].
- [3] (2016). History of American Sign Language | DawnSignPress. <https://www.dawnsign.com/news-detail/history-of-american-sign-language>. [Online; accessed 2023-02-01].
- [4] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. *OSDI*, 16(265-283):1.
- [5] Abdel-Fattah, M. A. (2005). Arabic Sign Language: A Perspective. *The Journal of Deaf Studies and Deaf Education*, 10(2):212–221.
- [6] Abdulwahab, S., Jabreel, M., and Moreno, D. (2017). *Deep Learning Models for Paraphrases Identification*. PhD thesis.
- [7] Adaloglou, N., Chatzis, T., Papastratis, I., Stergioulas, A., Papadopoulos, G. T., Zacharopoulou, V., Xydopoulos, G. J., Atzakas, K., Papazachariou, D., and Daras, P. (2021). A comprehensive study on deep learning-based methods for sign language recognition. *IEEE Transactions on Multimedia*, 24:1750–1762.
- [8] Adithya, V. and Rajesh, R. (2020). A deep convolutional neural network approach for static hand gesture recognition. *Procedia computer science*, 171:2353–2361.
- [9] Aggarwal, C. C. et al. (2018). Neural networks and deep learning. *Springer*, 10(978):3.
- [10] Alam, M. (2016). Codes in matlab for training artificial neural network using particle swarm optimization.
- [11] Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.

- [12] Ali, M. M., Mahale, V. H., Yannawar, P., and Gaikwad, A. (2016). Overview of fingerprint recognition system. In *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pages 1334–1338. IEEE.
- [13] Altwaijry, N. and Al-Turaiki, I. (2021). Arabic handwriting recognition system using convolutional neural network. *Neural Computing and Applications*, 33(7):2249–2261.
- [14] Aparna, C. and Geetha, M. (2020). Cnn and stacked lstm model for indian sign language recognition. In *Machine Learning and Metaheuristics Algorithms, and Applications: First Symposium, SoMMA 2019, Trivandrum, India, December 18–21, 2019, Revised Selected Papers 1*, pages 126–134. Springer.
- [15] Aran, O. and Akarun, L. (2010). A multi-class classification strategy for fisher scores: Application to signer independent sign language recognition. *Pattern Recognition*, 43(5):1776–1788.
- [16] Athitsos, V., Wang, H., and Stefan, A. (2010). A database-based framework for gesture recognition. *Personal and Ubiquitous Computing*, 14:511–526.
- [17] Bheda, V. and Radpour, D. (2017). Using deep convolutional networks for gesture recognition in american sign language. *arXiv preprint arXiv:1710.06836*.
- [18] Binh, N. D., Shuichi, E., and Ejima, T. (2005). Real-time hand tracking and gesture recognition system. *Proc. GVIP*, pages 19–21.
- [19] Bono, M., Kikuchi, K., Cibulka, P., and Osugi, Y. (2014). A colloquial corpus of japanese sign language: Linguistic resources for observing sign language conversations. In *LREC*, pages 1898–1904.
- [20] Bradski, G. (2000). The OpenCV library. <https://opencv.org/>. Accessed: March 19, 2023.
- [21] Bui, T. D. and Nguyen, L. T. (2007). Recognizing postures in vietnamese sign language with mems accelerometers. *IEEE sensors journal*, 7(5):707–712.
- [22] Chang, S. and Sung, D. (2019). Modal-energy-based neuro-controller for seismic response reduction of a nonlinear building structure. *Applied Sciences*, 9:4443.
- [23] Chollet, F. (2015). Keras: Deep learning library for theano and tensorflow. *GitHub repository*, 2.
- [24] Chollet, F. and Team, K. (2021). Keras documentation: Optimizers. <https://keras.io/api/optimizers>. Accessed on May 7th, 2023.
- [25] Corporation, M. (accessed 2023). Visual studio code. <https://code.visualstudio.com/>.
- [26] Daugman, J. (2007). New methods in iris recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(5):1167–1175.
- [27] Davydov, M., Nikolski, I., and Pasichnyk, V. (2010). Real-time ukrainian sign language recognition system. In *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 1, pages 875–879. IEEE.

- [28] Debbagh, N. and Baghdadi, R. (2020). *Graduation Thesis - Deep Learning Operators Optimization in Tiramisu (Sparse Neural Networks and Recurrent Neural Networks)*. PhD thesis.
- [29] Dreuw, P. and Ney, H. (2008). Visual modeling and feature adaptation in sign language recognition. In *ITG Conference on Voice Communication [8. ITG-Fachtagung]*, pages 1–4. VDE.
- [30] El Zaar, A., Benaya, N., and El Allati, A. (2022). Sign language recognition: High performance deep learning approach applied to multiple sign languages. In *E3S Web of Conferences*, volume 351, page 01065. EDP Sciences.
- [31] Faundez-Zanuy, M. (2007). On-line signature recognition based on vq-dtw. *Pattern Recognition*, 40(3):981–992.
- [32] Flasiński, M. and Myśliński, S. (2010). On the use of graph parsing for recognition of isolated hand postures of polish sign language. *Pattern Recognition*, 43(6):2249–2264.
- [33] Gadekallu, T. R., Srivastava, G., Liyanage, M., Iyapparaja, M., Chowdhary, C. L., Koppu, S., and Maddikunta, P. K. R. (2022). Hand gesture recognition based on a harris hawks optimized convolution neural network. *Computers and Electrical Engineering*, 100:107836.
- [34] Gajalakshmi, P. and Sharmila, T. S. (2019). Hand gesture recognition by histogram based kernel using density measure. In *2019 2nd International Conference on Power and Embedded Drive Control (ICPEDC)*, pages 294–298. IEEE.
- [35] Gao, Q., Liu, J., Ju, Z., Li, Y., Zhang, T., and Zhang, L. (2017). Static hand gesture recognition with parallel cnns for space human-robot interaction. In *Intelligent Robotics and Applications: 10th International Conference, ICIRA 2017, Wuhan, China, August 16–18, 2017, Proceedings, Part I 10*, pages 462–473. Springer.
- [36] Garcia, B. (2006). The methodological, linguistic and semiological bases for the elaboration of a written form of French Sign Language (LSF). In Vettori, C., editor, *Proceedings of the LREC2006 2nd Workshop on the Representation and Processing of Sign Languages: Lexicographic Matters and Didactic Scenarios*, pages 31–36, Genoa, Italy. European Language Resources Association (ELRA).
- [37] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Chapter 8: Optimization for Training Deep Models.
- [38] Google (2021a). Mediapipe: Cross-platform framework for building multimodal applied machine learning pipelines. <https://mediapipe.dev/>. Accessed: March 19, 2023.
- [39] Google (2021b). Mediapipe face mesh: Face landmark detection. https://google.github.io/mediapipe/solutions/face_mesh.html. Accessed: March 19, 2023.
- [40] Google (2021c). Mediapipe hands: Hand tracking. <https://google.github.io/mediapipe/solutions/hands.html>. Accessed: March 19, 2023.

- [41] Google (2021d). Mediapipe holistic: Simultaneous face, hand, and pose estimation. <https://google.github.io/mediapipe/solutions/holistic.html>. Accessed: March 19, 2023.
- [42] Google (2021e). Mediapipe pose: Real-time human pose estimation. <https://google.github.io/mediapipe/solutions/pose.html>. Accessed: March 19, 2023.
- [43] Grauman, K. and Leibe, B. (2011). *Visual object recognition*. Number 11. Morgan & Claypool Publishers.
- [44] Grobel, K. and Assan, M. (1997). Isolated sign language recognition using hidden markov models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 1, pages 162–167. IEEE.
- [45] Gunes, H. and Piccardi, M. (2005). Fusing face and body gesture for machine recognition of emotions. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pages 306–311.
- [46] Gupta, S., Thakur, K., and Kumar, M. (2021). 2d-human face recognition using sift and surf descriptors of face’s feature regions. *The Visual Computer*, 37:447–456.
- [47] He, S. (2019). Research of a sign language translation system based on deep learning. In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pages 392–396.
- [48] Hindi, A., Dwairi, M. O., and Alqadi, Z. (2020). Analysis of procedures used to build an optimal fingerprint recognition system. *International Journal of Computer Science and Mobile Computing*, 9(2):21–37.
- [49] Jiang, S., Sun, B., Wang, L., Bai, Y., Li, K., and Fu, Y. (2021). Skeleton aware multi-modal sign language recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3413–3423.
- [50] Jinye, P. et al. (2008). Application of improved sign language recognition and synthesis technology in ib. In *2008 3rd IEEE Conference on Industrial Electronics and Applications*, pages 1629–1634. IEEE.
- [51] Johnston, T. and Schembri, A. (2007). *Australian Sign Language (Auslan): An introduction to sign language linguistics*. Cambridge University Press.
- [52] Karami, A., Zanj, B., and Sarkaleh, A. K. (2011). Persian sign language (psl) recognition using wavelet transform and neural networks. *Expert Systems with Applications*, 38(3):2661–2667.
- [53] Kausar, S. and Javed, M. Y. (2011). A survey on sign language recognition. In *2011 Frontiers of Information Technology*, pages 95–98.

- [54] Kong, W. and Ranganath, S. (2008). Signing exact english (see): Modeling and recognition. *Pattern Recognition*, 41(5):1638–1652.
- [55] Kortli, Y., Jridi, M., Al Falou, A., and Atri, M. (2020). Face recognition systems: A survey. *Sensors*, 20(2).
- [56] Kshirsagar, K. P. and Doye, D. (2010). Object based key frame selection for hand gesture recognition. In *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 181–185. IEEE.
- [57] Kurzynski, M., Puchala, E., Wozniak, M., and Zolnierrek, A. (2007). *Computer Recognition Systems: Proceedings of 4th International Conference on Computer Recognition Systems CORES'05*. Advances in Intelligent and Soft Computing. Springer Berlin Heidelberg.
- [58] Li, J., Li, C., Han, J., Shi, Y., Bian, G., and Zhou, S. (2022). Robust hand gesture recognition using hog-9ulbp features and svm model. *Electronics*, 11(7):988.
- [59] Li, Y., Yang, Q., and Peng, J. (2009). Chinese sign language recognition based on gray-level co-occurrence matrix and other multi-features fusion. In *2009 4th IEEE Conference on Industrial Electronics and Applications*, pages 1569–1572. IEEE.
- [60] Liao, Y., Xiong, P., Min, W., Min, W., and Lu, J. (2019). Dynamic sign language recognition based on video sequence with blstm-3d residual networks. *IEEE Access*, 7:38044–38054.
- [61] Lin, C., Chang, Q., and Li, X. (2019). A deep learning approach for mimo-noma downlink signal detection. *Sensors*, 19:2526.
- [62] Maebatake, M., Suzuki, I., Nishida, M., Horiuchi, Y., and Kuroiwa, S. (2008). Sign language recognition based on position and movement using multi-stream hmm. In *2008 Second International Symposium on Universal Communication*, pages 478–481. IEEE.
- [MANUAL] MANUAL, U. 3space® fastrak® user’s manual.
- [64] Maraqa, M. and Abu-Zaiter, R. (2008). Recognition of arabic sign language (arsl) using recurrent neural networks. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pages 478–481. IEEE.
- [65] Massone, M. I. and Curiel, M. (2004). Sign order in argentine sign language. *Sign Language Studies*, 5(1):63–93.
- [66] Mekala, P., Gao, Y., Fan, J., and Davari, A. (2011). Real-time sign language recognition based on neural network architecture. In *2011 IEEE 43rd Southeastern symposium on system theory*, pages 195–199. IEEE.

- [67] Mithe, R., Indalkar, S., and Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1):72–75.
- [68] Mittal, A., Kumar, P., Roy, P. P., Balasubramanian, R., and Chaudhuri, B. B. (2019). A modified lstm model for continuous sign language recognition using leap motion. *IEEE Sensors Journal*, 19(16):7056–7063.
- [69] Munib, Q., Habeeb, M., Takruri, B., and Al-Malik, H. A. (2007). American sign language (asl) recognition based on hough transform and neural networks. *Expert systems with Applications*, 32(1):24–37.
- [70] Nadgeri, S. M., Sawarkar, S., and Gawande, A. D. (2010). Hand gesture recognition using camshift algorithm. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 37–41. IEEE.
- [71] Oliphant, T. E. (2006). A guide to numpy. *USA: Trelgol Publishing*, 1(1):1–71.
- [72] Ozcan, T. and Basturk, A. (2019). Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition. *Neural Computing and Applications*, 31:8955–8970.
- [73] Pahlevanzadeh, M., Vafadoost, M., and Shahnazi, R. (2007). Sign language recognition. In *Signal Processing and Its Applications. ISSPA 2007. 9th International Symposium on*, pages 1–4. IEEE.
- [74] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.
- [75] Perry, G., Seidl, R., Bellve, A., and Rammer, W. (2022). An outlook for deep learning in ecosystem science. *Ecosystems*, 25:1–19.
- [76] Prasad, V. (2015). Voice recognition system: speech-to-text. *Journal of Applied and Fundamental Sciences*, 1(2):191.
- [77] Quan, Y. and Jinye, P. (2008). Chinese sign language recognition for a vision-based multi-features classifier. In *2008 International Symposium on Computer Science and Computational Technology*, volume 2, pages 194–197. IEEE.
- [78] Rana, S., Liu, W., Lazarescu, M., and Venkatesh, S. (2009). A unified tensor framework for face recognition. *Pattern Recognition*, 42(11):2850–2862.
- [79] Rastgoo, R., Kiani, K., and Escalera, S. (2021). Sign language recognition: A deep survey. *Expert Systems with Applications*, 164:113794.

- [80] Rathmann, C. (2003). The optionality of agreement phrase: Evidence from german sign language (dgs). In *Texas Linguistics Forum*, volume 53, pages 181–192. Citeseer.
- [81] Rezaei, A., Vafadoost, M., Rezaei, S., and Daliri, A. (2008a). 3d pose estimation via elliptical fourier descriptors for deformable hand representations. In *2008 2nd International Conference on Bioinformatics and Biomedical Engineering*, pages 1871–1875. IEEE.
- [82] Rezaei, A., Vafadoost, M., Rezaei, S., and Shekofteh, Y. (2008b). A feature based method for tracking the 3-d trajectory and the orientation of a signer's hand. In *2008 International Conference on Computer and Communication Engineering*, pages 346–351. IEEE.
- [83] Riesenhuber, M. and Poggio, T. (2000). Models of object recognition. *Nature neuroscience*, 3(11):1199–1204.
- [84] Rokade, U. S., Doye, D., and Kokare, M. (2009). Hand gesture recognition using object based key frame selection. In *2009 International Conference on Digital Image Processing*, pages 288–291. IEEE.
- [85] S, K. and Chandra, E. (2016). A review on automatic speech recognition architecture and approaches. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 9:393–404.
- [86] Sadeddine, K., Djeradi, R., Chelali, F. Z., and Djeradi, A. (2018). Recognition of static hand gesture. In *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*, pages 1–6. IEEE.
- [87] Sahoo, A. K., Mishra, G. S., and Ravulakollu, K. K. (2014). Sign language recognition: State of the art. *ARPJ Journal of Engineering and Applied Sciences*, 9(2):116–134.
- [88] Sahoo, J. P., Prakash, A. J., Plawiak, P., and Samantray, S. (2022). Real-time hand gesture recognition using fine-tuned convolutional neural network. *Sensors*, 22(3):706.
- [89] Sarkaleh, A. K., Poorahangaryan, F., Zanj, B., and Karami, A. (2009). A neural network based system for persian sign language recognition. In *2009 IEEE International Conference on Signal and Image Processing Applications*, pages 145–149. IEEE.
- [90] Shanableh, T. and Assaleh, K. (2007). Arabic sign language recognition in user-independent mode. In *2007 International Conference on Intelligent and Advanced Systems*, pages 597–600. IEEE.
- [91] Sutton-Spence, R. and Woll, B. (1999). *The linguistics of British Sign Language: an introduction*. Cambridge University Press.
- [92] Tan, Y. S., Lim, K. M., and Lee, C. P. (2021). Hand gesture recognition via enhanced densely connected convolutional neural network. *Expert Systems with Applications*, 175:114797.
- [93] Tandel, N. H., Prajapati, H. B., and Dabhi, V. K. (2020). Voice recognition and voice comparison using machine learning techniques: A survey. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 459–465.

- [94] Thampi, S., Gelbukh, A., and Mukhopadhyay, J. (2014). *Advances in Signal Processing and Intelligent Recognition Systems*. Advances in Intelligent Systems and Computing. Springer International Publishing.
- [95] Thangali, A., Nash, J. P., Sclaroff, S., and Neidle, C. (2011). Exploiting phonological constraints for handshape inference in asl video. In *CVPR 2011*, pages 521–528. IEEE.
- [96] Tsechpenakis, G., Metaxas, D., and Neidle, C. (2006). Learning-based dynamic coupling of discrete and continuous trackers. *Computer Vision and Image Understanding*, 104(2-3):140–156.
- [97] Valli, C. and Lucas, C. (2000). *Linguistics of American sign language: An introduction*. Gallaudet University Press.
- [98] Vanjikumaran, S. and Balachandran, G. (2011). An automated vision based recognition system for sri lankan tamil sign language finger spelling. In *2011 International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 39–44. IEEE.
- [99] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [100] Vishwakarma, D. K. (2017). Hand gesture recognition using shape and texture evidences in complex background. In *2017 International Conference on Inventive Computing and Informatics (ICICI)*, pages 278–283. IEEE.
- [101] Wang, F., Hu, R., and Jin, Y. (2021). Research on gesture image recognition method based on transfer learning. *Procedia Computer Science*, 187:140–145.
- [102] Wang, W., He, M., Wang, X., Ma, J., and Song, H. (2022). Medical gesture recognition method based on improved lightweight network. *Applied Sciences*, 12(13):6414.
- [103] Xiaolong, T., Bian, W., Weiwei, Y., and Liu, C. (2005). A hand gesture recognition system based on local linear embedding. *Journal of Visual Languages Computing*, 16(5):442–454.
- [104] Zhang, F., Liu, Y., Zou, C., and Wang, Y. (2018). Hand gesture recognition based on hog-lbp feature. In *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6. IEEE.
- [105] Zhou, Y., Yang, X., Lin, W., Xu, Y., and Xu, L. (2011). Hypothesis comparison guided cross validation for unsupervised signer adaptation. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–4. IEEE.
- [106] Zhou, Z., Du, Y., and Belcher, C. (2009). Transforming traditional iris recognition systems to work in nonideal situations. *IEEE Transactions on Industrial Electronics*, 56(8):3203–3213.