Democratic and Popular Algerian Republic

Ministry of Higher Education and Scientific Research

Yahia fares university of Médéa

Faculty of Sciences

Department of Mathematics and Computer Science

Final Master Project

Branch: computer science

Specialty: Systems Engineering and Web Technologies

Theme

Deep-rsl: Deep learning for sign language recognition from a video sequence

Proposed and directed by

— Dr.KHELDOUN AHMED

Prepared and presented by

— KOUAR IMENE

— KOUAR EL BACHIR

Year: 2022–2023



Acknowledgments

We are deeply grateful and infinitely thankful to Allah for giving us the power and patience during this work accomplishment. We would like to express our deepest gratitude and appreciation to all those who have contributed to the completion of this work. Their support, guidance, and assistance have been invaluable throughout our research journey.

First and foremost, we would like to extend our heartfelt thanks to our advisor,Dr Kheldoun Ahmed. His hard work, continuous support, and provision of necessary knowledge have been instrumental in shaping the outcome of this work. We are truly grateful for his dedication and commitment.

We are also grateful to the deaf schools in Ben Slimane, Blida, and Rouiba for their invaluable contribution. The opportunity to interact with deaf students and gain a profound insight into the deaf world has enriched our research significantly. Their openness, resilience, and willingness to share their stories have broadened our understanding and shaped the direction of our study.

Our sincere appreciation goes to Ms. Majdoub Naima and Ms. Bachira for their invaluable teaching of academic sign language. Their expertise and guidance have bridged the communication gap, enabling us to establish a deeper connection with the deaf community.

We would like to express our gratitude to Ms. Boudkhani Nessrine for sharing her knowledge, tips, and advice on building our model. Her guidance has enhanced the quality and effectiveness of our research.we extend a special thanks to Mr. Soumil for his invaluable assistance throughout our journey, providing us with advice and tips for developing our application.

Furthermore, we extend our appreciation to the following individuals: Amir Fedjri, Antri Arselane Chakib, Tair Nour el Houda, Hentabli Wissem, Khelil Chorfi Abdessalam, Hasnaoui Houssem, Hentabli Rym, Aya , Amine, Kouar Mohamed, and Kouar Nadia. Their contributions, hard work, and dedication have played a crucial role in the successful completion of this work.

We would like to acknowledge the unwavering emotional support and dedication of our father. His belief in our abilities and his continuous support in taking us to various places have been invaluable throughout this academic journey.

Additionally, we express our heartfelt thanks to Hefri Souad for her unwavering emotional support and care, not only for us but also for our entire family.

Finally, we would like to extend our gratitude to all the individuals who have been there for us, offering their support, encouragement, and understanding. Your presence and belief in our abilities have been a constant source of motivation and inspiration.

Once again, we express our deepest appreciation to everyone involved in this work. Your contributions have been instrumental in our academic and personal growth. Thank you for being there for us.



Dedication

In loving memory of our dear deaden mother,

This document is dedicated to you as a humble tribute and perpetual remembrance of your unconditional love, unwavering support, and profound impact on our lives. Your presence may be absent, but your guidance will forever inspire us. May this work serve as a صدقة for your beautiful soul, a testament to the values and strength you instilled in us.

To all the deaf students,

This work is dedicated to you, with the utmost hope for a brighter future and inclusive society. May it serve as a catalyst for change, fostering greater understanding, acceptance, and opportunities for you to engage fully in all aspects of life. We dedicate this document to your journey, with the aspiration that barriers will crumble, and your voices will resonate with the world. May you interact, learn, and flourish just as any other member of the community, embracing your unique identities and contributing your invaluable perspectives to the tapestry of humanity.

With heartfelt dedication,

Kouar Imene and Kour EL Bachir

Table of Content

Table of content.....I
List of Figures.....II
List of Tables.....III
List of Abbreviations.....IV

General Introduction 1
Chapter I: Sign Language.....4
I. Introduction.....5
I.1 Background.....5
I.2 History 6
I.3 Algerian Sign Language 7
I.3.1 Definition.....7
I.3.2 Background.....7
I.3.3 Types of Signs 8
I.3.3.1 Gestures that involve the entire body 8
I.3.3.2 Head and Face Gestures.....8
I.3.3.3 Hand and Arm Gestures.....8
I.3.4 How sign language is structured.....9
I.3.4.1 Handshape 9
I.3.4.2 Movement.....9
I.3.4.3 Location 10

Table of Content

I.3.4.4 orientation 11
I.3.4.5 Facial Expressions 11
I.3.5 Algerian Sign Language Alphabet (ALSA) 12
I.4 Sign Language Recognition 13
I.4.1 Isolated Sign Language Recognition 13
I.4.2 Continuous Sign Language Recognition 14
I.5 Conclusion 14
Chapter II: Literature review 15
II.1 Introduction 16
II.2 Artificial intelligence 16
II.2.1 machine learning 16
II.2.2 Deep learning 18
II.3 Artificial neural network 18
II.3.1 The characteristics of an artificial neuron 18
II.3.2 The different layers of a neural network 19
II.3.3 Activation function 20
II.3.4 Loss function 21
II.3.5 Batch size 21
II.3.6 Backpropagation 21
II.3.7 The learning rate 22
II.4 Convolutional Neural Networks (CNNs) 22
II.5 Recurrent Neural Networks 24
II.5.1 Long Short-Term Memory Networks 25
II.6 The Self-Attention Mechanism 25
II.7 The Transformer 26

Table of Content

II.7.1 The Transformer Encoder 27

II.7.2 The Transformer Decoder 28

II.8 Autoencoder 28

II.9 Evaluation metrics of models 30

II.10 Optimizers 32

II.11 Related Works 33

II.11.1 SLR with CNNs 33

II.11.2 SLR with LSTM Models 33

II.11.3 SLR with the Transformer 34

II.12 Conclusion 34

Chapter III: Conception 35

III.1 Introduction 36

III.2 Overview 36

III.3 Algerian Sign Language Dataset (SIGAL) 37

III.3.1 Dataset 37

III.3.2 Recording 39

III.4 Pre-processing 40

III.4.1The Key Frame Selection Mechanism 40

III.5 The Data Augmentation. 53

III.6 Model Development 54

III.7 Use case diagram 57

III.7.1 Identification of actors ..... **Error! Bookmark not defined.**

III.8 Conclusion 58

Chapter IV: Implementation and performance testing 59

Table of Content

IV.1 Introduction.....60

IV.2 Development environment.....60

IV.2.1 Hardware specifications.....60

IV.2.2 Software specifications and libraries.....60

IV.2.2.1 Software.....60

IV.2.2.2 libraries.....61

IV.3 Implementation details and results.....62

IV.3.1 preprocess.....62

IV.3.1.1 Choice of preprocessing variant.....62

IV.3.1.2 Comparison with other variants.....63

IV.3.1.3 Parameter tuning and adjustments of variant.....64

IV.3.2 The model.....68

IV.3.2.1 The choice of parameters.....68

IV.4. Results.....69

IV.4.1 classifications report.....71

IV.4.2 Confusion Matrix.....73

IV.4.3 Test Models on SIGAL Dataset.....76

IV.4.4 Comparative analysis.....76

IV.4.4.1 different datasets on our chosen model.....76

IV.4.4.2 LSA64 comparison.....77

IV.4.4.3 ASl_vd20 comparison.....78

IV.6 Discussion.....79

IV.7 The graphical interface.....80

IV.8 Conclusion.....81

Chapter V: Business Model Canvas.....83

Table of Content

V.1 Introduction 84

V.2 What is a business model? 84

V.2.1 Definition 84

V.2.1 Originality of the BMC 84

V.2.2 Why use the Business Model Canvas? 84

V.3 The 9 building blocks of the Business Model Canvas 85

V.3.1 Desirability 86

V.3.2 Viability 86

V.3.2 Feasibility 87

V.4 SignAtlas 88

V.5 Conclusion 90

General Conclusion 91

List of Figures

Figure I.2: Sign of the word « موعد » 10

Figure I.3: Sign of the word «دواء» 10

Figure I.4: Sign of the letter «V» and the word «two» 11

Figure I.5: Sign of the word « سعيد » 11

Figure I.6: Algerian Sign Language Alphabet (the ones with red border are the dynamic signs) [9].....12

Figure II.1: Example of a simple perceptron. 19

Figure II.2: general architecture of a neural network.[29].....20

Figure II.3: The structure of a classic CNN model.....22

Figure II.4: The convolution process 23

Figure II.5: The process of different pooling layer.....23

Figure II.6: The structure of a LSTM unit 24

Figure II.7: The structure of a transformer. 27

Figure II.7: The structure of a transformer. 28

Figure II.9: Deep Autoencoder architecture. 30

Figure II.10: confusion matrix Table structure 32

Figure III.1: General diagram of the proposed approach 37

Figure III.2: snapshots of six key-frames random distinct signs extracted from the SIGAL database 38

Figure III.3: The total number of videos per person.....39

Figure III.4: the configuration of the recording setup 40

List of Figures

Figure III.5: sample of key-frames extraction using variant 01 41

Figure III.6: Chart flow of Variant 01 for Key-frame Extraction.....42

Figure III.7: sample of 8 key-frames extraction using variant 02 43

Figure III.8: Chart flow of Variant 02 for Key-frame Extraction.....45

Figure III.9: Chart flow of Variant 03 for Key-frame Extraction.....47

Figure III.10: sample of key-frames extraction using variant 04 48

Figure III.11: Chart flow of Variant 04 for Key-frame Extraction.....49

Figure III.12: sample of optical flow key-frames extraction using variant 05.....51

Figure III.13: sample of depth key-frames extraction using variant 05 51

Figure III.14: Chart flow of Variant 05 for Key-frame Extraction.....52

Figure III.15: Architecture of our auto encoder model.....55

Figure III.16: detailed summery of developed model 56

Table III.3 - Use Case Scenario for Sign Language Recognition System.....57

Figure III.17: Use Case Diagram Of our Sign Language Recognition System.....57

Figure IV.1 – Logos of software specifications and libraries used. 61

Figure IV.1 – Logos of software specifications and libraries used. 64

Figure IV.3: Comparison between two different crop scale on same frame with extraction of vari-ant 4.....65

Figure IV.4: Variant 4 with the Addition of the Background Parameter, showing a Successful Background Remover and anther Failed attempt and taken time for single frame extraction. 67

Figure IV.5: Training and Validation Accuracy on SIGAL dataset with our model and variant 4.....69

Figure IV.5: Training and Validation Loss on SIGAL dataset with our model and variant 4.....70

Figure IV.6: Training and Validation Loss on SIGAL dataset with our model and variant 4.....74

Figure IV.7: Zoomed-in section of the top-left corner of the confusion matrix.....75

Figure IV.8: graphical interface of the home page of the computer software.....80

Figure IV.9: graphical interface of the webcam interface 81

Figure V.1: BMC divided sections 815

List of Tables

Table II.1 - Summary of machine learning approaches.....17

Table III.1: provides a detailed list of the signs used in our SIGAL dataset, along with the ID and hand(s) used for each sign. 38

Table III.2 - Summary of variants of keyframe extraction applied on same video.53

Table IV.2: Comparison of Preprocessing Variants 4 with different number of frames.....65

Table IV.4: Model Parameters and their Corresponding Values with their descriptions 68

Table IV.5: Classifications Report.....71

Table IV.6: Performance Comparison of Different Models on SIGAL Dataset 76

Table IV.7: Performance Comparison of Different Datasets with our model only.77

Table IV.8: Performance Comparison of LSA64 dataset.....77

Table IV.9: Performance Comparison of ASl_vd20 79

Table V.1.: Business Model Canvas for SignAtlas 90

List of Abbreviations

AI	Artificial intelligence
ALSA	Alphabet Langue des signes algériens
ASL	American Sign Language
BSL	British Sign Language
CGSLR	Continuous Gesture-Based Sign Language Recognition
CNN	Convolutional Neural Network
CSLR	Continuous sign language recognition
CVSLR	Continuous Vision-Based Sign Language Recognition
DL	Deep learning
DQN	Deep Q-Network
EMG	Electromyography
FP	False Positive
FPS	Frame per second
FN	False Negative
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IMU	Inertial Measurement Unit
ISLR	Isolated sign language recognition
LSTM	Long Short-Term Memory
LSA	Langue des signes algériens
ML	Machine Learning
RGB	Red, Green and Blue
RNN	Recurrent neural network
SIGAL	Algerian sign language dataset
SLR	Sign language recognition

List of Abbreviations

SVM	Support Vector Machine
TP	True Positive
TN	True Negative
TNF	Total number of frames per video

General Introduction

Sign language plays a crucial role in facilitating communication and interaction for the deaf community. However, the recognition of sign language poses unique challenges, especially in the context of Algerian Sign Language (LSA), where limited research has been conducted. This gap in knowledge presents an opportunity to delve into uncharted territory and contribute to the development of effective LSA recognition systems.

The motivation behind our research stems from two critical aspects. Firstly, we recognize the pressing need to address the challenges faced by the deaf community, whose language and communication needs often receive inadequate attention. By focusing our efforts on LSA recognition, we aim to bridge this gap and provide a technological solution that enhances the communication and accessibility of deaf individuals.

Secondly, we identified the absence of an available dataset for LSA, which presented a significant hurdle in our research. To overcome this challenge, we took a proactive approach and visited several deaf schools across Algeria, immersing ourselves in the language and culture of the deaf community. This involved actively engaging with deaf individuals, sign language interpreters, and native signers to capture a wide range of LSA signs, gestures, and expressions. By actively involving the deaf people and gathering a diverse dataset, we ensured that our research reflects the linguistic and regional variations within LSA.

Our main objective is to develop a real-time application that utilizes machine learning (deep learning) techniques, specifically an autoencoder with attention, to interpret and recognize LSA gestures accurately. Real-time recognition, with response times of less than 2 seconds, was a significant challenge. Balancing the trade-off between accuracy and response time, we aimed to achieve reliable and efficient LSA recognition in real-world scenarios.

Additionally, we focused on ensuring the robustness and scalability of our recognition system. This included incorporating a wide range of words and gestures into our dataset, allowing for the recognition of any LSA signer, not just those who were part of the training process. By minimizing overfitting and optimizing the performance of our model, we aimed to create a system that is capable of accurately interpreting LSA gestures with high generalizability.

Throughout our research, we collaborated closely with the deaf community, valuing their expertise and incorporating their feedback to ensure the accuracy and cultural authenticity of the collected data. By

actively involving them in the research process, we strived to create a recognition system that aligns with their needs and preferences.

By addressing the challenges of LSA data collection, incorporating machine learning (deep learning) techniques, and engaging with the deaf community, our research endeavors seek to contribute to the advancement of LSA recognition systems. Our ultimate goal is to make a tangible impact on the lives of the deaf community in Algeria, empowering individuals who rely on sign language as their primary mode of communication and ensuring their inclusion and participation in various aspects of social, educational, and professional interactions.

In the following chapters, we will delve into the literature review, methodology, implementation details, and performance testing of our LSA recognition system. We will present our findings, discuss the results, and provide insights for future research and improvements. Through our research, we aim to contribute to the growing field of sign language recognition and facilitate enhanced communication and accessibility for the deaf community in Algeria and beyond.

In this work, we will try to answer these questions:

- How can we effectively recognize Algerian Sign Language (LSA) using machine learning techniques?
- What are the challenges in collecting a comprehensive LSA dataset, and how can we address them?
- How can we develop a real-time LSA recognition system that balances accuracy and response time?
- What impact can LSA recognition technology have on enhancing communication and accessibility for the deaf community in Algeria?

This report will contain five chapters

❖ **Chapter I: Sign Language:**

This chapter introduces the concept of sign language, focusing on Algerian Sign Language (LSA), its definition, and its structure.

❖ **Chapter II: Literature Review:**

This chapter provides an overview of relevant literature in the field of sign language recognition, including AI techniques, neural networks, and related works.

- **Chapter III: Conception:**

  This chapter outlines the development process, including the creation of an Algerian Sign Language Dataset (SIGAL), the implementation of pre-processing techniques and the explanation of the proposed machine learning (deep learning) model.

- **Chapter IV: Implementation and Performance Testing:**

  This chapter discusses the implementation details, hardware and software specifications, performance testing of the sign language recognition model and the graphical interface of our LSA recognition system.

- **Chapter V: Business Model Canvas:**

  This chapter presents a summary of the business model canvas and its implications for the project.

Chapter I

Sign Language

I. Introduction

Sign languages, unlike spoken languages, are natural forms of communication that rely on visual and non-verbal cues such as hand gestures and body language. Despite their differences, both sign and spoken languages share a common purpose, that is to facilitate communication. Sign languages are used primarily by people who are deaf, mute, or hard of hearing to communicate with others through gestures.

In this chapter, we provide an introduction to the world of sign languages, including an overview of the different universal sign languages and their unique characteristics. We also delve into the specific features of Algerian Sign Language, including its origin and it structure. Furthermore, we explore the concept of automatic sign recognition and its advantages.

I.1 Background

Sign languages are methods of communication that do not depend on spoken words but employ a diverse array of visual components, such as hand gestures, body motions, facial expressions, and finger-spelling, to transmit information. These features are organized and merged based on linguistic regulations to articulate various concepts and notions.

Due to their reliance on visual elements rather than acoustic information, sign languages have gained popularity within the deaf community, which is comprised of over 70 million individuals worldwide. These individuals tend to view their deafness as a unique life experience, rather than a disability, and take pride in their use of sign languages [1]. Beyond the deaf community, sign languages are also utilized by some hearing individuals who face challenges with verbal communication or have deaf family members. This demonstrates the versatility and inclusivity of sign languages as a mode of communication.

Despite their widespread usage, sign languages are frequently misconstrued and subject to numerous biases. One of the most pervasive misunderstandings is the assumption that sign languages are mere visual translations of spoken languages. Nevertheless, this assumption is unfounded, as sign languages are markedly distinct from spoken languages, both in terms of their grammar and vocabulary. To illustrate, the word "right" in English can denote the opposite of "left," as well as imply correctness or authority. Conversely, no single sign in British Sign Language (BSL) or American Sign Language (ASL) can convey all of these meanings simultaneously [2]. This example highlights the fact that each sign in sign languages corresponds to a specific meaning, rather than a word in a spoken language.

One more common misconception regarding sign languages is that they are universally understood and used globally, much like a language that can be spoken and comprehended by individuals worldwide.

Nevertheless, this is not an accurate belief. As with spoken languages, there is no one universal sign language that is universally used and comprehended across the world. Ethnologue's latest edition in 2022 [3] reports over 150 distinct sign languages used around the world. While these sign languages may share some similarities, they have evolved independently and are not mutually understandable. For instance, even though the United Kingdom and the United States share the same spoken language of English, a deaf individual from the UK may have difficulty communicating with a deaf individual from the US due to the differences between their respective sign languages.

I.2 History

The origins of systematic sign language can be traced back to the 18th century, during which Charles-Michel de l'Épée, often referred to as the "father of the deaf," created a series of synthesized signs based on French grammar for the first deaf school in Paris. This institution is considered the birthplace of modern French Sign Language [4]. Following this development, there was much debate over the use of sign languages in deaf education.

In the early days of educating deaf students, educators insisted on teaching them to speak and lip-read, while avoiding the use of sign language during lectures. The main goal of this approach was to facilitate the integration of deaf students into the hearing community. In 1880, the Second International Congress on the Education of the Deaf (ICED) held in Milan passed several resolutions that prohibited the use of sign language in classrooms. Nevertheless, deaf students' hearing impairment makes it challenging for them to understand word pronunciations and comprehend others' meanings through lip-reading, which can seriously affect their confidence in learning. On the other hand, sign languages have been demonstrated to be more effective in boosting the confidence and communication skills of deaf students [6]. As a result, many arguments and protests have arisen against these resolutions since their adoption [5]. Eventually, the ICED committee acknowledged the negative impact of these resolutions on millions of deaf individuals and issued a formal apology in 2010 [7]

At present, most people agree that sign language is the best way for the deaf to communicate. Deaf individuals commonly use sign language in their ordinary lives.

As explained in Section I.1., sign languages exhibit significant differences from spoken languages regarding their lexicons and linguistic grammars. Consequently, hearing individuals face substantial difficulties communicating through sign languages without proper training, creating a communication gap between the deaf and hearing communities. To address this issue, recent attention has focused on the potential solutions offered by technologies and applications of sign language recognition. However, sign language

recognition represents a challenging research field, incorporating diverse techniques such as gesture recognition, language modeling, and computer vision. Therefore, this document aims to develop DL frameworks that effectively recognize sign language, enabling the breakdown of communication barriers between the hearing and the deaf.

I.3 Algerian Sign Language

I.3.1 Definition

The Algerian Sign Language (LSA) is a gestural language used by deaf individuals and their families in Algeria as a means of communication through signs. The Algerian Sign Language is officially recognized by the law of May 8, 2002, as the first language of the deaf community in Algeria, making it the only country in the Arab world and in Africa to officially recognize sign language [8].

This language (LSA) is a gestural language that relies entirely on signs, which are produced through different parts of the body, including one or both hands, the face, shoulders, or even the entire body. Despite its unique mode of expression, the LSA is considered to be a language like any other, possessing a distinct vocabulary and organized syntax. To effectively learn and understand the LSA, one must simply become familiar with its alphabet. In fact, every sign within the LSA alphabet is generated by one or two particular hand postures, highlighting the precision and complexity of the language's gestural system. [9]

I.3.2 Background

Algerian Sign Language can be traced back to its origin from French Sign Language (Langue des Signes Française, LSF). It is noteworthy that Algerian Sign Language is not uniform, and comprises various regional variations [34]. These variations are not exhaustive and include, but are not limited to:

- Algerian Jewish Sign Language, AJSL or Ghardaia Sign Language since it was mainly developed and used in the village of Ghardaia by the Algerian Jewish individuals at that time [35][36] [37].
- Algerian Sign Language of Laghouat, which is used by many Deaf people in Laghouat province and other cities (or villages) around it [38].
- Algerian Sign Language of Oran, it is used by the Deaf in the North of Algeria, particularly in the city of Oran [39]
- Algerian Sign Language of Adrar, which is used by the Algerian Deaf community in Adrar, in the South of Algeria [24].

Therefore, it appears that various Algerian Deaf communities residing in distinct provinces or villages throughout Algeria may develop their own unique dialects of Algerian Sign Language. It is probable that multiple Algerian Sign Languages exist due to the existence of numerous Deaf communities in Algeria, particularly in larger urban areas, despite similarities or differences between them. These Algerian Sign Languages may have originated primarily in certain villages and are used by Algerian Deaf individuals, as well as some of their acquaintances, relatives, friends, or those who interact with them in various capacities.

I.3.3 Types of Signs

When discussing the use of sign language, it is important to understand the concept of signs. A sign is a category of gestures, which may or may not involve a certain duration of time [9]. These gestures can be classified according to the parts of the body involved. Typically, three types of gestures are distinguished [10]:

I.3.3.1 Gestures that involve the entire body

The field of research on body language explores the interaction of the entire body with its environment, such as analyzing the gestures of an athlete to improve their performance. Within this domain, gestures can be categorized into two main types: dynamic and static gestures. A static gesture, also known as a posture, pertains to the body or a part of the body's configuration at a fixed moment in time. In contrast, a dynamic gesture refers to a continuous succession of postures. This classification of gestures based on their temporal and dynamic features enables a deeper understanding of body language and its role in communication.

I.3.3.2 Head and Face Gestures

There are few specific meanings attached to head gestures; however, the orientation of the head is crucial for detecting the field of vision.

I.3.3.3 Hand and Arm Gestures

This category of gestures represents the main category of interactive gestures. The hand allows for precise and complex gestures Since sign language does not rely on vocalizations, hand and arm gestures are the only means of conveying information. As a result, it is crucial to accurately recognize and interpret these gestures when learning or using sign language. Additionally, the subtle differences in hand and arm movements can affect the meaning of a sign, underscoring the importance of proper technique and practice

when communicating in sign language. Research in this field mainly focuses on hand position recognition, sign language interpretation, and enabling the manipulation and interaction with data or elements in an environment.

I.3.4 How sign language is structured

The movements of a hand can be analyzed based on five separate parameters, which are independent and can either be dynamic or consistent throughout the signing process [11]. These parameters have been defined as follows:

I.3.4.1 Handshape

A configuration in sign language refers to the hand shape at a specific moment, defined by the positioning of the fingers and palm. However, due to natural variation, hand shapes can differ from person to person and are never entirely stable. As such, configurations of the same sign can vary slightly between signers. The figure I.1 illustrates an example of a configuration in Algerian Sign Language, depicting as a hand with three fingers spread out and the other two fingers, the thumb and index finger touching.



Figure I.1: Sign of the word « وجد »

I.3.4.2 Movement

In sign language, movement refers to the trajectory followed by one or both hands during signing, which can take various forms, such as lines, arcs, and circles. Additionally, head movements may also be incorporated in signing. Movement in sign language involves three parameters: trajectory, direction (upward,

downward, etc.), and speed (fast, medium, slow, etc.). In some cases, these movements may be repetitive or periodic.

As an example, in Figure I.2, the sign "موعد" in Algerian Sign Language can be described by two parameters: configuration and movement. The movement parameter is represented by a hand moving in an arc trajectory.



Figure I.2: Sign of the word « موعد »

I.3.4.3 Location

Location refers to the position of the hand in space relative to the signer's body or a specific object, which can completely change the meaning of a gesture. For example, the hand can be placed in front of the signer, near the signer's head, on the mouth, eyes, arm, palm, etc.

For instance, the Algerian word "دواء" is represented in Algerian Sign Language by the sign shown in Figure I.3, where the hand is placed in front of the face, near the mouth.



Figure I.3: Sign of the word «دواء»

I.3.4.4 orientation

The various directions towards which the hands and fingers are pointed can be grouped into distinct orientations [12], as illustrated in Figure I.4. The orientation parameter describes the orientation of the hand with respect to the signer. Consequently, the direction of the hand is determined by the position of the palm relative to the signer's body [9]

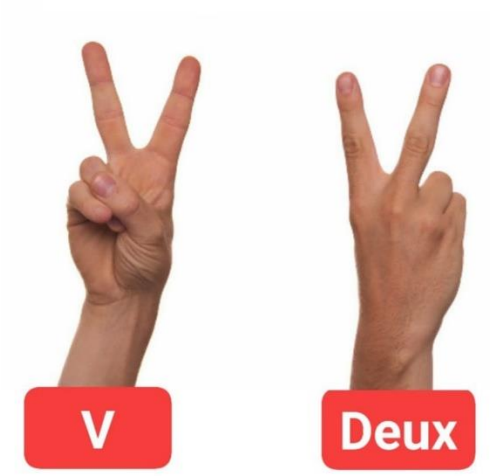


Figure I.4: Sign of the letter «V» and the word «two»

I.3.4.5 Facial Expressions

In reality, certain sign language expressions comprise hand gestures in conjunction with facial expressions, or facial expression mimics. The latter is crucial in ascribing meaning to a single sign, and is fundamental to sentence construction. For instance, in Algerian Sign Language, a facial expression mimic is employed in the term "سعيد" as illustrated by Figure I.5.



Figure I.5: Sign of the word « سعيد »

I.3.5 Algerian Sign Language Alphabet (ALSA)

Algerian Sign Language is composed of a total of 42 signs for alphabet, which include 35 static and 6 dynamic signs. It is noteworthy that these signs are produced using a single hand gesture (refer to Figure I.6). Moreover, each static sign is identified by two distinct features: configuration and orientation

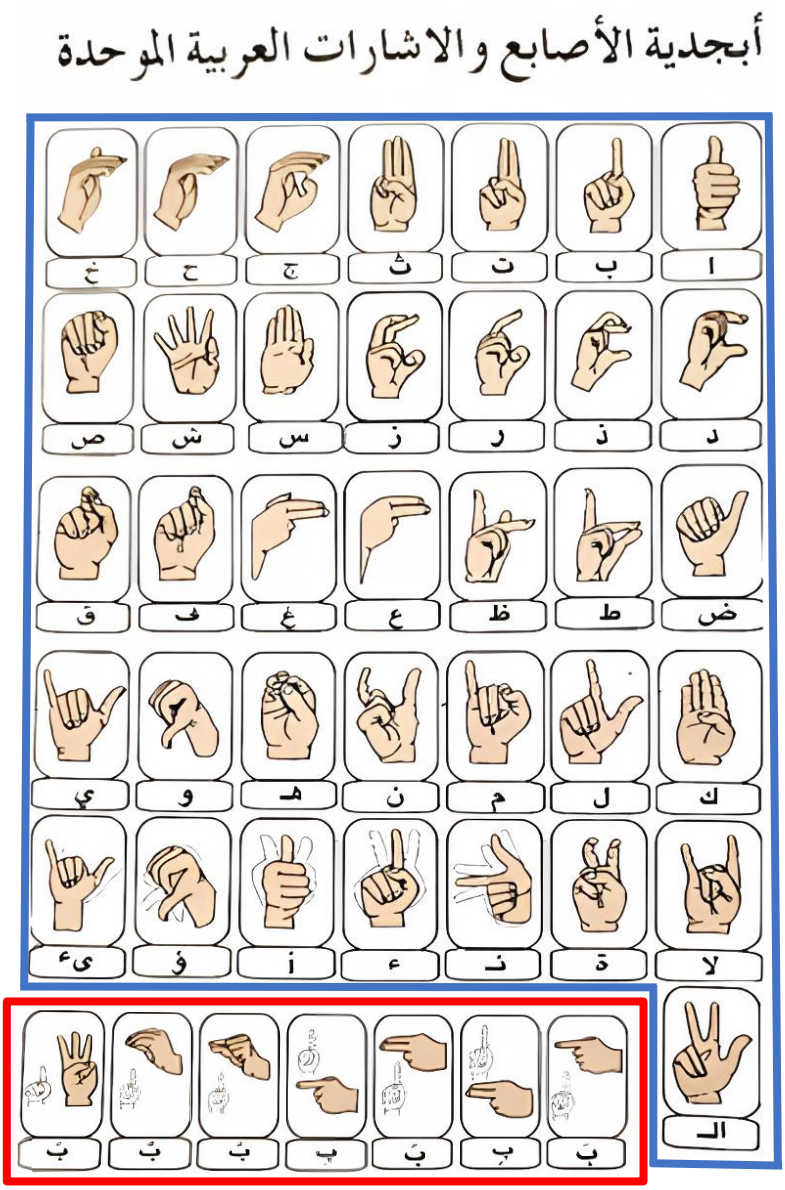


Figure I.6: Algerian Sign Language Alphabet (the ones with red border are the dynamic signs) [9].

As an example, the sign for the letter "ب" in Algerian Sign Language is defined by two parameters [9]:

- Configuration: making a fist with the index finger extended.
- Orientation: the palm is facing upwards (or with the wrist in a downwards position).

I.4 Sign Language Recognition

Sign language recognition (SLR) is a developing area of research that aims to interpret the meaning of sign languages as conveyed by signers through their movements and gestures.

However, the SLR is a huge novel area of research that involves interpreting sign languages performed by signers using data collected from various sensors, including cameras and smartwatches. Based on the objectives of recognition tasks, SLR can be categorized into two main branches: isolated sign language recognition (ISLR) and continuous sign language recognition (CSLR). ISLR recognizes each sign language gloss individually, while CSLR interprets complete sequences of glosses as a whole. A gloss in sign language refers to a specific sign that has a similar meaning to a word or phrase in spoken languages, and it may consist of one or more small hand or body movements. In the following subsections, we will examine both ISLR and CSLR in detail.

I.4.1 Isolated Sign Language Recognition

Isolated Sign Language Recognition (ISLR) aims to identify a single sign in sign languages and translate it into a spoken language word or phrase with a similar meaning. An easy-to-understand example of ISLR is depicted in Figure I.6, which displays the recognition of letters in ALSA. However, the majority of glosses in sign languages are more complex and require video demonstrations instead of images.

ISLR research began with the recognition of simple finger-spellings in the 18th century [13], and since then, numerous studies have explored the performance of various machine learning [14] and deep learning methods [15]. While many of these studies have achieved satisfactory results on some ISLR datasets, they often overlook a crucial and challenging preprocessing step of frame selection that can improve recognition accuracy in ISLR. Frame selection is a vital preprocessing method in ISLR that reduces data size, removes noisy data, and improves recognition speed and accuracy. However, most existing works use a sliding window with a fixed stride to select some data frames from the raw data without considering frame selection. This is insufficient for ISLR due to three main reasons:

1. To start with, certain data frames in ISLR are practically irrelevant as they do not correspond to any gloss (i.e., the gloss has either not yet begun or already ended). To prevent their selection, it is crucial to determine the beginning and end points of each gloss in every data sample. However, a sliding window with a fixed stride cannot fulfill this requirement.
2. Secondly, sign language users tend to perform sign languages at varying speeds, based on their personal preferences. This can result in significant fluctuations in the time taken to complete the same gesture

in ISLR. Therefore, it is essential to adjust the selection process based on the signing speed in each data sample, to avoid choosing redundant or inadequate data frames. In practice, this can be challenging to achieve through a sliding window approach since it fails to consider signing speed.

3. Thirdly, the rapid movements made by signers can cause motion blurs and loss of data, which can significantly impact the accuracy of subsequent recognition processes in ISLR. Therefore, it is crucial to select frames with minimal motion blur and data loss during the frame selection process. However, using a sliding window with a fixed stride for frame selection cannot effectively accomplish this, as it does not consider the quality of the data frames.

I.4.2 Continuous Sign Language Recognition

Continuous sign language recognition (CSLR) aims to interpret a continuous stream of sign language glosses into a coherent sentence of spoken words with the correct order. In contrast to ISLR, CSLR involves a larger number of glosses in each data sample. An instance of CSLR is illustrated in [16], where a signer conveys the meaning of “nice to meet you” or another example like the word “premier secours” that you can find in our dataset.

In general, CSLR has a wider range of applications than ISLR because it aligns with the public's preference for communicating through sentences instead of isolated words or phrases [17]. Consequently, recent research studies in SLR have gradually shifted from ISLR to CSLR. Depending on the input sources, these studies can be divided into three major research directions: continuous vision-based sign language recognition (CVSLR) [18]-[19], continuous gesture-based sign language recognition (CGSLR) [20]-[21], and continuous multimodal sign language recognition (CMSLR) [22]-[23]. Essentially, CVSLR and CGSLR only use RGB images and smartwatch data, respectively, as their input modalities, while CMSLR combines information from multiple modalities, including RGB images, hand images, depth images, joint coordinates, and smartwatch data

I.5 Conclusion

In this chapter, we have presented an overview of Sign Language, covering its fundamental properties, primary types, and distinctive features. Our emphasis was on the Algerian Sign Language, which is the language we will be utilizing in our project. Moving forward, we will delve into related projects in the subsequent chapter.

Chapter II

Literature review

II.1 Introduction

In this chapter, we will provide an overview of the most commonly used techniques in the field of sign language recognition. We will begin by discussing the fundamental concepts of artificial intelligence, followed by an exploration of the most commonly used deep learning methods found in the literature. After that, we will delve into the evaluation metrics used to assess the performance of these models. Finally, we will conclude with a discussion of previous studies on the topic.

II.2 Artificial intelligence

Artificial Intelligence (AI) can be defined as a «collection of theories and techniques used to develop machines that are capable of simulating human intelligence» [24]. This includes a range of methods that seek to replicate various aspects of human intelligence.

AI is a technology that encompasses different techniques that replicate specific aspects of human intelligence such as learning. Its versatility is demonstrated through its various applications, including the field of sign language where it has made remarkable contributions, especially after the development of digital sign language recognition. AI is composed of two primary components: machine learning and deep learning.

II.2.1 machine learning

Machine learning is a specialized subdomain of artificial intelligence that involves developing a system capable of learning and refining its performance through the analysis of data. The central objective of this kind of learning is to automatically extract and process information, without the need for manual programming. The field of machine learning is generally organized around four main types of learning, which are detailed below.

— Supervised learning: [25] is a machine learning technique where the input data is already labeled and comes with expected output results. The primary goal of this learning approach is to enable the algorithm to learn and improve by comparing its actual output with the predicted output. This is achieved by employing models that help the algorithm make accurate predictions based on the input data.

— Semi-supervised learning: [26] is algorithm that combines labeled and unlabeled datasets to learn the underlying data structure while making predictions. This approach allows the algorithm to leverage the limited labeled data available and potentially improve its accuracy by incorporating the additional unlabeled data.

— Unsupervised learning: [25] unsupervised learning algorithms receive unlabeled data without providing them with examples of expected output results

— Reinforcement learning: [25] Reinforcement learning algorithms are based on a system of rewards and punishments learned through trial and error in pursuit of maximizing a reward. These algorithms do not require input-output data sets.

Table II.1 - Summary of machine learning approaches.

Approach	Description	Type	Example
Supervised learning	The algorithm learns from labeled data.	Classification - Regression	SVM – random forest
Semi-supervised learning	Some of the data is labeled, while others are not.	Classification – regression - clustering	Self-training semi-supervised SVM
Unsupervised learning	The algorithm is trained using unlabeled data.	Clustering – Association	k-means - k medoids
Reinforcement learning	The algorithm interacts with its environment by performing actions and learning from its mistakes and successes.	Based on a reward system.	Deep Q-Network (DQN)

II.2.2 Deep learning

Deep learning is a subfield of machine learning that utilizes artificial neural networks with multiple layers to learn complex patterns and relationships in data, drawing inspiration from the structure and function of the human brain. These models can learn to represent data without being explicitly programmed with task-specific rules, through exposure to large datasets and examples. The layers of interconnected neurons receive input from preceding neurons, and their connections determine how information is transmitted through the model.

During training, the model's architecture and parameters are optimized to improve its accuracy in making predictions on new data. Each neuron in the deep learning model receives input from preceding neurons, and their connections determine how information is transmitted through the model. By adjusting the model's architecture and parameters, DL can achieve state-of-the-art results in various applications such as computer vision and natural language processing.

II.3 Artificial neural network

Artificial neural networks are systems that draw inspiration from biological neural networks. They are capable of learning to perform tasks through exposure to various sets of data and examples, without being provided with specific task rules. A neural network consists of components such as neurons, connections, weights, biases, and a propagation function. The neurons receive input from predecessor neurons, and the connections between them govern how the output of one neuron is transmitted to the next. The propagation function calculates the input and output values and combines the functions of predecessor neurons with the appropriate weights [27].

II.3.1 The characteristics of an artificial neuron

An artificial neuron, also referred to as a 'node' or 'perceptron' (refer to figure II.1), is a mathematical function utilized to transform raw data into interpretable data. It receives inputs from prior nodes or external sources and generates an output. Each input is assigned a weight, pi, that reflects its relative importance compared to other inputs [28]. The node applies a function, f, to the weighted sum of its inputs, as shown in the following formula:

$$f(\,output\,) = f(X) = f\left(\sum_{i=1}^{n} p_i x_i + z\ biais\right) \qquad (2.1)$$

Where f is the activation function used to activate or not the neuron, n is the number of input nodes, and b is the bias used to adjust the output.

The figure presented below provides an example of an artificial neuron that receives numerical inputs (x1, x2, and x3) with corresponding weights (p1, p2, and p3). The bias term is utilized to fine-tune the output. The calculated output f_{output} can be determined using (2.1) equation.

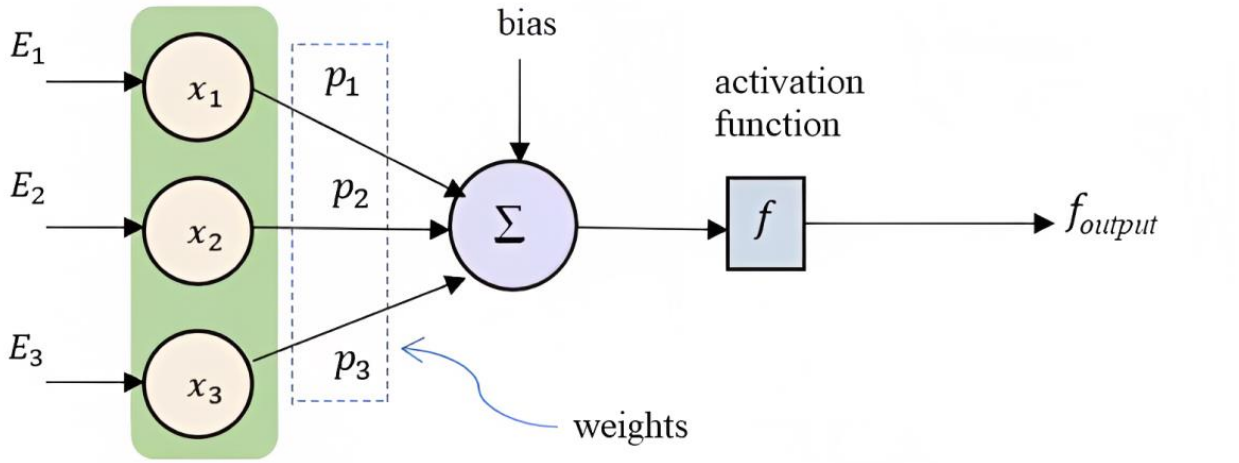


Figure II.1: Example of a simple perceptron.

II.3.2 The different layers of a neural network

— **Input layer:** This is the first layer of a neural network and is responsible for receiving input data. The number of nodes in this layer is determined by the number of input features. Each node in the input layer represents a feature of the input data [28].

— **Hidden layers:** These are the layers between the input and output layers, and their purpose is to process the input data. They are called "hidden" because their values are not directly observed. A neural network may have one or more hidden layers, and the number of nodes in each layer can vary. The nodes in each hidden layer apply a mathematical function to the input data they receive and pass the output to the next [28].

— **Output layer:** This is the final layer of a neural network, and it is responsible for producing the network's output. The number of nodes in this layer depends on the problem being solved. For example, a binary classification problem would have one output node that represents the probability of the positive class. The output layer applies a mathematical function to the input it receives from

the last hidden layer and produces the network's output [28]. Figure II.2 illustrates the general architecture of a neural network.

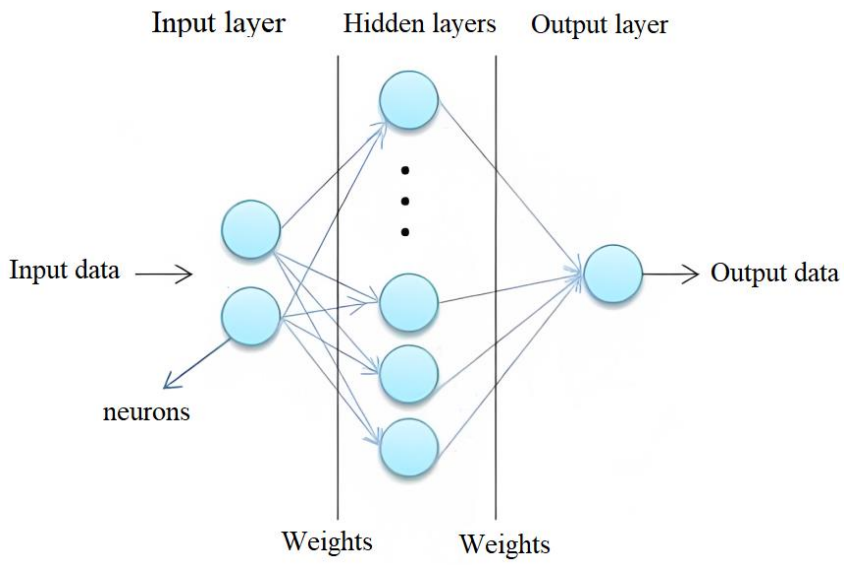


Figure II.2: general architecture of a neural network.[29]

II.3.3 Activation function

An activation function is used to introduce non-linearity into the output of a neuron, in order to mimic biological reactions. In each layer, neurons apply the activation function [30]. The following are the most commonly used functions:

The SoftMax () function: is a mathematical operation that transforms a real-valued vector into a probability vector. It is frequently employed in the final layer of a model to convert a score into a probability distribution, as determined by the following formula:

$$\sigma(\bar{X})_i = \frac{e^x i}{\sum_{j=1}^{K} e^x j} \qquad (2.2)$$

The ReLu () function: is a mathematical operation that replaces all negative values with zero, as given by the following formula:

$$f(x) = (x, 0) \qquad (2.3)$$

The Sigmoid () function: is a mathematical function that maps input values to a range between 0 and 1, according to the following formula:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2.4)$$

The tanh () function: is a mathematical operation that represents the hyperbolic tangent function. It is a shifted version of the sigmoid () function, and produces output values between -1 and 1, as given by the following formula:

$$f(x) = \frac{e^{x} + e^{-x}}{e^{x} + e^{-x}} \qquad (2.5)$$

II.3.4 Loss function

A loss function is a unique mathematical function that is applied to the entire model to evaluate its ability to predict outcomes. It measures the difference between the predicted value and the actual value. Therefore, optimizing a model involves minimizing the result of its loss function [28].

The categorical cross-entropy function: is a loss function used in multi-class classification tasks. It measures the difference between two probability distributions. [31] It is computed using the following formula:

$$L_{CE} = -\sum_{i=1} y_i log\ \ log\ (S_i) \qquad (2.6)$$

Where y_i is the predicted value, and Si is the result of the SoftMax function for the i^{th} class.

II.3.5 Batch size

Batch size is the number of training examples used in each iteration of the training process. [32] This parameter is employed to avoid overfitting. Although larger batch sizes may result in a reduction in model quality, they also generally enhance the model's generalization capability.

II.3.6 Propagation

This is a stage in the neural network learning process consisting of two steps:

- **Forward propagation:** also known as the feedforward phase, is the process of computing and storing intermediate variables (including outputs) of a neural network in order from the input layer to the output layer [33].
- **Backward propagation:** also known as backpropagation, is a step in the neural network learning process that involves calculating the gradient of the loss function with respect to the network's

weights, using the chain rule of calculus. This gradient is then used to update the weights of the network through an optimization algorithm, such as gradient descent [33].

II.3.7 The learning rate

The learning rate is a parameter that is fine-tuned in optimization algorithms to determine the step size taken at each iteration towards the minimum value of a loss function [28]. It is a crucial hyperparameter that affects both the convergence speed and the quality of the final solution. If the learning rate is too high, the optimization algorithm may fail to converge, whereas setting it too low may result in slow convergence and a suboptimal solution. Therefore, selecting an appropriate learning rate is critical in effectively training a neural network. Furthermore, techniques such as learning rate schedules and adaptive learning rate algorithms can be employed to adjust the learning rate during training, enhancing convergence and optimizing performance.

II.4 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network, also known as a CNN, is a well-known and widely used type of neural network. It utilizes a specialized mathematical operation called convolution to analyze input data. This operation is preferred over matrix multiplication due to its shift-invariant benefits and low computational costs, making it highly efficient in processing pixel data. As a result, CNNs are commonly utilized in applications related to images or videos, such as image classification [41], [42] and video captioning [43], [44].

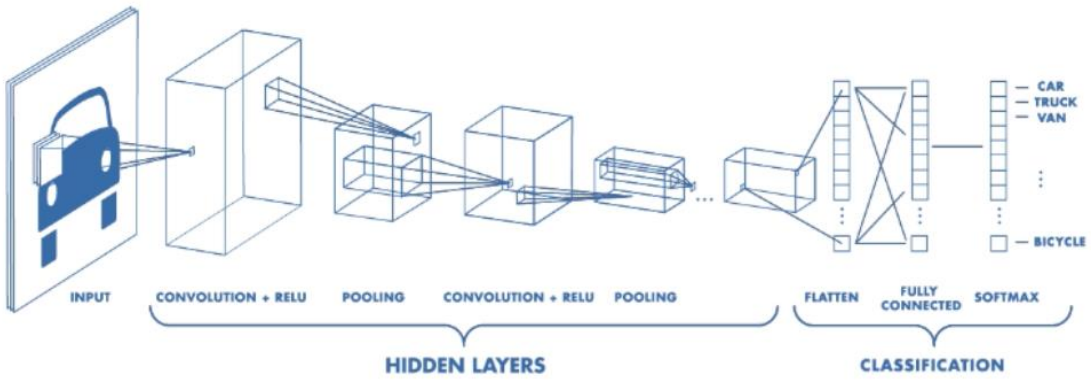


Figure II.3: The structure of a classic CNN model

Figure .3 depicts the arrangement of a CNN that features three unique layers: convolutional, pooling, and fully-connected. The purpose of each layer is distinct and crucial for the successful operation of the network.

1. **Convolutional Layer:** First of all, the fundamental layer in a CNN is the convolutional layer, responsible for convolving the input data to generate feature maps. The process involves transforming an input tensor, typically an image, of size $N \times C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}$ to an output feature map of size $N \times C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}$ via a convolution operation. The conversion equation for this process can be expressed as:

$$\text{out}(N^i, C_{\text{out}}^j) = \text{bias}(C_{\text{out}}^j) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}\,(C_{\text{out}}^j, k) \star \text{input}(N^i, k) \qquad (2.6)$$

The 2D convolution operation is represented by the symbol $\star$, where N denotes the batch size, C represents the number of channels, and W and H respectively denote the widths and heights of the vectors. The convolution process can be better understood through a visualization, which is provided in Figure II.4

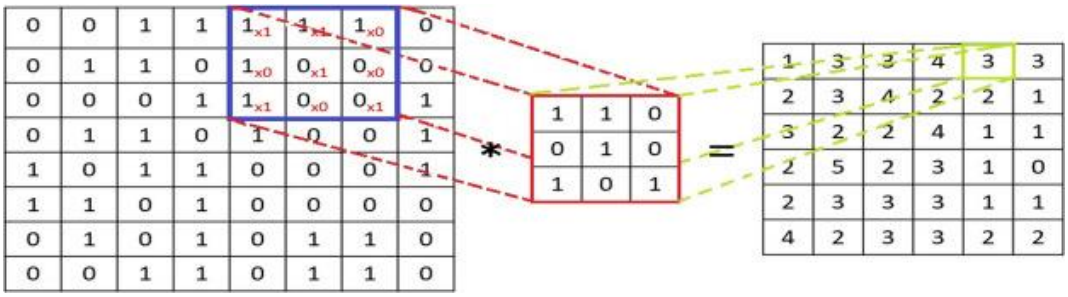


Figure II.4: The convolution process

2. **Pooling Layer:** Secondly to compress input data, pooling layers are frequently utilized in conjunction with convolutional layers by combining a group of neurons into a single neuron. Max pooling and average pooling are the two most commonly used methods for integrating neurons in pooling layers. Max pooling selects the highest value from the cluster of neurons, while average pooling computes the average value of the neurons. To better understand the process, please refer to Figure II.5 below.

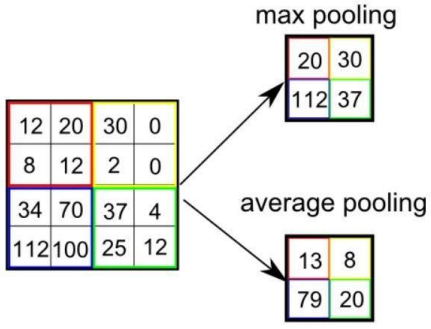


Figure II.5: The process of different pooling layer

3. **Fully Connected Layer:** Lastly, the fully connected layers are essentially similar to the classic multi-layer perceptron (MLP) [45]. This layer type consists of neurons in different layers that are all interconnected. The computation within a fully connected layer can be formulated as follows:

$$y = \phi(Wx + b) \qquad (2.7)$$

Here, ϕ denotes the activation function, x is the input vector, y is the output vector, b is the bias, and W is the weight vector. In CNNs, fully connected layers are typically used to project flattened feature vectors from feature space into the number of possible categories for recognition and classification.

The combination of convolutional and pooling layers in CNNs makes them particularly suitable for processing data with grid structures, such as images. With their ability to learn the spatial dependency among the grid points, CNNs have emerged as crucial components in numerous advanced vision-based speech and language recognition approaches.

II.5 Recurrent Neural Networks

Recurrent neural networks (RNN) are widely acknowledged in deep learning for their robust capabilities in processing sequential data. At various time points, an RNN executes the same operation in which its internal state is leveraged to process input signals and update itself, thereby allowing for storage and re-use of sequential data.

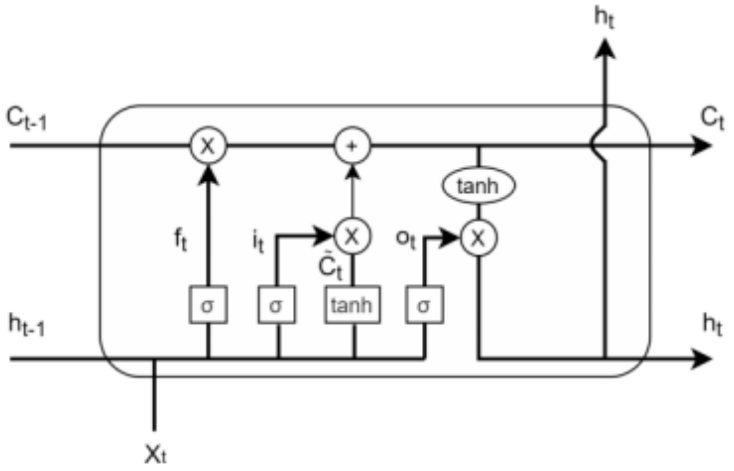


Figure II.6: The structure of a LSTM unit

However, RNNs have been demonstrated to be inadequate in maintaining information from a distant past, making them susceptible to long-term dependency issues [46]. As a result, researchers often employ a variant of RNNs known as long short-term memory (LSTM) in their investigations.

II.5.1 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) is a recurrent neural network architecture that has garnered significant attention for its ability to address the issue of long-term dependency in deep learning. In the past few years, LSTM models have gained traction in various applications that require sequential analysis, including speech and language recognition (SLR) [47]. Fundamentally, an LSTM unit comprises a cell, a forget gate, an input gate, and an output gate. The cell's function is to store the relevant information, while the gates regulate the information flow.

Figure 2.4 illustrates the architecture of a LSTM unit, wherein the symbol $\otimes$ signifies the element-wise multiplication, while $\oplus$ denotes the element-wise addition. The equations governing the behavior of a LSTM unit can be expressed mathematically as:

$$\begin{aligned} f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right), \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\ \tilde{c}_t &= tanh\,(W_c x_t + U_c h_{t-1} + b_c), \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t, \\ h_t &= o_t \otimes tanh\,(c_t). \end{aligned} \tag{2.8}$$

The equations describing the LSTM unit's structure are presented on the left-hand side of the equation, with f_t, i_t and ot representing the forget gate, input gate, and output gate activation vectors, respectively. Meanwhile, $\tilde{c}_t$ is the input of the cell, h_t is the hidden state, and c_t is the cell state at time t. On the right-hand side of the equation, x_t represents the input vector at time t, while σ denotes the sigmoid function [48] and tanh denotes the hyperbolic tangent function [49]. W, U, and b are the weight matrices and bias vectors that will be trained to optimize the LSTM's performance.

II.6 The Self-Attention Mechanism

The self-attention mechanism is a machine learning technique inspired by cognitive attention. Its function is to assign weights to different nodes of the input sequences to enable neural networks to concentrate more attention on critical portions of input data. The self-attention mechanism has emerged as a powerful tool in natural language processing and computer vision applications.

The self-attention mechanism generates three representations of key (k_i), query (q_i), and value (v_i) for each input node i along the input sequence. Specifically, the mechanism first generates these representations, which can be denoted as

$$q_i = x_i W_i^Q$$
$$k_i = x_i W_i^K \qquad (2.9)$$
$$v_i = x_i W_i^V$$

The weight vectors W_i^Q, W_i^K, and W_i^V are used to compute the representations of the input data x_i. Then, the attention scores are calculated by taking the dot product between the query of input i and the keys of all inputs (including i) and passing them through a SoftMax function. Finally, the output for input i is computed by multiplying the attention scores with their corresponding values. This process is repeated for every node in the input sequence, and can be mathematically expressed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \qquad (2.10)$$

While the constant d is utilized for regularization, and the i^{th} rows of the matrices Q, K, and V are represented as vector q_i, k_i, and v_i, respectively. The SoftMax function can be denoted as:

$$\text{softmax}\,(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K \qquad (2.11)$$

The symbol "e" within the given context refers to Euler's number.

To summarize, the self-attention mechanism enables a recognition model to access all input tokens in the input sequence and weigh them based on their relevance, allowing each token to access information from other tokens, regardless of the distance between them. Additionally, since the self-attention mechanism processes all input tokens concurrently, it can significantly reduce computation time through parallel computation. Therefore, self-attention mechanism is highly suitable for CSLR language modelling and is the focus of numerous research studies [50], [51]– [52].

II.7 The Transformer

Vaswani et al. [53] introduced the transformer in 2017, which is an advanced deep learning model that utilizes the multi-head self-attention mechanism to determine the importance of each component in sequential input data. As depicted in Figure II.5, a transformer model comprises a transformer encoder and a transformer decoder.

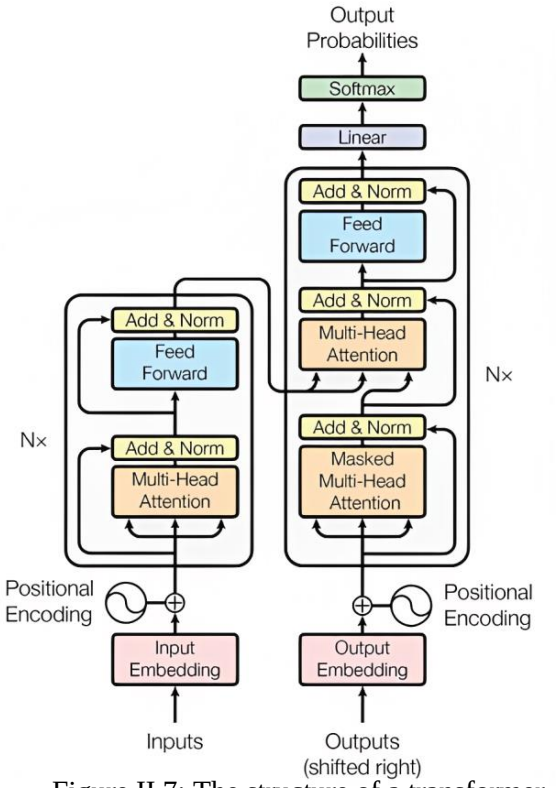


Figure II.7: The structure of a transformer.

II.7.1 The Transformer Encoder

The transformer encoder is comprised of a series of identical encoder layers [53], each consisting of a feedforward sub-layer and a multi-head attention sub-layer. The feedforward sub-layer is equivalent to the fully-connected layer discussed in Section II.X.4, which links all neurons between two neighboring layers. The multi-head attention sub-layer, on the other hand, is an enhanced variant of the self-attention mechanism, as depicted in Figure II.6. In the multi-head attention, one set of (Q, K, V) is referred to as an attention head. As previously discussed in Section II.X.6, the significance of two tokens in an input sequence can be calculated using this attention head through Equation (2.10). However, there may be numerous types of "relevance" between tokens. For instance, one attention head may represent time relevance, while another may denote relationships between verbs and their objects. As a result, the transformer model incorporates multiple attention heads in each layer to represent various interpretations of "relevance." More formally, the operation in multi-head attention can be represented as:

$$
\begin{aligned}
Q_i &= xW_i^Q \\
K_i &= xW_i^K \\
V_i &= x^V W_i^V \\
head_i &= Attention\,(Q_i, K_i, V_i) = softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i \\
MultiHead\,(Q, K, V) &= concatenate\,(head_1, \ldots head_i) W^o
\end{aligned}
\tag{2.12}
$$

The aforementioned equations utilize (Q_i , K_i , V_i) to represent an attention head. Additionally, the weight matrices (W^o, W_i^Q, W_i^K, and W_i^V) are included for the purpose of training.

II.7.2 The Transformer Decoder

The transformer decoder [53] has a structure that is similar to the transformer encoder, with an additional masked multi-head attention sub-layer in each decoder layer. The transformer decoder takes input sequence embeddings as inputs, but to ensure accurate recognition results, the transformer model must not use information from the current or future output to predict the current output. To achieve this, a masked multi-head attention mechanism is used to partially mask the output sequences and prevent the leftward flow of information. In mathematical terms, the masked multi-head attention scores the current and future outputs as -∞, ensuring that no information from them is used during recognition.

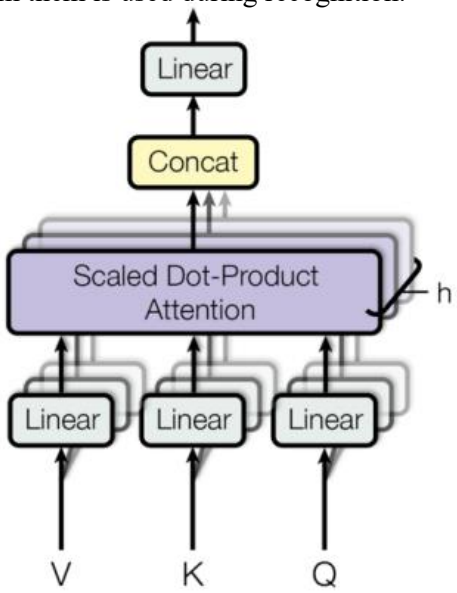


Figure II.8: The multi-head attention in a transformer.

II.8 Autoencoder

Autoencoder is a powerful technology used for synthesizing data in artificial neural networks, first introduced by Ballard in 1987 as a method to learn meaningful representations from data in an unsupervised

manner [54]. The autoencoder architecture comprises three parts: the encoder (fenc), the latent factors (code), and the decoder (fdec), as illustrated in Figure 2.4. The number of neurons in the output layer is the same as the number of inputs, with a symmetric hidden layer in the middle [56].

The AE, or incomplete autoencoder, gets its name from the fact that its internal representation has a lower dimension than the input data. A critical feature of the AE is the hidden layer's lower dimensionality compared to the input or output layer. This characteristic allows the AE to efficiently perform dimensionality reduction on the data, aligning with the core principles of principal component analysis.

In the same way as the general deep learning model, the learning method of the AE utilizes a function represented by f. The input to this function is the original data $\boldsymbol{qi}$, and the output is $f(qi)$ for i = 1, . . .I. The neural network in AE is trained with a loss function, which is as follows:

$$\sum_{i=1}^{l} \|\boldsymbol{q}_i - f(q_i)\|_2 \qquad (2.13)$$

Where $\|q_i\|_2 := \text{sqrt}\,(o^2 + \cdots + o^2)$.

The goal of AE is to minimize the loss function, and once trained, the AE is able to

$$\widehat{\boldsymbol{\beta}} := argmin \sum_{f}^{l} \|\boldsymbol{q}_i - f(q_i)\|_2 \qquad (2.14)$$

We can generate synthesized data using the following approach:

$$\boldsymbol{q}_i^* := \hat{f}(q_i), i = 1, \ldots, I \qquad (2.15)$$

The encoder is a fully connected feed-forward non-recurrent artificial neural network (ANN) that functions as an encoding function F_e, taking an input data vector $x \in \mathbb{R}^D$and mapping it to a hidden $z \in \mathbb{R}^K (z = F_e(x))$. The latent factors section is a single layer of an ANN, with a size smaller than that of the input layer, typically denoted as K, and must be meaningful and compact [56].

The decoder is the inverse operation of the encoder, serving as a fully connected feed-forward non-recurrent ANN. It is considered a decoding function F_d that takes the latent representation z as input and maps it back to a reconstructed vector $\hat{x} \in \mathbb{R}^D (\hat{x} = F_d(z))$ [56]. Thus, the output can be expressed as $\hat{x} = F_d(F_e(x))$

Autoencoders learn to copy inputs to outputs, while being trained to limit the size of the latent representation or add noise to the input and then restore the original data [55]. This allows for data synthesis and representation learning in artificial neural networks [54].

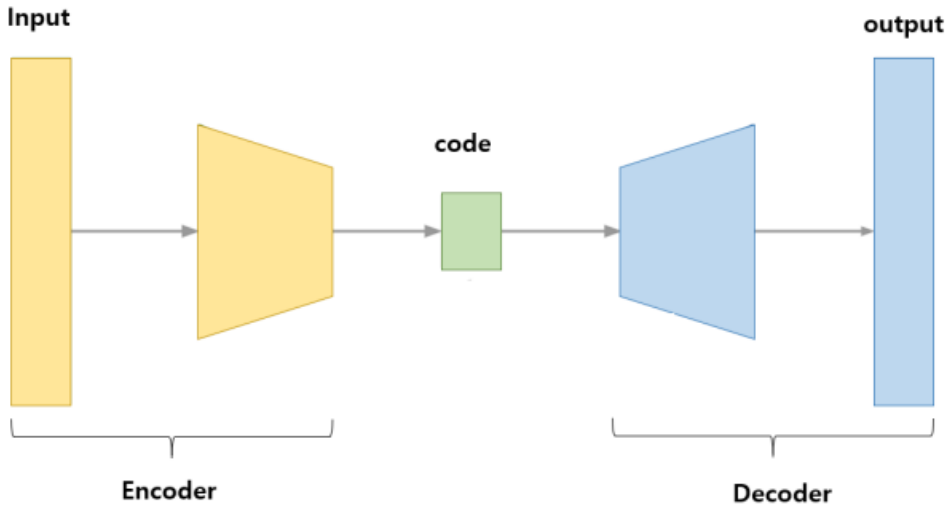


Figure II.9: Deep Autoencoder architecture.

II.9 Evaluation metrics of models

Evaluation metrics play a crucial role in assessing the performance of machine learning models. They enable a clear understanding of a model's effectiveness and allow for comparisons and enhancements. These numerical indicators provide information on the accuracy, error rates, and alignment with ground truth for a model's predictions. By examining these metrics, we can identify the strengths and limitations of different models, ultimately choosing the best fit for a specific task.

But before diving into various evaluation metrics, it's important to understand some basic terms that are commonly used in assessing the performance of classification models. These terms are True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN):

- **True Positives (TP):** The instances where our prediction is positive, and the actual result is also positive.
- **True Negatives (TN):** The instances where our prediction is negative, and the actual result is indeed negative.

— **False Positives (FP):** The instances where our prediction is positive, but the actual result turns out to be negative.

— **False Negatives (FN):** The instances where our prediction is negative, yet the actual result is in fact positive.

From the most common evaluation metrics, we can identify:

- **Accuracy:** The proportion of correctly classified instances to the total number of instances. It is a widely used metric for classification tasks, indicating how well a model performs. It is calculated as follows:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \qquad (2.16)$$

- **Precision:** The ratio of true positive predictions to the total positive predictions (true positives + false positives). Precision measures a model's ability to correctly identify positive instances, assessing its reliability. The formula for this metric is:

$$Precision = \frac{TP}{(TP + FP)} \qquad (2.17)$$

- **Recall (Sensitivity):** The ratio of true positive predictions to the total actual positive instances (true positives + false negatives). Recall evaluates a model's ability to capture all the positive instances, reflecting its completeness. The expression for this metric is:

$$Recall = \frac{TP}{(TP + FN)} \qquad (2.18)$$

- **F1-Score:** The harmonic means of precision and recall, providing a balance between the two metrics. It is particularly useful when dealing with imbalanced datasets, as it considers both precision and recall in its calculation.

$$\boldsymbol{F1 - Score} = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \qquad (2.19)$$

- **Confusion matrix:** table that illustrates the performance of a classification model by showing the counts of correct and incorrect predictions for each class. In a binary classification, the matrix consists TP, TN, FP, FN. It helps in calculating evaluation metrics like accuracy, precision, recall, and F1-score to rate a model's performance.

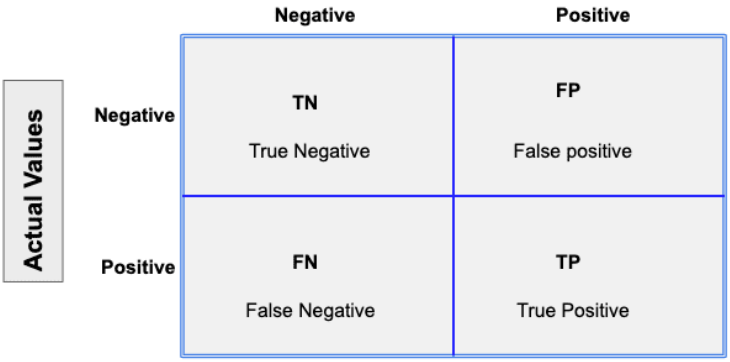


Figure II.10: confusion matrix Table structure

- **Mean Squared Error (MSE):** The average squared difference between the predicted and actual values. It is commonly used for regression tasks, quantifying the average prediction error.
- **Mean Absolute Error (MAE):** The average absolute difference between the predicted and actual values. It is less sensitive to outliers than MSE, offering a robust measure of prediction error in regression tasks.

II.10 Optimizers

During the training of machine learning and deep learning models, an optimizer is a computational method that adjusts model parameters, such as weights and biases. The optimizer's purpose is to decrease a specific metric, previously discussed in Chapter II.3.4 as the loss function. By accomplishing this, the optimizer contributes to improving the model's effectiveness and precision. By reducing the loss function, the optimizer helps improve the model's performance and accuracy. Various optimization techniques exist, and choosing the right one depends on the specific problem and model architecture. While there are many optimization techniques available, we would like to shift our attention to a few specific familiar ones:

— **Stochastic Gradient Descent (SGD):** A simple and popular optimization algorithm that updates the model's parameters based on the gradient of the loss function with respect to each parameter. It is a foundational optimization technique, but it can sometimes suffer from slow convergence or getting stuck in local minima.

— **Momentum:** A variation of SGD, it incorporates a momentum term that helps the optimizer move faster towards the global minimum by accumulating gradients from previous steps. This can lead to faster convergence and a reduction in oscillations during the training process.

— **Adam (Adaptive Moment Estimation):** A highly efficient optimization algorithm that combines the benefits of both momentum and RMSprop. It maintains separate moving averages for the

gradients and squared gradients, and adjusts the learning rate for each parameter adaptively. Adam has gained popularity for its effectiveness in a wide range of deep learning tasks.

II.11 Related Works

II.11.1 SLR with CNNs

In 2015, Huang et al. [57] made the first attempt to use a CNN in SLR. Their study on a real-world dataset shows that CNN-extracted features are more effective than traditional hand-crafted features. That same year, Kang et al. [58] also applied a CNN in SLR while using depth images as input data. They achieved a remarkable accuracy of 99.9% in recognizing 31 alphabets and numbers by tuning the network configurations. In 2016, Koller et al. [59] proposed to embed a CNN within the expectation-maximization (EM) algorithm, making CNN training more efficient. With this method, their CNN model is trained on 1 million hand images, surpassing the state-of-the-art methods at that time. Additionally, CNNs are used in gesture-based SLR, such as Kim et al. [60] who built a CNN-based hand gesture recognition system using reflected impulses with an average recognition accuracy over 90%. Shin et al. [61] also employed a CNN to process data collected from an electromyography (EMG) sensor and an inertial measurement unit (IMU) sensor.

Furthermore, in order to improve recognition performance, numerous CNN-based network structures pretrained on large-scale image datasets are used in SLR. For instance, Mercanoglu et al. [62] employed VGG16 [63] to extract representative features from RGB and depth images, achieving high accuracy in recognizing Italian Sign Language. Along with VGG16, other well-known CNN-based models like Residual Neural Network (ResNet) [64], [65] and Inception [66], [67] have also been utilized in SLR.

Moreover, researchers have explored the potential of combining machine learning (ML) approaches with CNNs in SLR. For instance, Li et al. [68] utilized a CNN and a principal component analysis (PCA) method to develop a hierarchical model for recognizing fingerspelling in American Sign Language. In 2019, a SVM and a CNN were integrated as a feature descriptor in [69] to improve recognition accuracy. Besides, CNNs have also been combined with hidden Markov models (HMMs) in [70] and [71].

II.11.2 SLR with LSTM Models

In 2016, Liu et al. [72] were the first to adopt a Long Short-Term Memory (LSTM) model in SLR. They developed an end-to-end LSTM-based model to recognize the Chinese Sign Language and achieved superior performance compared to other research studies using Hidden Markov Model (HMM) methods. Similarly, Avola et al. [73] utilized an LSTM in their recognition model for the American Sign Language.

Variants of LSTMs have also been explored in SLR studies. For instance, bidirectional LSTMs were utilized in [65] and [74] to enhance the recognition accuracy. Additionally, Mittal et al. [21] introduced a new reset gate in their LSTM model to enable independent recognition of each gloss. Furthermore, some studies have investigated the integration of LSTMs with HMM approaches [71], [75] and CNN models [74], [76]. Generally, the combination of a CNN and LSTM is the most popular network structure in SLR, with a large number of research studies being conducted [62], [77], [47], [78]. In these studies, the CNN is employed for feature extraction, while the LSTM is responsible for sequential analysis

II.11.3 SLR with the Transformer

In 2019, Zhang et al. [18] were the first to introduce a transformer and CNN model in SLR. According to their experimental results, the transformer outperformed other existing methods. Since then, numerous studies have been conducted to explore the potential of combining transformers with various CNN models such as Inception [79], ResNet [19], and other customized CNNs [85], [80] for SLR. Typically, CNN models are utilized for feature extraction, while transformers carry out sequential analysis and language modeling tasks. Additionally, researchers in SLR are also investigating the integration of transformers with new CNN technologies [81]. For instance, in 2021, Pan et al. [51] proposed a transformer-based model that utilizes content-aware and position-aware convolutions for SLR. Moreover, in SLR, joint coordinates of signers are widely used as input features for transformer models [82], [83], [50].

In recent years, researchers in the field of natural language processing have introduced several effective variants of the transformer model. One notable example is the bidirectional encoder representations from transformers model, which has been shown to be more powerful than the original transformer in language modeling [84]. However, existing studies typically make only minor modifications to the original transformer, which limits their ability to achieve further improvements in recognition accuracy in the context of SLR.

II.12 Conclusion

In this chapter, we present a thorough overview of the theoretical aspects and characteristics of machine learning. We also conduct a critical review of the current research in this field, which employs various machine learning and deep learning technologies such as CNNs, RNNs, LSTMs, self-attention mechanisms, and transformers. Additionally, we provide in-depth explanations of these technologies. In the upcoming chapter, we will discuss the steps involved in implementing our model and its application in SLR classification

Chapter III

Conception

III.1 Introduction

In this chapter, we present an extensive examination of our video classification system and the techniques implemented in it. We start with an overview of our methodology and after that we introduce our video database that we created. Then, we discuss the various pre-processing techniques applied to the videos in the database, as well as the different variant techniques used to extract the most critical keyframes. Following that, we discuss the architecture of the proposed neural network and the diverse metrics used to evaluate the model's performance. Lastly, we conclude the chapter with the use case diagram of our application and the Desktop implementation.

III.2 Overview

In proposals to solve the problem of SLR, we propose a solution in the form of an application that can interpret sign language in real time. This application is aimed at assisting not only the deaf community but also those who interact with them. The development process begins with the creation of a unique sign language dataset for the Algerian sign language (SIGAL), an integral component of our work. This dataset then undergoes careful optimization to ensure its suitability for real-time processing. The foundation of our system is a machine learning model, designed specifically to interpret the complex movements that form sign language. This model is trained and evaluated using our unique dataset, with the ensuing results analyzed to determine the model's effectiveness in recognizing sign language. Figure III.1. presents the proposed approach for developing our solution. we will delve into each phase in more detail, examining its nuances and subtleties where each phase represent:

- Phase 01: **Algerian Sign Language Dataset (SIGAL).**
- Phase 02: **Pre-processing.**
- Phase 03: **The Data Augmentation.**
- Phase 04: **Model Development.**

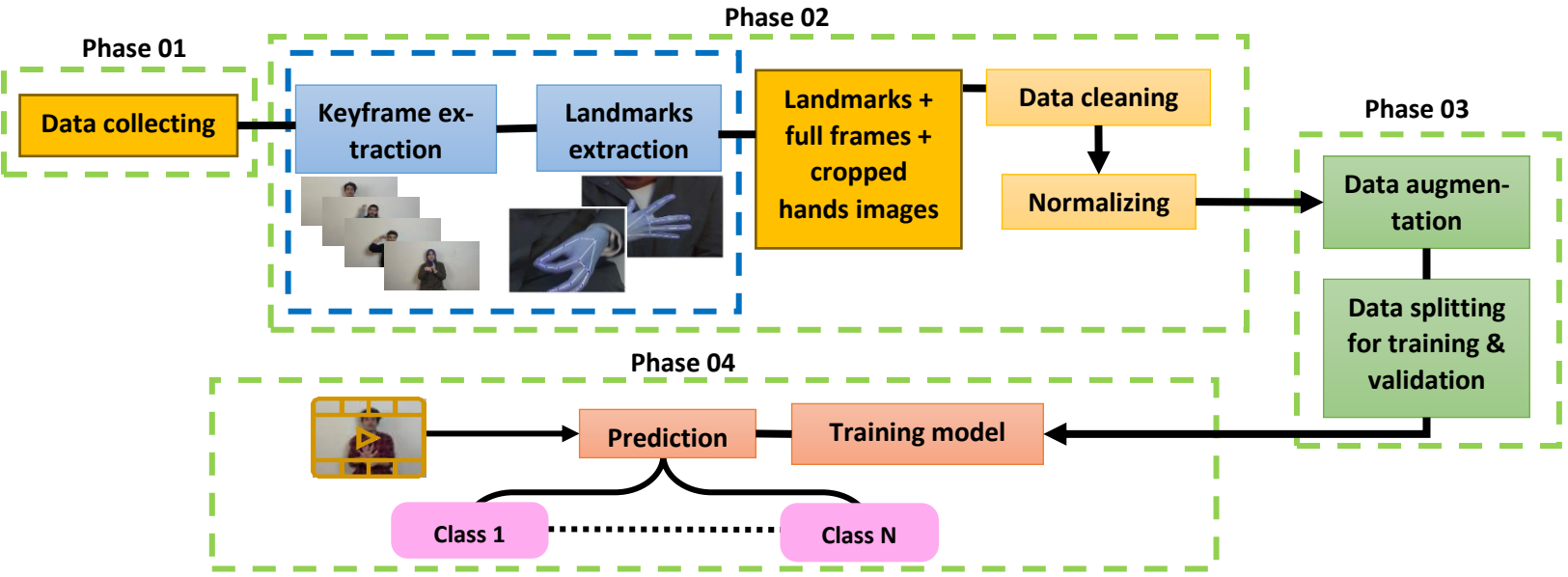


Figure III.1: General diagram of the proposed approach

III.3 Algerian Sign Language Dataset (SIGAL)

In this passage, we address the challenge of sign language recognition, which varies significantly across different regions of the world, with each having its own lexicon and unique set of signs. Consequently, the problem of sign language recognition requires a different approach in each region, as new movements, handshapes, or combinations thereof necessitate new training data and may introduce new challenges that were not previously considered. Notably, to our knowledge, there are no existing datasets available for Algerian Sign Language (LSA), with only a handful of dictionaries focusing on teaching the language. However, these resources were not designed for automatic recognition and suffer from poor image quality, limited annotations, and only one sample for each sign, rendering them unsuitable for training automatic recognition systems. Thus, there is a pressing need for a research-level dataset that encompasses the group of signs used in LSA. This master's document presents such a dataset, named SIGAL, consisting of 89 signs from LSA, recorded with normal RGB cameras.

III.3.1 Dataset

The development of an Algerian Sign Language (LSA) database, aimed at facilitating the creation of a dictionary and training an automatic sign recognition system. This database contains 4885 (Figure III.3) videos showcasing ten non-expert subjects and one deaf individual each perform five repetitions of 89 distinct sign types (Table III.1). The selected signs, comprising both verbs and nouns, represent the most commonly used elements of the LSA lexicon and are classified into nine thematic categories: time, colors,

places, justice-related terms, medical terminology, months, interrogatives, family, general communication, and specific verbs. Figure III.2 provides several examples of these signs.



Figure III.2: snapshots of six key-frames random distinct signs extracted from the SIGAL database

Table III.1 showing the ID, name, and hand(s) used in each sign of the dataset. The "H" column specifies whether the sign was performed with the right hand ("R"), left hand ("L"), or both hands ("B").

Table III.1: provides a detailed list of the signs used in our SIGAL dataset, along with the ID and hand(s) used for each sign.

ID	Name	H	ID	Name	H	ID	Name	H	ID	Name	H
00	Semaine	B	24	Aéroport	B	48	Juillet	R	72	Prénom	B
01	Dimanche	R	25	Banque	B	49	Aout	R	73	Nom de famille	B
02	Lundi	R	26	Restaurant	B	50	Septembre	B	74	Oui	R
03	Mardi	R	27	Hôtel	R	51	Octobre	B	75	Non	R
04	Mercredi	R	28	Rue	B	52	Novembre	B	76	Bonjour	R
05	Jeudi	B	29	École	B	53	Décembre	B	77	Merci	R
06	Vendredi	R	30	Université	R	54	Mère	R	78	Derien	R
07	Samedi	R	31	Loi	B	55	Père	R	79	Question	R
08	Année	B	32	Avocat	B	56	Frère	R	80	Message	B
09	Heur	B	33	Juge	B	57	Sœur	R	81	Toilette	R
10	Minute	B	34	Liberté	B	58	Fille	R	82	Quand	B
11	Couleur	R	35	Témoin	B	59	Garçon	R	83	Comment	B
12	Blanc	R	36	Héritage	B	60	Entrer	B	84	Ou	R
13	Noir	R	37	Malade	B	61	Sortir	B	85	Combien	B
14	Rouge	R	38	Vaccination	R	62	Aimer	R	86	Hier	R
15	Bleu	R	39	Médecin	R	63	Acheter	B	87	Demain	R
16	Jaune	R	40	Médicament	R	64	Manger	R	88	Rendez-vous	B
17	Vert	R	41	Premier secours	B	65	Chercher	B			
18	Rose	R	42	Janvier	R	66	Demande	B			
19	Hôpital	B	43	Février	R	67	Écrire	B			
20	Police	B	44	Mars	B	68	Appeler	B			
21	Tribunal	B	45	Avril	B	69	Perdre	B			
22	Mosquée	B	45	Mai	R	70	Trouver	R			
23	Pharmacie	R	47	Juin	R	71	Traduire	B			

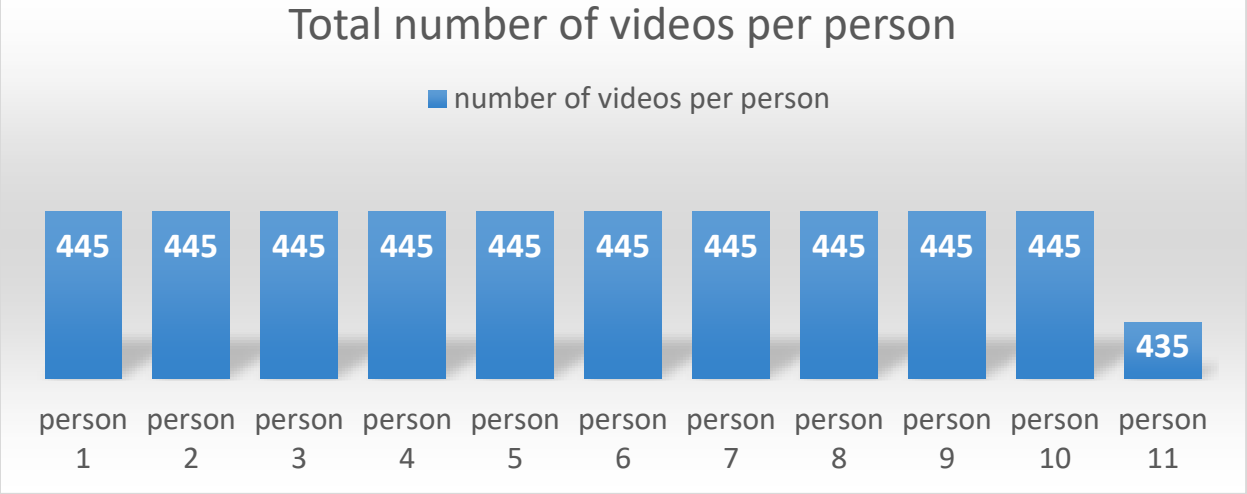


Figure III.3: The total number of videos per person

III.3.2 Recording

The SIGAL database was captured in eleven different sessions, primarily within the indoor setting of our university classrooms, with the management kindly approving our use of the premises. Each session focused on a specific set of signs, with most signs being captured in a single session. However, some signs required additional sessions for refinement and correction. More repetitions than required were captured during each session to account for potential errors. We utilized indoor lighting for most of the recordings and supplemented it with an artificial white projector to ensure consistent lighting across all sessions, regardless of the different recording times.

Due to unforeseen circumstances, we had to adjust the recording location for Subject 11, who was unable to participate in the same session as the others. We arranged a session at the deaf school in Algeria, where the individual was located. Despite this change, we strove to maintain the same recording conditions to preserve the dataset's quality and integrity.

The majority of the subjects wore dark-colored clothing and performed the signs against a white wall background to maintain consistency across all videos. To enhance the database's diversity and realism, we imposed minimal constraints on the subjects while performing the signs, ensuring a natural execution and a more realistic representation of real-world signing conditions. We believe that incorporating non-expert subjects added a greater level of diversity to the dataset.

All recordings were captured using a OnePlus 8 smartphone, which records videos at 4K 30fps with a 16:9 ratio. A tripod was set 1.6 meters away from the wall and at a height of 1.4 meters to maintain consistency across all recordings (more details illustrated can be seen in figure III.4).

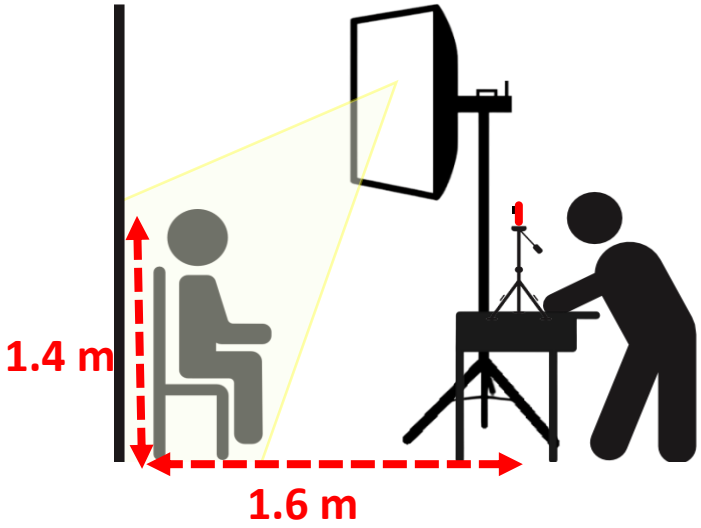


Figure III.4: the configuration of the recording setup

For more information on SIGAL dataset please refer to the annex chapter.

III.4 Pre-processing

After successfully creating the SIGAL dataset, the next crucial step in our methodology was data preprocessing. Data preprocessing is a fundamental stage in any machine learning project. It involves preparing and 'cleaning' the raw data to make it suitable for a machine learning model. For our project, preprocessing was particularly important as it ensured our dataset's suitability for real-time processing, a key requirement for our sign language interpretation application. In this section, we will detail the specific preprocessing techniques we applied to our SIGAL dataset and discuss any challenges encountered and how we addressed them

III.4.1The Key Frame Selection Mechanism

During the development of our dataset, we employed various frame extraction methods. Each method presented unique strengths and weaknesses, and a thorough evaluation was carried out to ascertain their suitability based on our project's specific needs. Frame extraction plays a pivotal role when working with video-based datasets, as it facilitates the conversion of continuous video motion into a series of separate frames, primed for individual processing and analysis. It's essential to recognize that the choice of frame extraction method can considerably impact the overall effectiveness of a machine learning model trained on video data. In the subsequent sections, we will delve into each method we implemented, explaining its conceptual framework, advantages, and disadvantages.

- **Variant 01:** This first frame extraction technique we employed upon extracting a target number of frames from a given video file. The process begins by calculating the total number of frames in the video and the necessary frame step to achieve the targeted frame count. For each step, the function checks the quality of the frame by converting it to grayscale and then calculating the variance of its Laplacian, which serves as an indicator of blur. If the variance is above a certain threshold (in this case 4), the frame is considered sharp enough to be included in the output. After checking for blurriness, the frame is resized and normalized before being appended to the list of frames.

  If the list of frames ends up with more than the targeted number, the list is truncated to meet the target. However, if there are fewer frames than the target, the list is extended by duplicating frames until it matches the target count. As a result, this technique ensures that every processed video has the same number of frames, which is crucial for subsequent machine learning tasks Figure III.5 showcases the process of keyframe extraction, along with the comparison between a frame.



Figure III.5: sample of key-frames extraction using variant 01

A. Advantages:

— This technique is computationally efficient, as it only processes a subset of frames rather than every frame in the video.

— By checking for frame blurriness, it ensures that only high-quality, sharp frames are included in the output, which can improve the performance of subsequent recognition tasks.

B. Disadvantages:

— The method may not be as robust as the other techniques in extracting the most critical frames since it relies on a fixed frame step.

— The preprocessing steps are less extensive than in the other techniques, potentially leading to less optimized input data for the model.

As shown in Figure III.6, the chart flow for Variant 01 of the key-frame extraction process is illustrated.

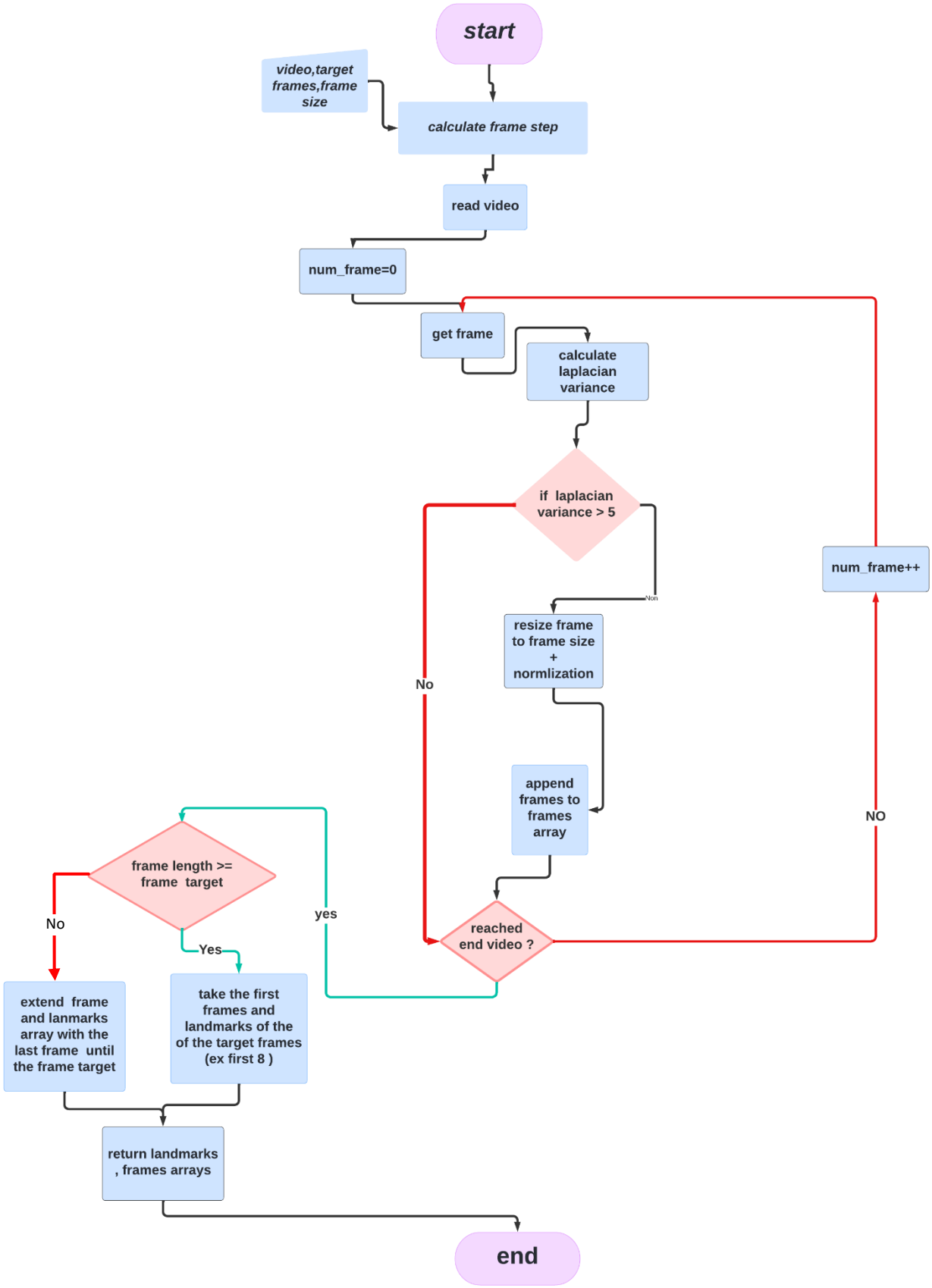


Figure III.6: Chart flow of Variant 01 for Key-frame Extraction

- **Variant 02:** The second technique for video preprocessing operates in a different manner, it is built upon OpenCV and MediaPipe, two powerful libraries for computer vision tasks. The core idea behind this technique is to extract a set of key frames from each video, which are most representative of the sign being performed. These frames are then preprocessed and used to train the machine learning model. Here is a step-by-step explanation of the process:

    1. **Hand Landmark Extraction:** For each frame in the video, we used MediaPipe's Hand solution to detect hand landmarks. These landmarks, essentially points of interest on the hand, provide valuable information about the hand's position and orientation in each frame. We then computed several features from these landmarks, including the centroid (average position), average distance from the centroid, angle, and length ratio.

    2. **Feature Scaling and Clustering:** After extracting these features from all frames, we performed feature scaling to standardize the feature values, which helps the subsequent clustering step. We used the K-means clustering algorithm to group the frames based on their features. The number of clusters is equal to the number of target frames we want to extract from each video (8 frames in Figure III.4).

    3. **Key Frame Selection:** For each cluster, we selected the frame whose features are closest to the cluster center as the representative key frame for that cluster. This ensures that the selected key frames are most representative of the different states of the sign being performed in the video.

    4. **Frame Preprocessing:** Finally, each key frame was subjected to a series of preprocessing steps to make it suitable for input into the machine learning model. These steps include converting the frame to gray-scale, applying a sharpening filter to enhance the details, removing the background using GrabCut method, and resizing and normalizing the frame.



Figure III.7: sample of 8 key-frames extraction using variant 02

A. Advantages:

— This technique offers several advantages. Firstly, the detailed hand landmark extraction process allows for a comprehensive understanding of the signs being performed, capturing even subtle changes in hand movements. Secondly, the incorporation of clustering helps in selecting the most representative frames, thereby reducing redundancy and retaining important information. Lastly, the preprocessing steps including resizing, grayscale conversion, sharpening, background removal, and normalization aid in enhancing the quality of the frames and making them more suitable for a machine learning model. Therefore, this method can be particularly useful for applications prioritizing high accuracy over real-time performance.

B. disadvantages:

— This method's main drawback is its lengthy processing time, making it less suitable for real-time applications. Extracting hand landmarks from each video frame is computationally intensive, leading to delays. Furthermore, the clustering step, even using MiniBatchKMeans, contributes to the time overhead. For instance, the technique takes approximately 15 seconds to preprocess a single video, an impractical duration for real-time systems. Therefore, despite its potential accuracy, the trade-off between speed and precision renders it less optimal for real-time sign language recognition tasks.

Figure III.8 depicts the chart flow for Variant 02 of the key-frame extraction process.

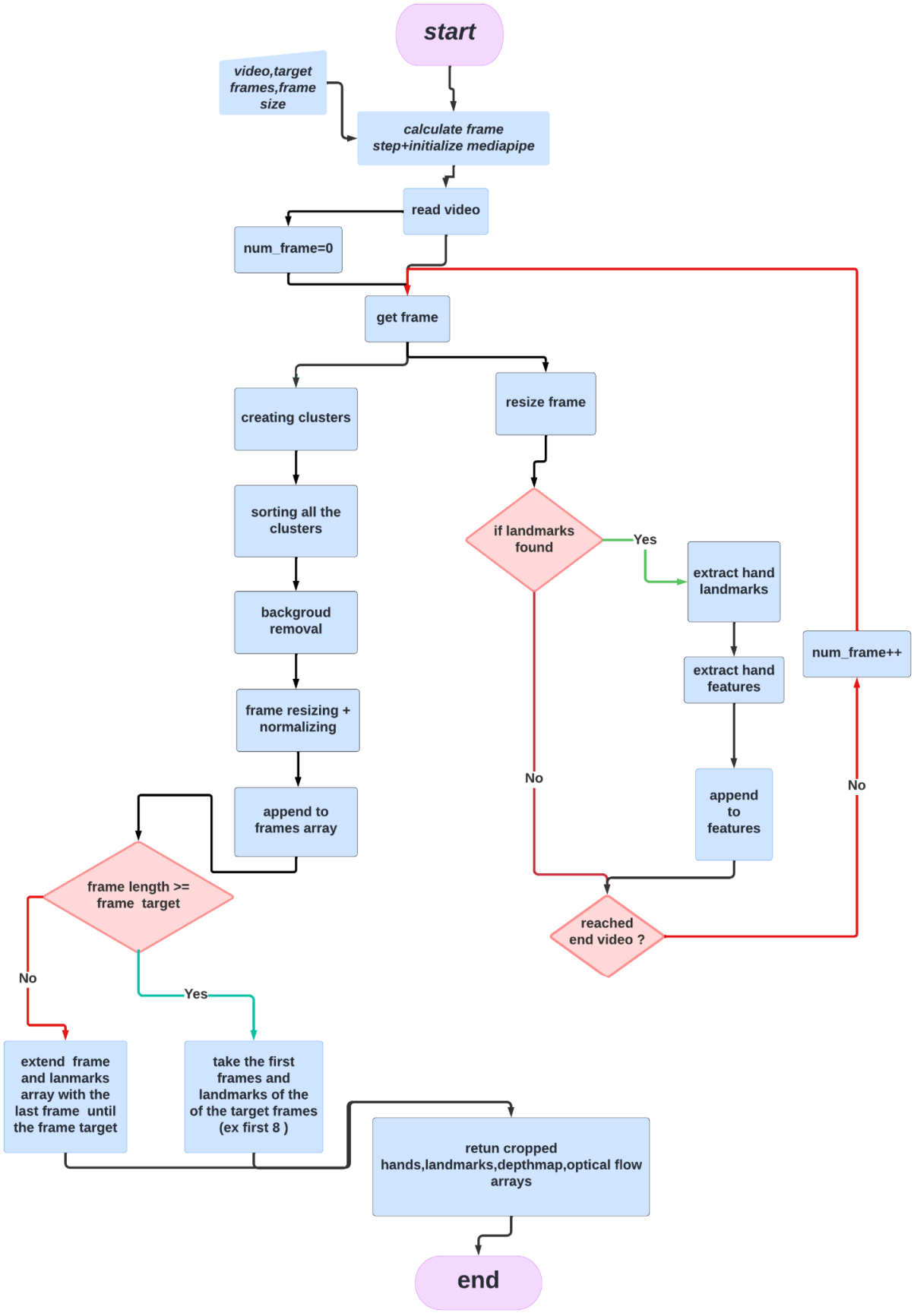


Figure III.8: Chart flow of Variant 02 for Key-frame Extraction

- **Variant 03:** The third technique focuses on extracting hand landmarks from each frame using MediaPipe's Hand solution only. The process involves reading each frame of the video and converting it to RGB before processing with MediaPipe Hands. If hand landmarks are detected, they are flattened into a 1D array and appended to a list. If no landmarks are found, a zero-filled array is added instead. This continues until the maximum number of frames is reached, or the video ends.

A. Advantages:

— By using MediaPipe's Hand solution, this technique effectively captures the spatial coordinates of the hand in each frame, which is crucial for sign language recognition.

— The process of padding ensures that all arrays have the same shape, which is a requirement for most machine learning models.

B. disadvantages:

— This technique doesn't incorporate any image preprocessing steps such as grayscale conversion, resizing, or normalization. This could potentially affect the quality of the input data for the model.

— The method relies on a fixed maximum number of frames, which may not be optimal for all videos

The process for key-frame extraction can be visualized in Figure III.9, which shows the chart flow for Variant 03

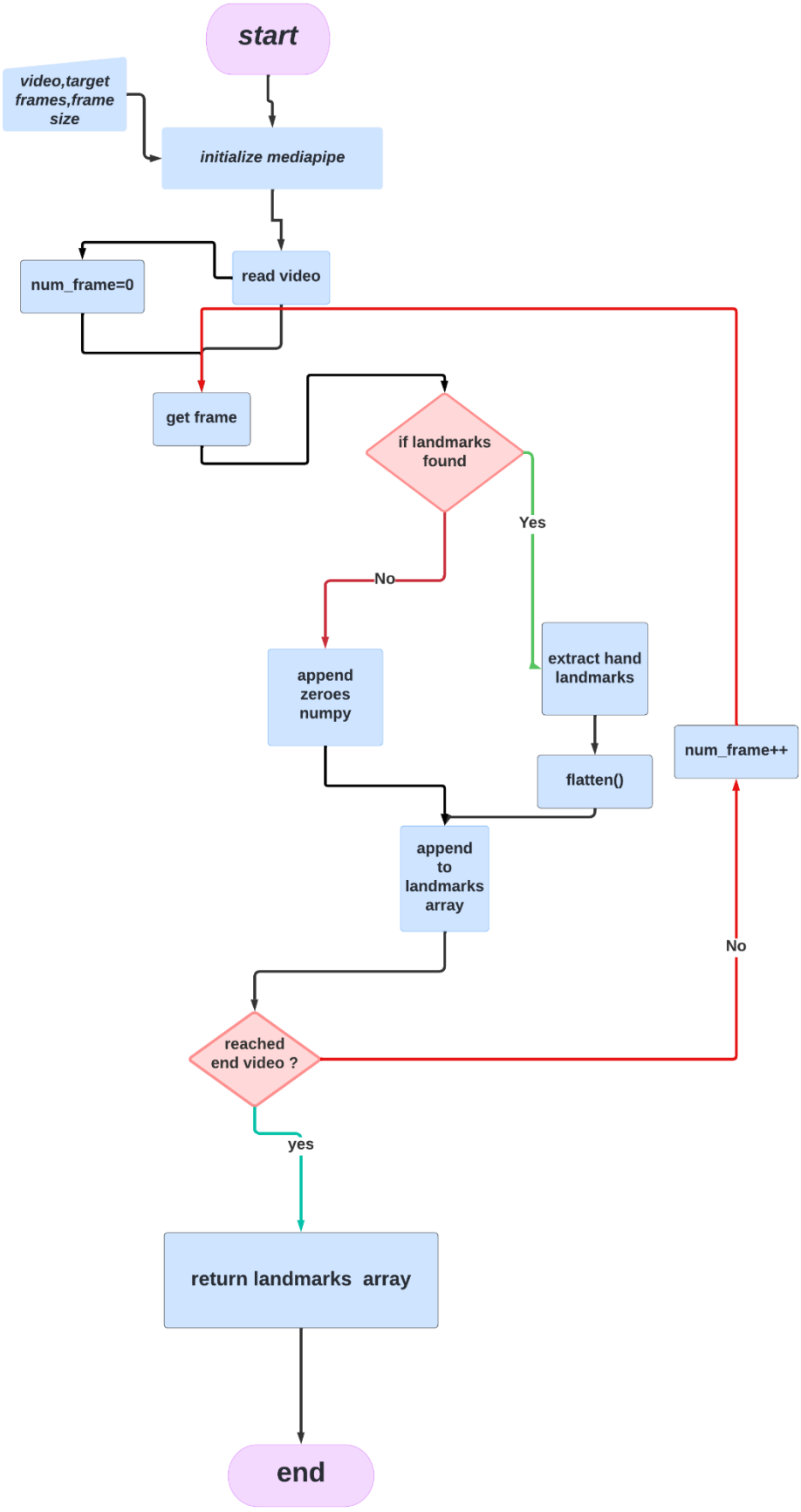


Figure III.9: Chart flow of Variant 03 for Key-frame Extraction

- **Variant 04:** In the fourth technique, the video preprocessing combines both hand cropping and frame resizing, as well as extracting hand landmarks using MediaPipe Hands. The process involves reading each frame of the video, converting it to RGB, and processing it with MediaPipe Hands. If hand landmarks are detected, they are used to create a bounding box around the hand, which is then scaled up and cropped. If two hands are detected, the cropped hand images are resized, stacked horizontally, and normalized. If only one hand is detected, the cropped hand image is resized and normalized. For videos with no detected hand landmarks, a zero-filled array is added. Here are the detailed steps that this method follows:

1. **Video Processing:** The video is read and a loop is initiated to process each frame. Frame step size is calculated to reach the target number of frames.

2. **Hand Detection and Feature Extraction:** For each chosen frame (based on the frame step), the function reads the frame, applies the MediaPipe Hands model to identify hand landmarks. If any hand landmarks are detected, the function calculates the bounding box coordinates of the hand(s), scales them up for better visibility, and stores these landmarks.
3. **Frame Processing:** Depending on the number of hands detected (one or two), the function either resizes a single hand frame or stacks two hand frames together horizontally after resizing each of them to half the target frame size (Figure III.10). The processed frames are then normalized and added to a list.
4. **Frame Selection and Duplication:** If the number of processed frames and corresponding landmarks exceed the target, the function selects the first few up to the target number. If the number of processed frames or landmarks is less than the target, it duplicates frames and landmarks until it reaches the target number.
5. **Completion:** Finally, the function closes the video and returns the processed frames and corresponding landmarks as numpy arrays.

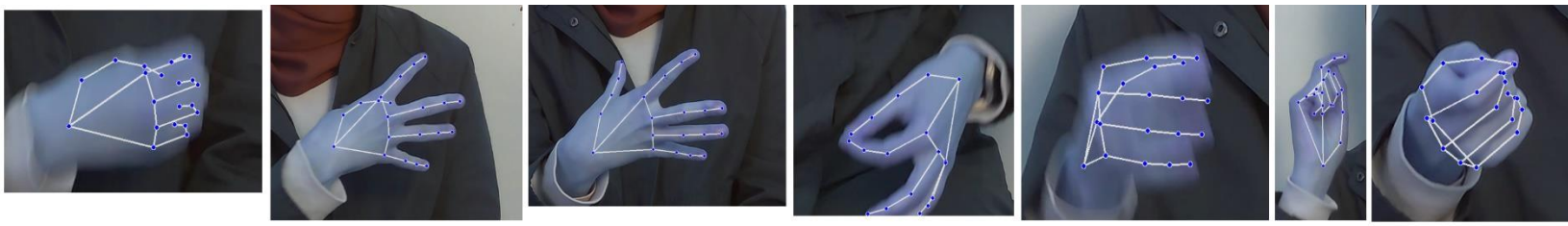


Figure III.10: sample of key-frames extraction using variant 04

A. Advantages:

— One major advantage of this technique is that it combines hand detection, cropping, and resizing, reducing the noise and irrelevant background information from the frames. It also extracts hand landmarks, which is a significant aspect for any sign language recognition task.

B. disadvantages:

— On the other hand, this technique is relatively time-consuming as it involves multiple preprocessing steps for each frame. Also, it relies on a fixed target number of frames which might not be optimal for all videos.

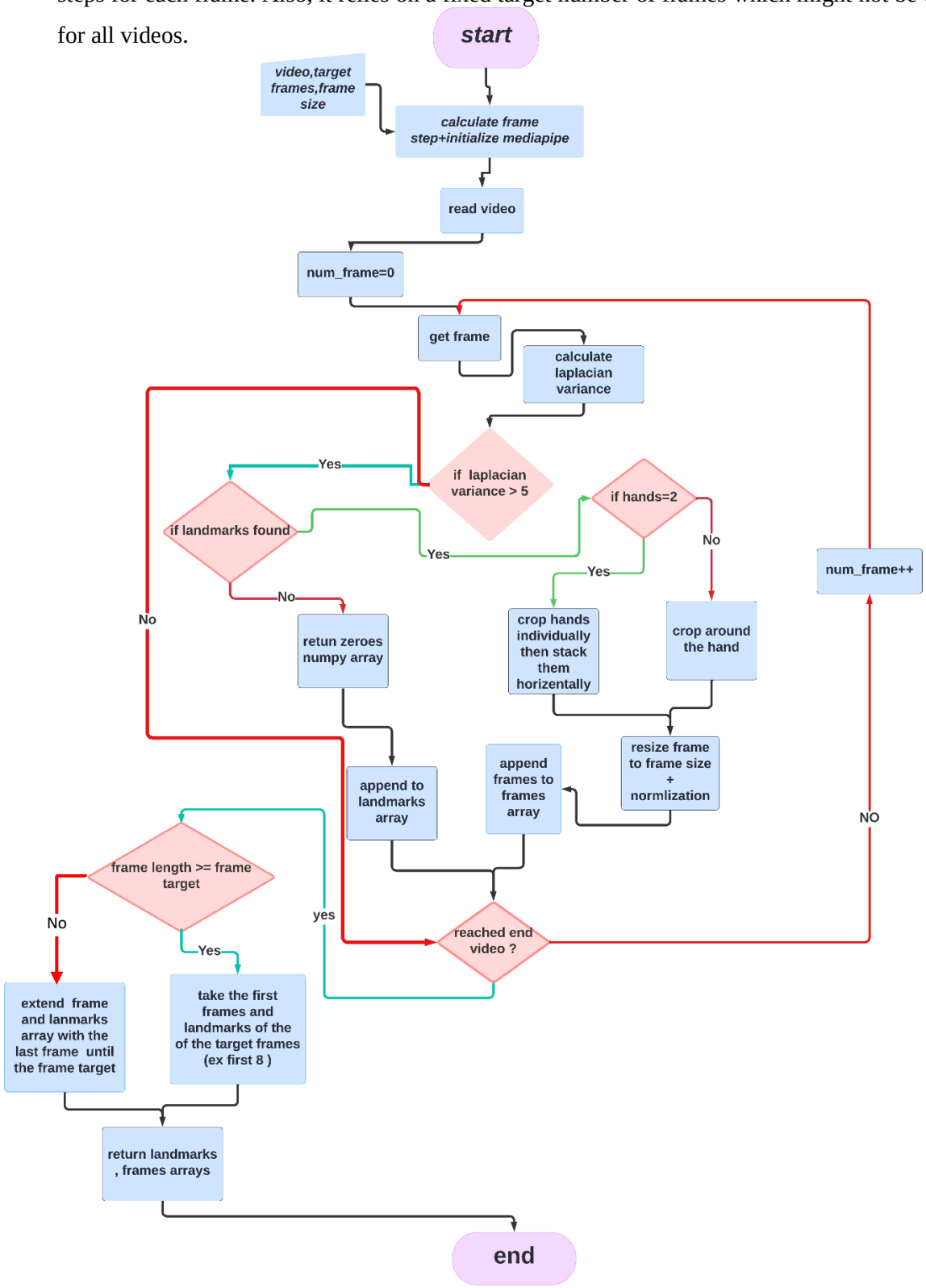


Figure III.11: Chart flow of Variant 04 for Key-frame Extraction

- **Variant 05:** This method represents a comprehensive approach to video preprocessing, enriching the initial preprocessing steps with additional techniques to derive more information from each frame. It incorporates depth map computation, optical flow calculation, and hand landmark extraction using MediaPipe's solution. These additional computations provide more context about the hand movements, such as relative depth of hand parts and motion direction and magnitude, which could lead to a richer representation of the sign language gestures. Here's a high-level overview of what it does:

1. **Loading the Video:** determine the total number of frames in the video. It then calculates which frames to extract based on a specified number of frames that should be evenly distributed across the video.
2. **Initializing MediaPipe Hands:** The method initializes the MediaPipe Hands object, which will be used to detect hand landmarks in each frame.
3. **Processing Selected Frames:** The method then enters a loop that processes each selected frame in the following way:
    - **Reading and Resizing the Frame:** The current frame is read from the video and resized to a specified size.
    - **Computing the Depth Map:** The depth map of the frame is computed, which provides information about the distance of objects in the frame.
    - **Computing Optical Flow:** The optical flow between the current frame and the previous frame is computed, which provides information about the motion of objects between the two frames. For the first frame, this is set to be an array of zeros.
    - **Extracting Hand Landmarks:** Hand landmarks are extracted from the frame using MediaPipe Hands. If no hands are detected, this is set to be an array of zeros.

4. **Releasing Resources and Returning Results:** Finally, the method releases the video capture and MediaPipe resources, and it returns the arrays of processed frames, depth maps, optical flows, and hand landmarks.

A. Advantages:

— This method offers several distinct advantages. Firstly, it calculates depth maps for frames, providing important spatial information about the position of the hands relative to the camera, adding a valuable dimension of data for models tasked with understanding hand gestures. Secondly, by calculating optical flow, the method captures the motion of the hands between consecutive frames, providing valuable context about hand movements, crucial for understanding dynamic sign language gestures. Furthermore, this method employs MediaPipe for robust hand landmark detection, even under challenging conditions. The method also normalizes the depth maps and optical flow maps to

a range between 0 and 1, making the data more suitable for many machine learning algorithms. Finally, it is highly flexible and adaptable due to its ability to process videos of any length and adjust to extract any number of frames, catering to different datasets and requirements.

B. disadvantages: While this method provides a richer representation of the video data, it does come with a few potential drawbacks

- Increased computational complexity: As the method involves additional steps such as depth map computation and optical flow calculation, it requires more computational resources and processing time. This might not be ideal for real-time applications or when processing a large number of videos.
- Dependency on lighting and texture: Both depth map computation and optical flow calculation depend on the texture and lighting of the video. Poor lighting or lack of texture can affect the accuracy of these calculations.
- Sensitivity to noise: Optical flow calculation is sensitive to noise in the video. Any abrupt changes in the video due to camera movement or flickering could affect the optical flow calculation.

For a clearer understanding of the concept of variant 05, please refer to Figure III.14



Figure III.12: sample of optical flow key-frames extraction using variant 05

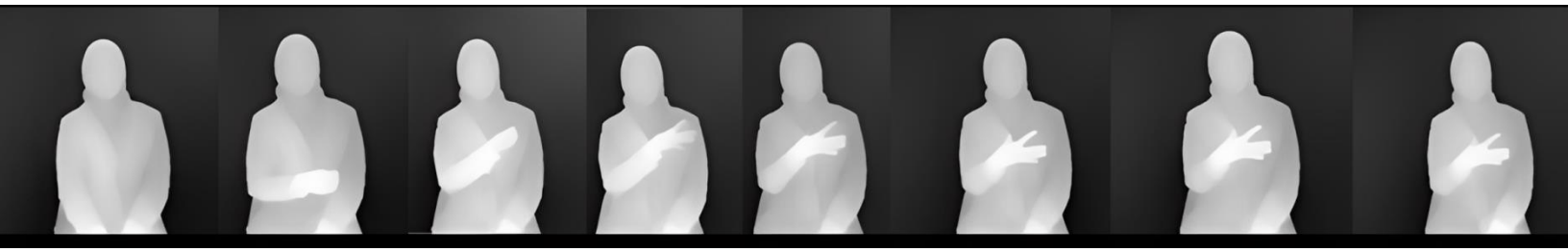


Figure III.13: sample of depth key-frames extraction using variant 05

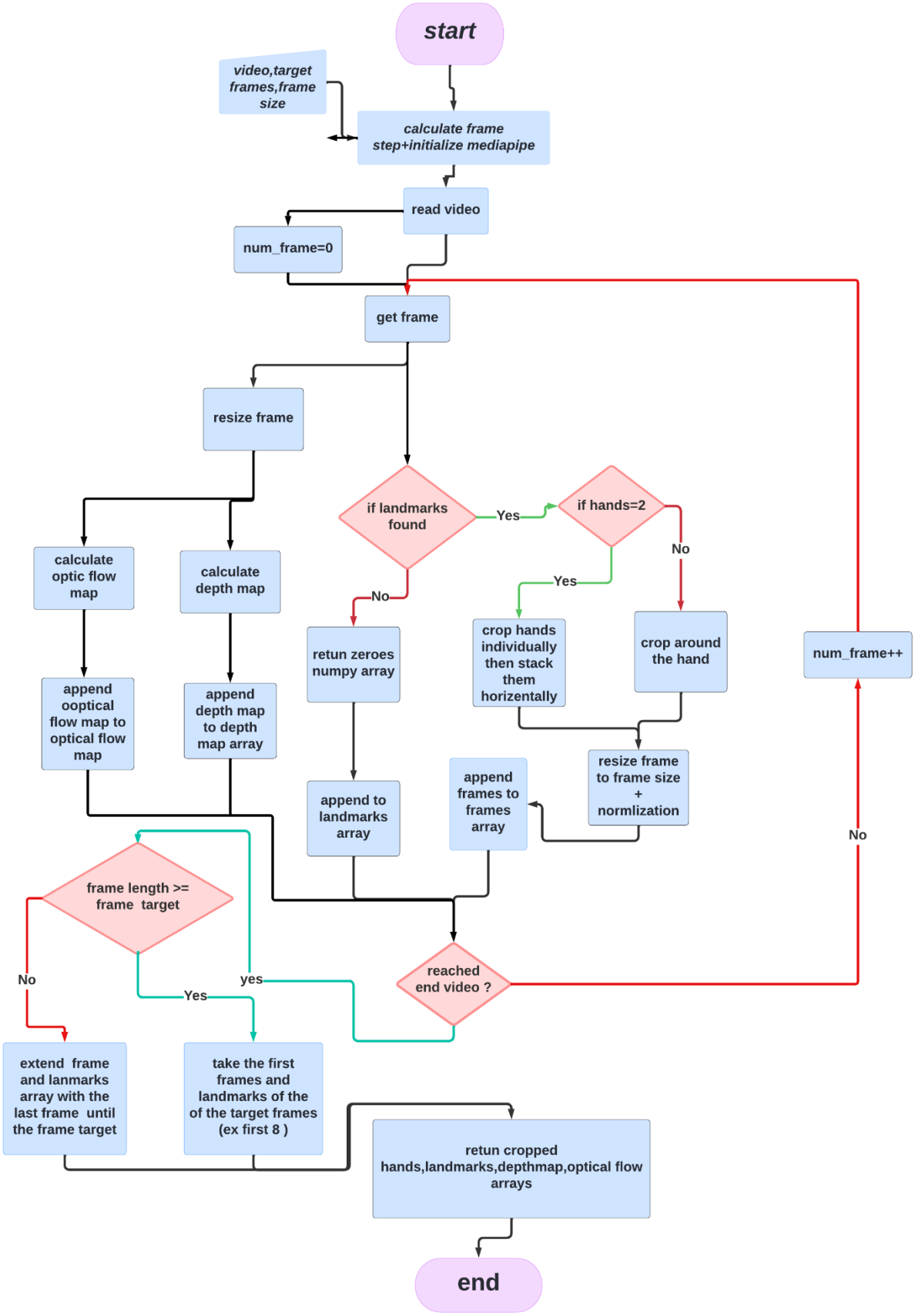


Figure III.14: Chart flow of Variant 05 for Key-frame Extraction

Table III.2 provides a summary of the five different keyframe extraction methods that were applied to the same one-second video. The aim of this comparison is to analyze the differences between the methods in terms of duration, number of frames processed, selected frames, and the type of output. The selected video has a duration of 1 seconds and 700 milliseconds, it consists of exactly 51 frames.

Table III.2 - Summary of variants of keyframe extraction applied on same video

Number of variants	**duration**	**number of frames processed**	**selected frames**	**type of output**
Variant 01	3.6 sec – 4 sec	9 Frames	8 Frames selected [0, 6, 9, 12, 18, 24, 30, 36, 42]	Frames
Variant 02	10 sec 13.2 sec	51 Frames	8 Frame selected [9, 12, 15, 20, 26, 36, 39, 44]	Frames with no background
Variant 03	8 Sec – 9 sec	51 Frames	51 Frames	hand landmarks
Variant 04	3.9 sec – 4.2 sec	9 Frames	8 Frames selected [0, 6, 9, 12, 18, 24, 30, 36, 42]	Cropped hand frames + hand landmarks
Variant 05	7 sec – 8.2 sec	9 Frames	8 Frames selected [0, 6, 9, 12, 18, 24, 30, 36, 42]	Frames + depth image +optical flow images + hand landmarks

III.5 The Data Augmentation.

This technique is primarily employed to artificially expand the diversity of the dataset without the need to collect additional data. Its primary objective is to enhance the robustness and invariance of the model to specific types of variations in the data, thereby improving the model's ability to generalize from the training dataset to unseen data.

Our approach to data augmentation involves the use of the ImageDataGenerator function, a part of the Keras library. The parameters for this function are set to conduct several transformations on the data, namely:

- **Rotation Range:** The images are subjected to random rotation within a range of -20 to +20 degrees. This adjustment aids the model in recognizing the signs irrespective of the angle at which they are viewed.
- **Width Shift and Height Shift:** These parameters allow the images to be translated along the x and y axes, respectively. This ensures that the model can identify the signs, regardless of their position within the image.
- **Horizontal Flip:** This transformation mirrors the image along the vertical axis. It adds variability to the data and aids the model in learning to recognize the signs, regardless of their orientation.

III.6 Model Development

For the development model phase, we present our proposed architecture that combines both convolutional neural networks (CNNs) and long-short-term memory (LSTM) networks along with an attention mechanism. This forms an autoencoder model for the task of sign language recognition. The autoencoder is a type of neural network used for learning efficient coding of input data (for more detailed explanation go back to II.8 Autoencoder). It's composed of an encoder, a bottleneck, and a decoder.

Encoder: The encoder part of our model consists of two branches. The first branch processes the video frames input using a series of convolutional layers, max-pooling layers, and dense layers. It applies a 3D convolutional operation in a time-distributed manner to capture temporal information. The second branch processes the hand landmarks input through a fully connected layer. Both branches extract relevant features from their respective inputs.

Bottleneck: The encoded features from the two branches are concatenated and passed through an attention mechanism, which selectively focuses on important parts of the combined feature representation. The attention mechanism enhances the model's ability to capture temporal dependencies and spatial relationships.

Decoder: The decoder part of our model utilizes an LSTM (Long Short-Term Memory) network to decode the encoded features. The LSTM layer takes the attention-weighted features as input and processes them in a recurrent manner, capturing sequential dependencies. The output of the LSTM layer is then fed into a series of dense layers with time-distributed connections. The final output is a sequence of probabilities representing the classification of sign language gestures.

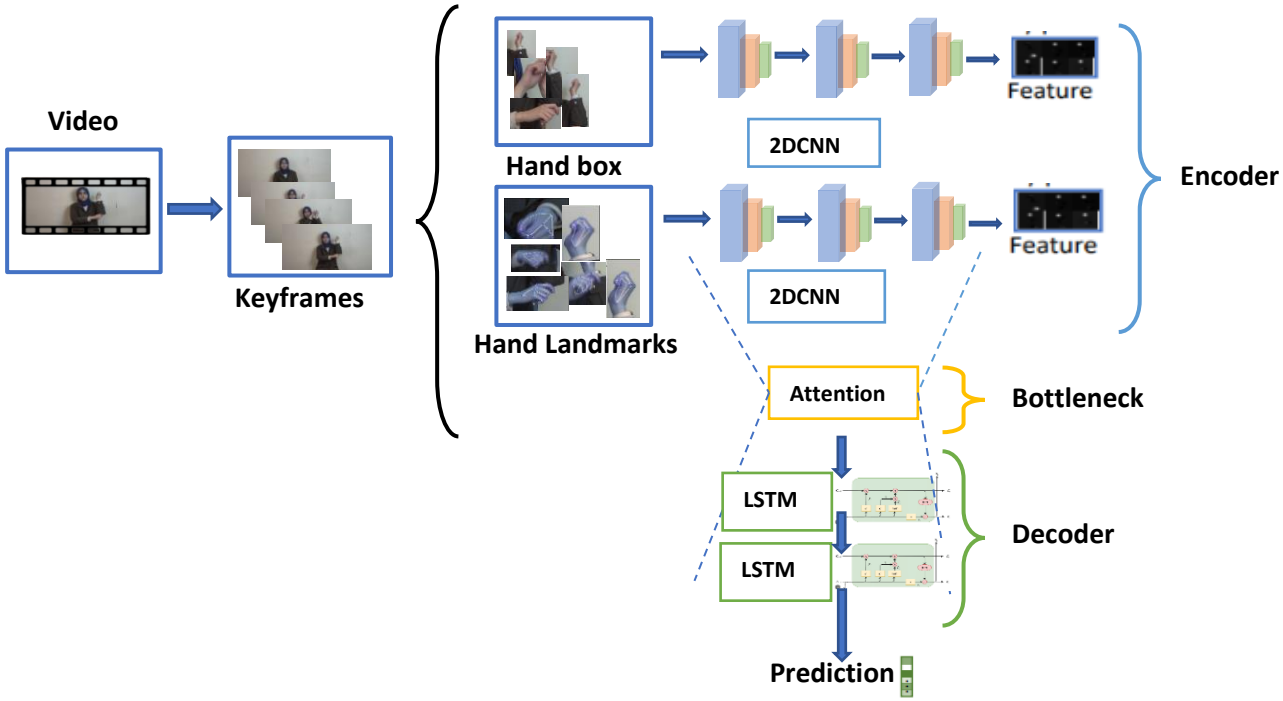


Figure III.15: Architecture of our auto encoder model

The model is compiled with the Adam optimizer and the mean squared error loss function, providing a balance between computational efficiency and model performance. It is this unique combination of spatial feature extraction (with the CNNs), temporal feature extraction (with the LSTM), and attention mechanism that allows our model to accurately recognize sign language from video sequences (more detailed summery of the model in the figure below Figure III.16).

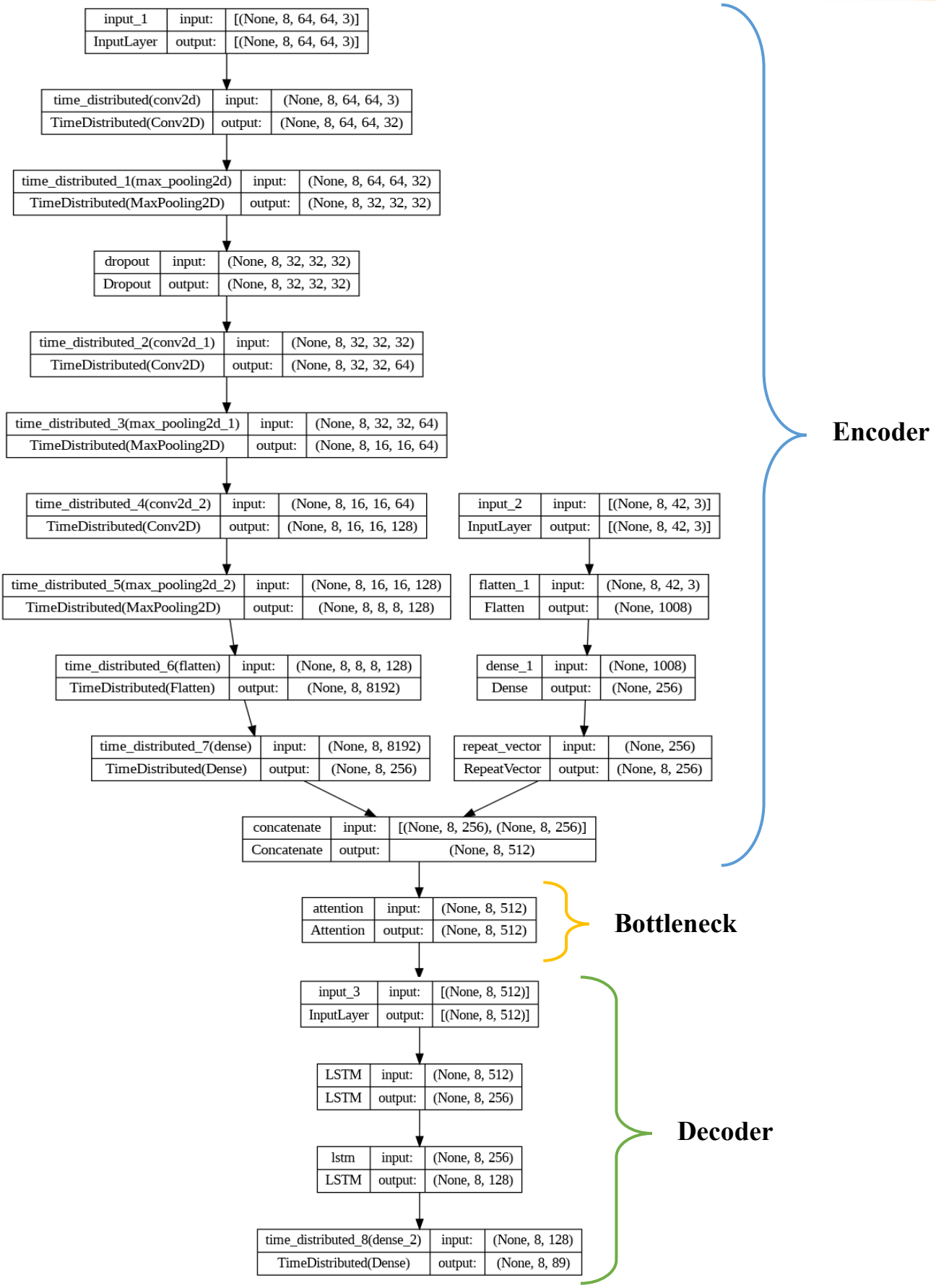


Figure III.16: detailed summery of developed model

III.7 Use case diagram

In our case, there is only one "user" actor that manipulates the graphical interface of the application "system". Table III.3 Explain the Use Case Scenario of our Sign Language Recognition System.

Usecase name	Sign Language Recognition
Participating Actors	User, System
Flow of Events	Start The System(U) Capturing Videos(S) | upload pre recoded video (U) Extract Frames(S) Translate Gesture(S) Recognizing Gestures(S) Display Result
Entry Condition	camera availability
Exit Condition	Pressing Exit button
Quality Requirements	Cam Pixels Clarity, Good Light Condition

Table III.3 - Use Case Scenario for Sign Language Recognition System

Figure III.17 is the use case diagram:

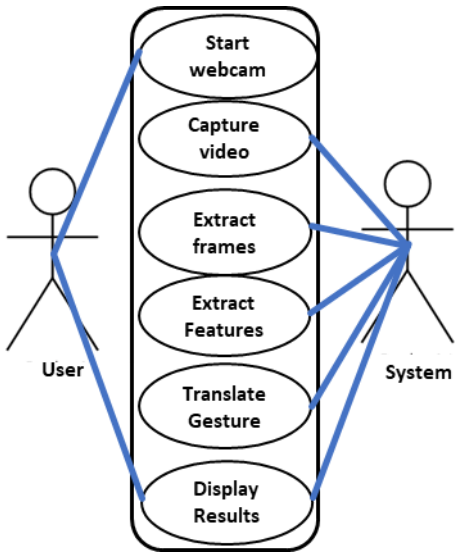


Figure III.17: Use Case Diagram Of our Sign Language Recognition System

III.8 Conclusion

To sum up, this chapter has presented an overview of our sign language classification system that utilizes the automatic encoder concept. We have provided a detailed description of the collected DataSet and discussed the variant of the preprocessing process we developed. The next chapter will focus on the implementation and evaluation of our system with the goal of identifying the most suitable solutions for addressing the problem of sign language classification. By utilizing the techniques presented in this chapter, we are confident that our system will make a significant contribution to improving the accessibility and inclusivity of technology for the deaf and hard of hearing communities.

Chapter IV

Implementation and performance testing

IV.1 Introduction

In this chapter, we begin by presenting the hardware and software development environment used to implement our system. We then interpret the results of the tests performed, comparing the performance of our model using both our own dataset and other publicly available datasets to other models from the literature. Finally, we conclude this chapter with a description of the proposed graphical user interface for the implementation of our system

IV.2 Development environment

To implement our solution, we require a set of hardware and software tools. This section presents an overview of these tools that been used.

IV.2.1 Hardware specifications

In order to implement our solution, we used:

- A desktop computer equipped with an Intel i5-3570K processor, operating at 3.40 GHz, 16 GB of RAM, a 256 GB SSD hard drive, and running the Windows 11 Pro operating system was utilized for pre-processing and model training purposes.
- Anther desktop computer been used for testing the other variants of key frame extraction, it equipped with i5-3570, 4.00 GHz and has 16.0 GB of RAM and running the Windows 10 as operating system
- Google Colaboratory: is a free and open-source platform introduced by Google, specifically designed for developers of data science applications, particularly machine learning applications.

IV.2.2 Software specifications and libraries

IV.2.2.1 Software

Visual Studio Code: also known as VS Code, is a source code editor developed by Microsoft for Windows, Linux, and MacOS. It includes various features such as debugging support, syntax highlighting, and intelligent code completion. Users can modify the theme, keyboard shortcuts, preferences, and install extensions that add extra features. [107]

Python: is a high-level, interpreted, multi-paradigm, and cross-platform programming language that was developed by Guido Van Rossum in the early 1980s. It is used for web development, AI, machine learning, operating systems, mobile application development, video games, and many other purposes [106]. **And the version of python that we worked is Python 3.10.0**

IV.2.2.2 libraries

NumPy: is an open-source project founded in 2005, based on the initial work of the Numerical and NumArray libraries. It enables numerical computation with Python, offering support for large, multidimensional arrays and matrices, as well as a wide range of high-level mathematical functions [86]. **our study was conducted with version numpy 1.24.2**

CV2 (Open-Source Computer Vision): also known as OpenCV-Python, is a library of programming functions mainly aimed at real-time computer vision. It is a free and open-source library that supports a wide range of image processing tasks such as object detection, image segmentation, and motion detection. The library is written in C++ and has interfaces for Python and other programming languages. [87]

Matplotlib: It is a library for creating static, animated, and interactive visualizations in Python. [88]

Keras: it is an open-source library written in Python. It offers consistent and simple APIs for interacting with neural network algorithms. [89]

Tensorflow: is a free and open-source library developed by Google for machine learning and artificial intelligence. It can be used for a variety of tasks, but it focuses specifically on training and inference of deep neural networks [90]. **For the version that we used it tensorflow2.11.0**

Mediapipe: it is an open-source cross-platform framework developed by Google for building multimodal machine learning pipelines, including tasks such as real-time video processing, hand and facial landmark detection, pose estimation, and object detection. [91] For more information refer to the annex chapter.

The following figure shows the logos of the specifications cited above:



Figure IV.1 – Logos of software specifications and libraries used.

IV.3 Implementation details and results

IV.3.1 preprocess

After evaluating and comparing the five variants discussed in Section III.3.1, and analyzing the preprocessing time taken by each variant as presented in Table III.2, it becomes evident that each variant has its own advantages and disadvantages. A notable observation is the trade-off between accuracy and real-time performance. It is observed that as the rapidity of response increases, the accuracy tends to decrease, and vice versa.

IV.3.1.1 Choice of preprocessing variant

Upon thorough evaluation of various preprocessing variants for recognizing Algerian sign language gestures, we have determined that Variant 4 best aligns with our system's objectives and requirements. Variant 4 showcases several advantages that make it the optimal choice for our project.

1. **Accuracy:** We found that Variant 4 showcases exceptional accuracy in capturing and representing LSA gestures. Through extensive testing, we achieved an impressive accuracy rate of an average accuracy of 99% across the dataset (see Table IV.5). This variant effectively captures intricate hand movements, finger configurations, facilitating precise recognition of gestures.
2. **Computational Efficiency:** Variant 4 has been optimized for computational efficiency, ensuring real-time processing on mobile devices. This is crucial for our mobile application, as it allows for smooth and responsive interaction with users with response time of 58 millisecond with computer camera and 3 sec for phone camera.
3. **Generalizability:** Variant 4 offers excellent generalizability by avoiding overfitting and effectively recognizing gestures across diverse individuals. It exhibits robustness to variations in camera quality and distance, ensuring accurate gesture recognition in challenging conditions.
4. **Compatibility:** We confirmed that Variant 4 is compatible with the computer vision and machine learning algorithms we plan to use in our system. This compatibility ensures seamless integration and efficient utilization of these algorithms for gesture recognition.
5. **Balancing Accuracy and Real-time Responsiveness:** Variant 4 strikes a balance between accuracy and real-time responsiveness. While it achieves high accuracy, it also maintains reasonable response times, making it suitable for practical real-time applications.

6. **The Overfitting:** refers to a phenomenon where a model becomes too specialized and performs well only on the specific individuals or conditions it was trained on, but fails to generalize to new or different instances.

IV.3.1.2 Comparison with other variants

To facilitate the comparative analysis, a comprehensive summary table (IV.1) is presented, outlining the essential attributes and performance metrics of each preprocessing variant. This table serves as a valuable resource for gaining profound insights into the distinctive advantages and limitations exhibited by each variant.

Table IV.1: Comparison of Preprocessing Variants for Recognizing Algerian Sign Language Gestures.

Prepressing Variant	selected	Accuracy	loss	overfitting	Real-time Responsiveness	Robustness to Camera Quality	Robustness to Distance
Variant 1		**99,8%**	**0.0001**	Very High	Moderate	Very Low	Moderate
Variant 2		96.25%	0.0003	High	Very Low	Very Low	Moderate
Variant 3		95%	0.00037	Low	High	High	High
Variant 4	✓	98,99%	0.00025	**Low**	**very High**	**Moderate**	**High**
Variant 5		99.1%	0.0002	Moderate	Low	Very Low	Moderate

In the context of recognizing Algerian sign language gestures, a high degree of overfitting indicates that the variant relies heavily on specific individuals' signing styles or other factors such as clothing appearance.

For instance, if a variant exhibits very high overfitting, it implies that the model can only accurately recognize gestures from individuals who were trained on with the same clothing appearance. If a person changes their clothing or if the model encounters individuals with different clothing styles, the model may fail to accurately recognize their gestures. This suggests that the extracted features in such a variant may not effectively focus on capturing the subtle nuances of gestures, as the majority of the image pixels may

be attributed to other factors like the face or clothing, rather than the actual hand movements and gestures themselves.

IV.3.1.3 Parameter tuning and adjustments of variant

After selecting Variant 4 as our preferred option, further experimentation was conducted to fine-tune its parameters. This involved exploring different configurations such as the number of frames extracted (4, 8, 16 frames), adjusting the size of the image, and varying the crop-scale of the extracted frames.

A. Number of frames

The number of frames selected is determined based on a technique that utilizes a variable called the frame step. The frame step is calculated by dividing the Total number of frames per video (TNF) by the number of selected frames. This method ensures that the selected frames are distinct from each other and effectively cover the entire video.

$$Frame\ step = \ TNF\ \boldsymbol{DIV}\ Number\ of\ extracted\ frame \qquad (4.1)$$

By using this frame extraction technique, we can guarantee that the chosen frames capture diverse moments throughout the video sequence (Figure IV.2), providing a comprehensive representation of the gestures performed. This leading to improved recognition accuracy and robustness in our system.

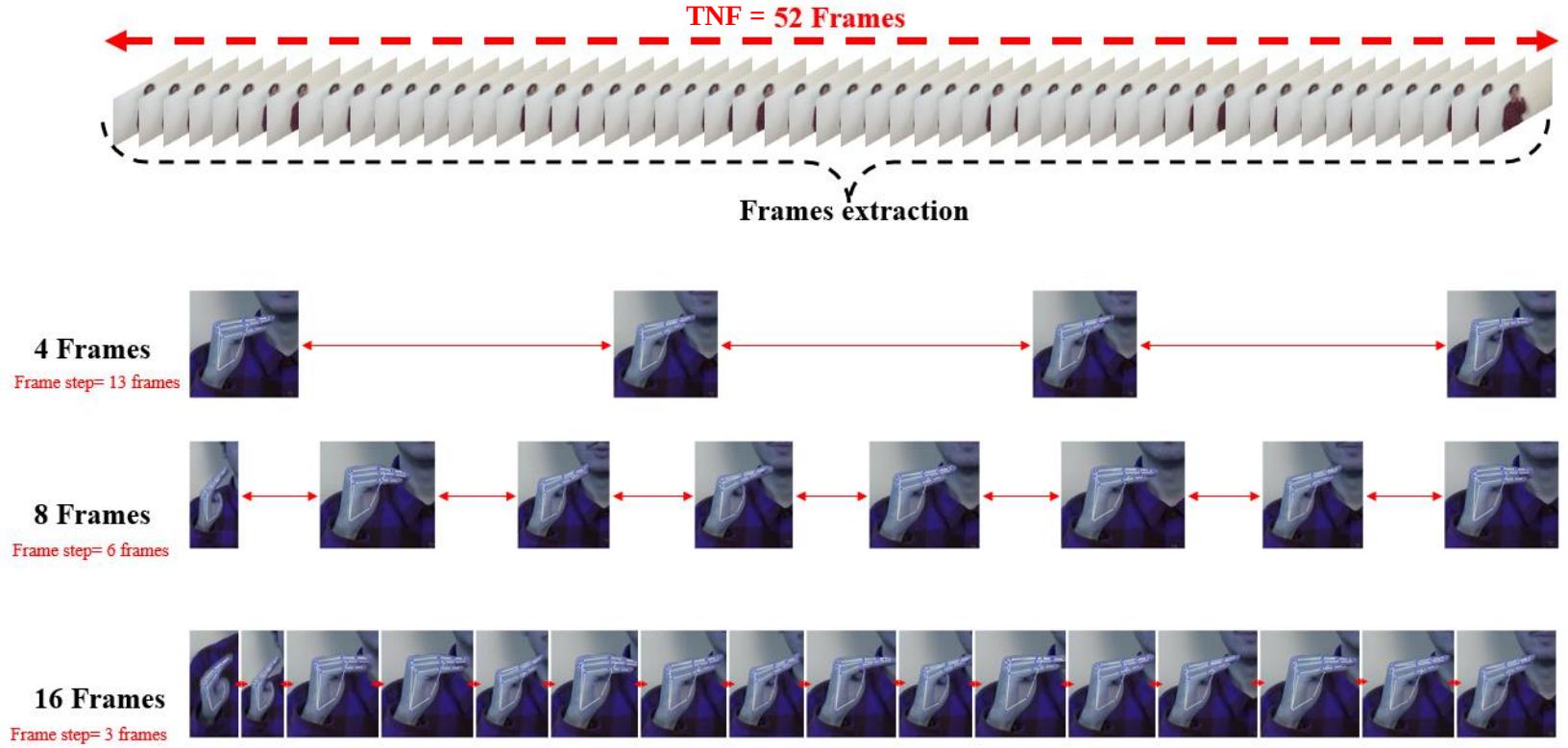


Figure IV.2: Comparison of Preprocessing Variants 4 with different number of frames.

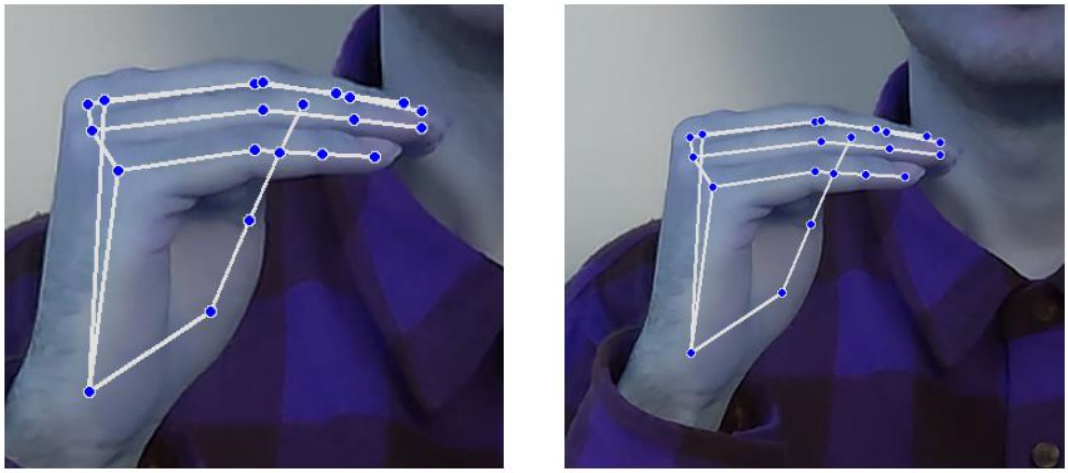
Table IV.2 provides valuable insights into the effects of parameter adjustments in frame step and the impact of it on the accuracy and time of preprocessing.

Table IV.2: Comparison of Preprocessing Variants 4 according to the frame step.

Number of extracted frames	se-lected	Frame step	Time of preprocess whole da-taset	Time of prepro-cess single video	Accuracy	Loss
4 Frames		7	**125min**	**1.68sec**	50.92%	0.0031
8 Frames	✔	3	274.41min	3.7sec	98,99%	0.00025
16 Frames		1	667min	8.99sec	**99.3%**	**0.00021**

B. Crop scale

Now we turn our attention to the second important parameter, namely Crop Scale. This parameter determines the size of the hand image that is extracted during the preprocessing stage. In Variant 4, we specifically extract cropped hand images. The choice of Crop Scale has a significant impact on the accuracy of our model. By adjusting the zoom level, we can control the level of detail captured in the hand images (Figure IV.3), thereby influencing the model's ability to accurately recognize gestures. Careful tuning of this parameter is crucial to strike the right balance between capturing sufficient hand details and avoiding excessive noise or loss of important information.



Crop scale = 1.5 Crop scale = 2

Figure IV.3: Comparison between two different crop scale on same frame with extraction of variant 4

A **Low Crop Scale value**, such as 1.5, indicates a larger zoom, resulting in more detailed hand images. On the other hand, a **Higher Crop Scale value**, such as 2, implies a lesser zoom, capturing a broader view of the hand. The choice of Crop Scale is crucial as it influences the level of detail and context available to the model during gesture recognition. Now, let's examine the performance of Variant 4 with different Crop Scale values in the following table:

Table IV.3: Comparison between two different crop scale for variant 4

Crop scale	selected	Accuracy	Loss
Crop scale = 1.5	✔	98,99%	**0.00025**
Crop scale = 2		97,61%	0.00034

Table IV.3 provides an overview of the performance metrics for each Crop Scale value. We can observe that a higher Crop Scale value of 1.5 yields a slightly higher accuracy compared to a Crop Scale value of 2. This suggests that a larger zoom level contributes to improved gesture recognition performance by capturing more intricate hand details

C. Image size

The Image Size parameter plays a crucial role in determining the dimensions of the hand images extracted for gesture recognition. In our study, we experimented with three different image sizes: **64x64x3**, **112x112x3**, and **128x128x3**. However, due to time constraints and the long preprocessing time associated with larger image sizes, we focused our analysis on these three specific sizes.

Among the tested sizes, we found that the **64x64x3** dimensions were the most suitable for our system. The larger sizes, such as **112x112x3** and **128x128x3**, proved to be impractical due to their extensive memory requirements and frequent crashing issues during the preprocessing phase. Consequently, we opted for the smaller image size to ensure smoother processing and reduce computational burdens.

While it is important to consider larger image sizes for capturing more detailed hand information, practical limitations, such as memory constraints and processing time, need to be taken into account. The chosen **64x64x3** dimensions strike a balance between computational feasibility and maintaining a sufficient level of detail for accurate gesture recognition.

D. background

Another parameter that we experimented with was background removal. The idea behind this parameter was to eliminate the background from the captured images, focusing solely on the hand region. However, we encountered several challenges with this approach.

One issue we encountered was that the background removal process sometimes resulted in the entire hand being removed along with the background. This led to a loss of important hand features necessary for accurate gesture recognition. Additionally, even when the background removal was successful in isolating the hand region, it significantly increased the processing time. On average, the background removal process added an additional 3 to 4 seconds per image, which was not feasible for real-time applications (Figure IV.4).

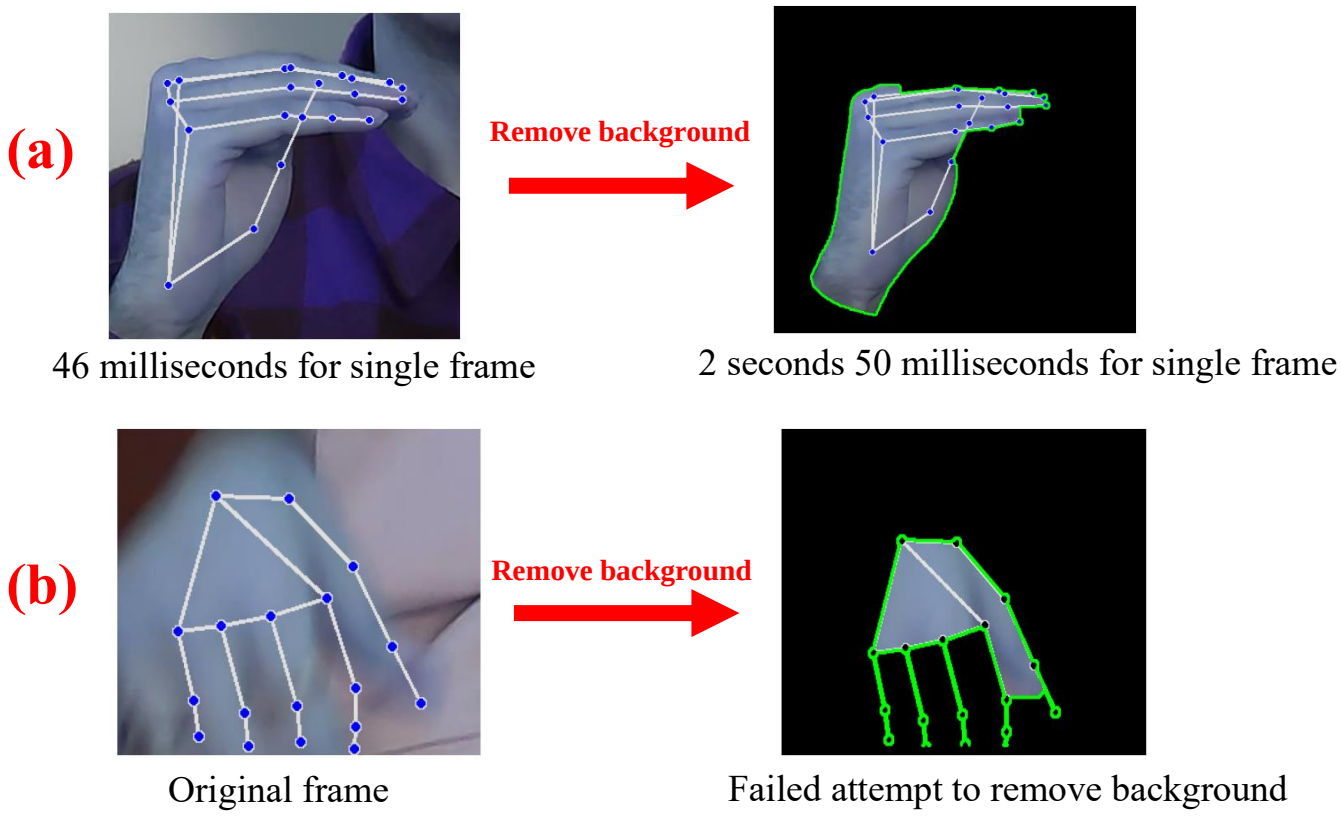


Figure IV.4: Variant 4 with (a) Successful Background Remover and taken time for single frame extraction (b) Failed attempt to remove background.

Considering the limitations and drawbacks of the background removal parameter, we made the decision to exclude it from our final variant. While the removal of background can potentially enhance the accuracy of gesture recognition, the trade-off in terms of time and loss of hand features outweighed the benefits in our specific context.

IV.3.2 The model

Within this context, our investigation focuses on the careful selection and significance of key parameters that profoundly influence the model's performance. By meticulously fine-tuning and optimizing these parameters, we aim to enhance the accuracy and robustness of the model in gesture recognition. Subsequently, we provide a comprehensive overview of the parameter selection process, emphasizing its profound implications for the overall effectiveness of the model.

IV.3.2.1 The choice of parameters

Table IV.4 provides an overview of the model's important parameters including the specific values assigned to each parameter, along with a brief description for the reasoning behind their selection.

Table IV.4: Model Parameters and their Corresponding Values with their descriptions

Parameters	**Values**	**description**
Input Shape 1	8x64x64x3	The size of the cropped hand image
Input Shape 2	8x42x3	Hands landmarks array shape
Optimization Function	Adam	The most commonly used optimizer in the literature
Batch Size	8	After several experiments, we found that smaller batch sizes yield better results
Number of Iterations	25	The maximum number of iterations we were able to perform during testing without encountering any limitations
Output Size	8x89	The output size of the model is an array of 8 elements, where each element represents a class from a total of 89 possible classes.

IV.4. Results

In this section, we present the results of our experiments on recognizing Algerian sign language gestures using Variant 4 with the SIGAL dataset and the chosen model. We begin by analyzing the classification performance for each class, providing insights into the accuracy and effectiveness of the model in accurately recognizing different gestures.

To gain a better understanding of the model's performance, we first examine the training and validation accuracy. Figure IV.5 illustrates the training and validation accuracy curves, which demonstrate the model's ability to learn and generalize from the training data while maintaining accuracy on unseen validation data.

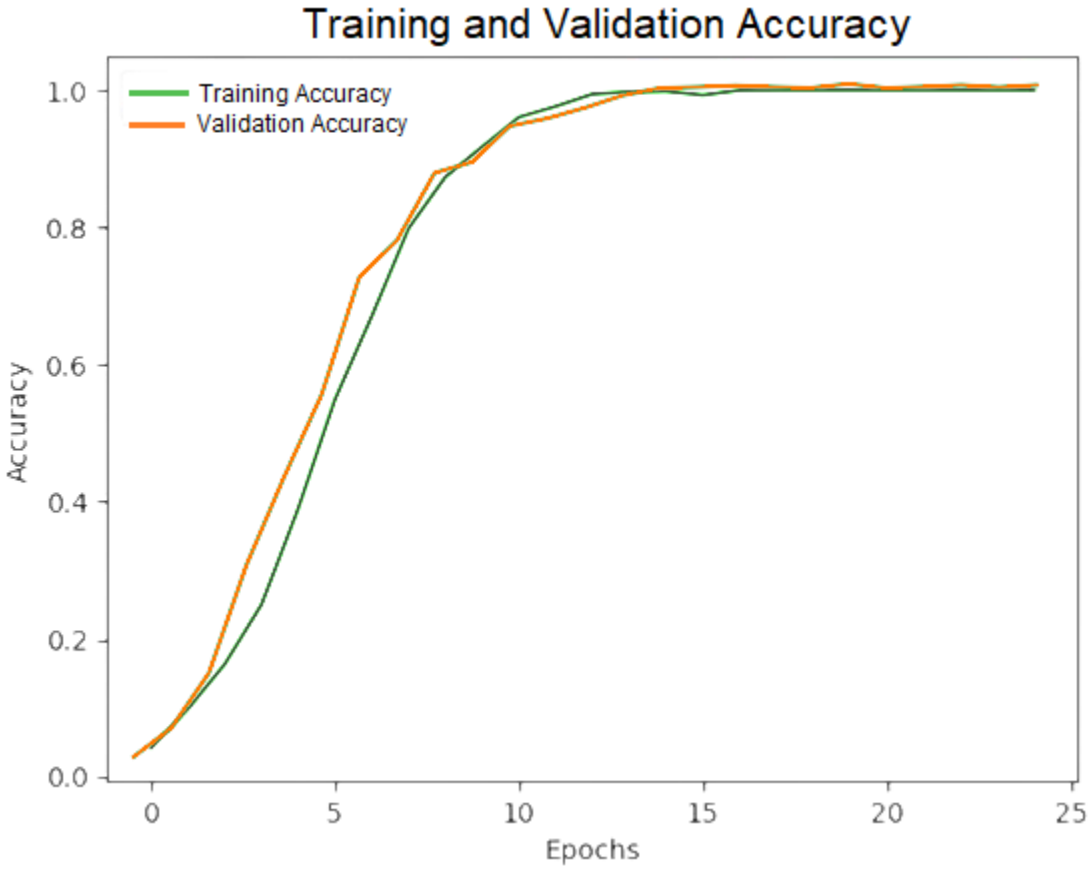


Figure IV.5: Training and Validation Accuracy on SIGAL dataset with our model and variant 4

The training and validation accuracy curves demonstrate a significant increase in accuracy during the initial epochs, particularly up to epoch 15. However, after reaching this point, the accuracy seems to stabilize and remain consistently high, with a value of approximately 1.0. This indicates that the model has successfully learned and generalized the patterns and features necessary for accurately recognizing Algerian sign language gestures.

Next, we explore the training and validation loss in Figure IV.6. The loss curves provide valuable insights into the convergence and stability of the model during the training process. Lower loss values indicate better optimization and improved generalization capabilities of the model.

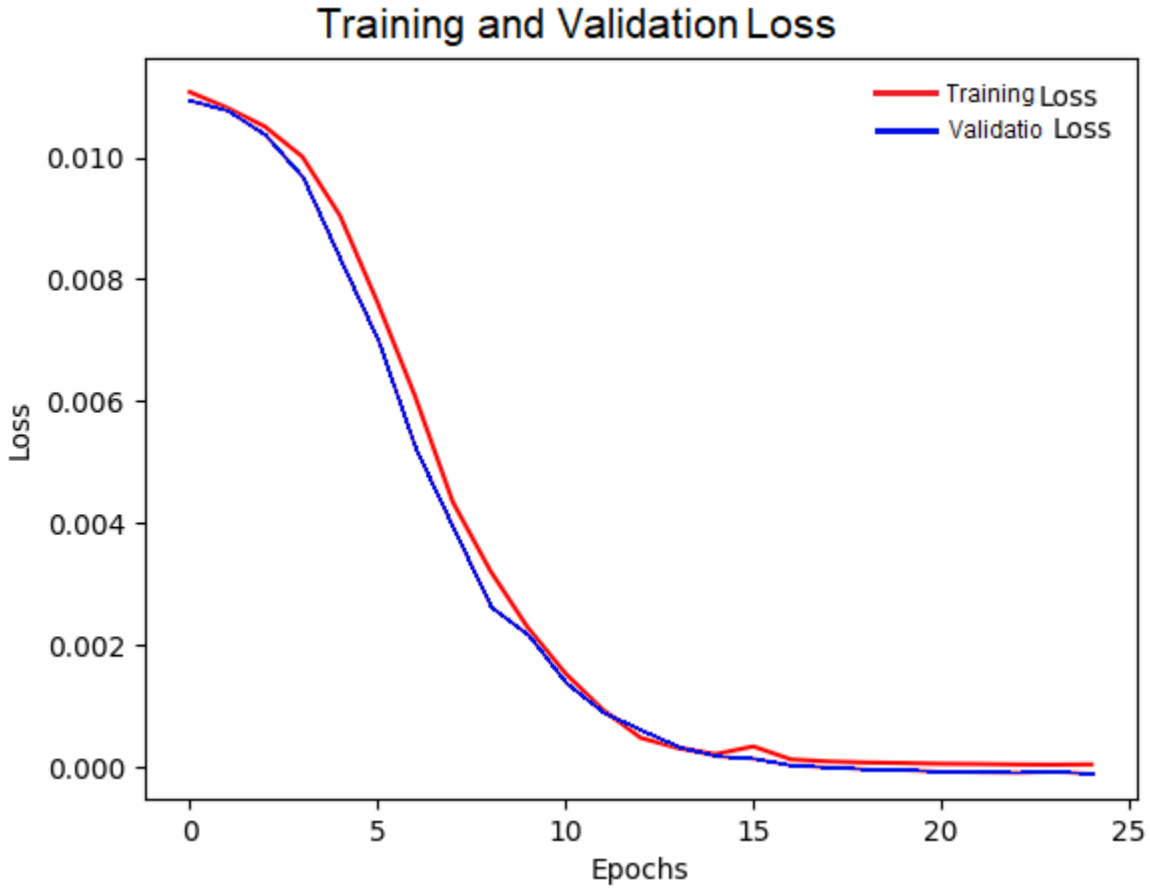


Figure IV.5: Training and Validation Loss on SIGAL dataset with our model and variant 4

The training and validation loss curves demonstrate the progression of the loss function during the training process. We observe a significant decrease in the loss values during the initial epochs, starting from 0.010 and reaching 0.00025 at epoch 17. This indicates that the model is effectively learning and minimizing the discrepancy between predicted and actual outputs. Subsequently, the loss values stabilize, suggesting that the model has reached a point of convergence and is consistently performing well in terms of minimizing errors.

By analyzing both the training and validation accuracy as well as the training and validation loss, we can assess the overall performance and generalization ability of the model. These graphs serve as visual representations of the model's learning progress and provide a basis for further analysis of its performance.

IV.4.1 classifications report

Table IV.5 presents the classifications report, which includes detailed metrics such as precision, recall, F1 score, and support for each class. These metrics offer valuable insights into the accuracy and effectiveness of the model in correctly identifying different gestures. By examining the precision, recall, and F1 score for each class, we can assess the model's performance in terms of both overall accuracy and its ability to correctly classify specific gestures.

Table IV.5: Classifications Report

class number	precision	recall	f1-score	support
0	1.00	1.00	1.00	6
1	0.91	1.00	0.95	10
2	1.00	0.90	0.95	10
3	0.92	1.00	0.96	11
4	1.00	1.00	1.00	6
5	1.00	1.00	1.00	13
6	1.00	1.00	1.00	12
7	1.00	1.00	1.00	11
8	1.00	1.00	1.00	12
9	1.00	1.00	1.00	9
10	1.00	1.00	1.00	9
11	1.00	1.00	1.00	9
12	1.00	1.00	1.00	8
13	0.83	1.00	0.91	5
14	1.00	0.92	0.96	13
15	1.00	1.00	1.00	12
16	1.00	1.00	1.00	14
17	1.00	1.00	1.00	9
18	1.00	1.00	1.00	6
19	1.00	1.00	1.00	4
20	0.91	1.00	0.95	10
21	1.00	1.00	1.00	12
22	1.00	1.00	1.00	13
23	1.00	1.00	1.00	12
24	1.00	1.00	1.00	11
25	1.00	1.00	1.00	14
26	1.00	1.00	1.00	7
27	1.00	1.00	1.00	6
28	1.00	0.90	0.95	10
29	1.00	1.00	1.00	12
30	1.00	1.00	1.00	8
31	1.00	1.00	1.00	6
32	1.00	1.00	1.00	3
33	1.00	1.00	1.00	11
34	1.00	1.00	1.00	16
35	1.00	1.00	1.00	12
36	1.00	1.00	1.00	7

37	1.00	1.00	1.00	12
38	1.00	1.00	1.00	10
39	1.00	0.89	0.94	9
40	1.00	1.00	1.00	12
41	1.00	0.88	0.93	8
42	1.00	1.00	1.00	11
43	1.00	1.00	1.00	10
44	1.00	1.00	1.00	7
45	1.00	1.00	1.00	17
46	1.00	1.00	1.00	7
47	1.00	1.00	1.00	11
48	1.00	1.00	1.00	8
49	1.00	1.00	1.00	9
50	1.00	1.00	1.00	10
51	1.00	1.00	1.00	9
52	1.00	1.00	1.00	9
53	0.89	1.00	0.94	8
54	1.00	1.00	1.00	11
55	1.00	1.00	1.00	11
56	1.00	0.91	0.95	11
57	1.00	1.00	1.00	14
58	1.00	1.00	1.00	11
59	1.00	1.00	1.00	13
60	1.00	1.00	1.00	13
61	1.00	1.00	1.00	5
62	0.92	1.00	0.96	12
63	1.00	1.00	1.00	9
64	1.00	0.77	0.87	13
65	1.00	1.00	1.00	13
66	1.00	1.00	1.00	8
67	1.00	1.00	1.00	18
68	1.00	1.00	1.00	12
69	1.00	1.00	1.00	9
70	1.00	1.00	1.00	10
71	1.00	1.00	1.00	8
72	1.00	1.00	1.00	6
73	1.00	1.00	1.00	9
74	1.00	1.00	1.00	10
75	1.00	1.00	1.00	8
76	1.00	1.00	1.00	9
77	1.00	1.00	1.00	12
78	1.00	1.00	1.00	10
79	0.90	1.00	0.95	9
80	1.00	0.90	0.95	10
81	1.00	1.00	1.00	6
82	1.00	1.00	1.00	7
83	1.00	1.00	1.00	13
84	1.00	1.00	1.00	6
85	1.00	1.00	1.00	16
86	0.80	1.00	0.89	12
87	1.00	1.00	1.00	10

88	1.00	1.00	1.00	10
accuracy			**0.99**	**890**
macro avg	**0.99**	**0.99**	**0.99**	**890**
weighted avg	**0.99**	**0.99**	**0.99**	**890**

The precision metric represents the proportion of correctly predicted positive instances out of the total predicted positive instances. **Recall**, also known as sensitivity, measures the proportion of correctly predicted positive instances out of the actual positive instances. **The F1**-score is the harmonic mean of precision and recall, providing a balanced measure of the model's accuracy. **Support** indicates the number of samples in the dataset for each class, giving insights into the dataset's distribution.

Analyzing the precision, recall, and F1-score for each class allows us to evaluate the model's overall accuracy and its ability to correctly classify specific gestures. The support values provide additional context by considering the sample sizes. It is worth noting that the model achieves an overall accuracy of 99%, indicating its strong performance in recognizing Algerian sign language gestures.

The 'accuracy' row in the table represents the accuracy of the model across all classes, while the 'macro avg' and 'weighted avg' rows provide the average values of precision, recall, and F1-score, considering either an equal weight or a weighted weight based on the number of samples.

IV.4.2 Confusion Matrix

Our model demonstrated strong classification performance, accurately predicting the majority of instances in our dataset. Out of a total of 890, the model only 7.

The confusion matrix (Figure IV.6) provides a detailed breakdown of the misclassifications, highlighting specific patterns and areas of improvement. Upon analysis, it is evident that the misclassifications primarily stem from several factors, including:

- **High Similarity Between Signs:** Many signs in the Algerian Sign Language exhibit significant visual resemblance posing challenges for accurate differentiation.
- **Imbalance in the Test Dataset:** The test dataset used for evaluation may have an imbalance in the distribution of samples across different classes.
- **Limited Size of the Dataset:** The dataset used for training and testing the model may be relatively small, which can limit the model's exposure to diverse instances and hinder its ability to capture the full variability of the sign language gestures.

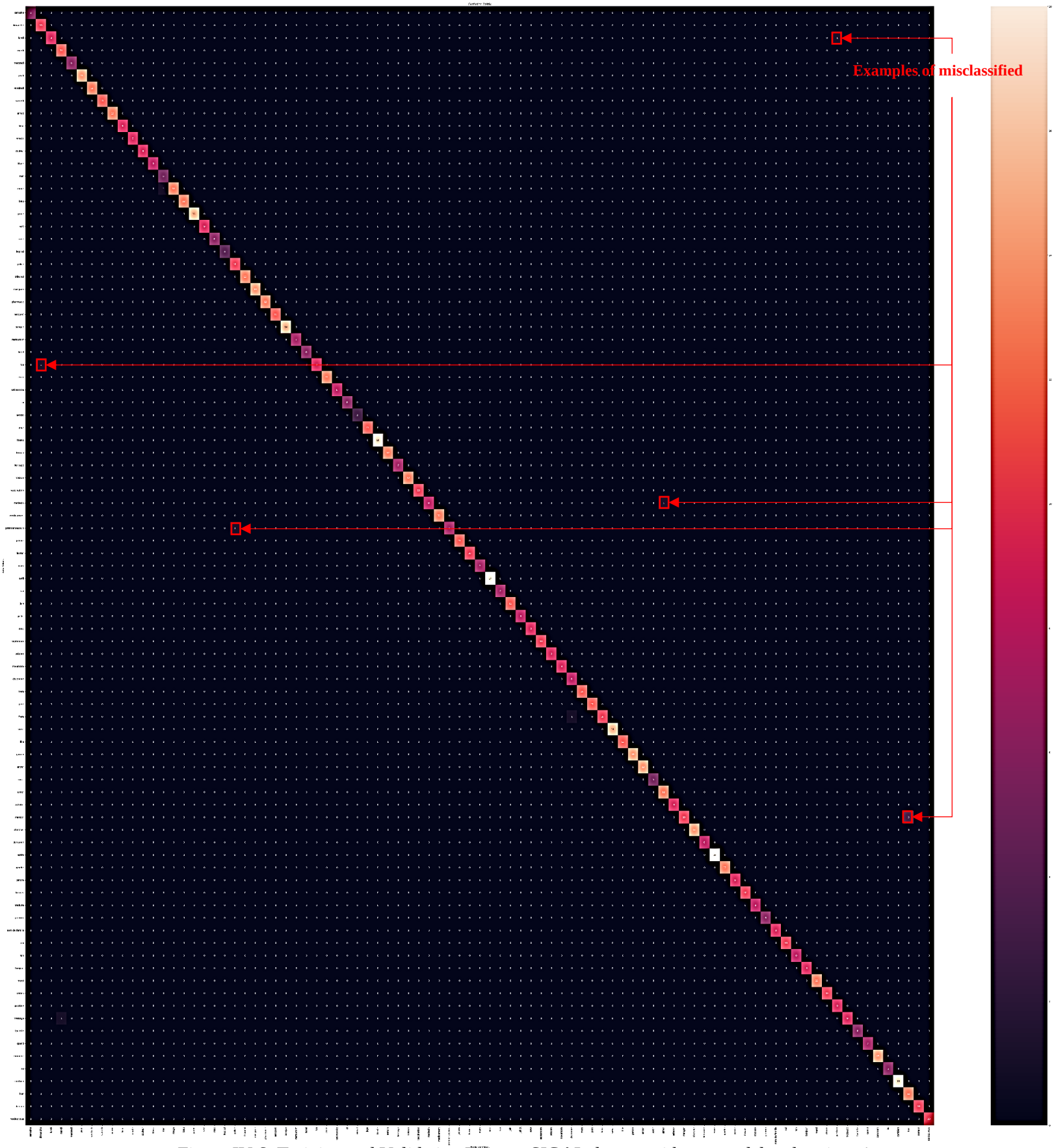


Figure IV.6: Training and Validation Loss on SIGAL dataset with our model and variant 4

Due to the large size of the original confusion matrix (Figure IV.6), a zoomed-in version is presented in Figure IV.7 focusing on 15 selected classes out of the total 89 classes to show on the cases of misclassified gestures. The misclassified gestures are highlighted with red frames, allowing for a clearer visualization of the classification errors.

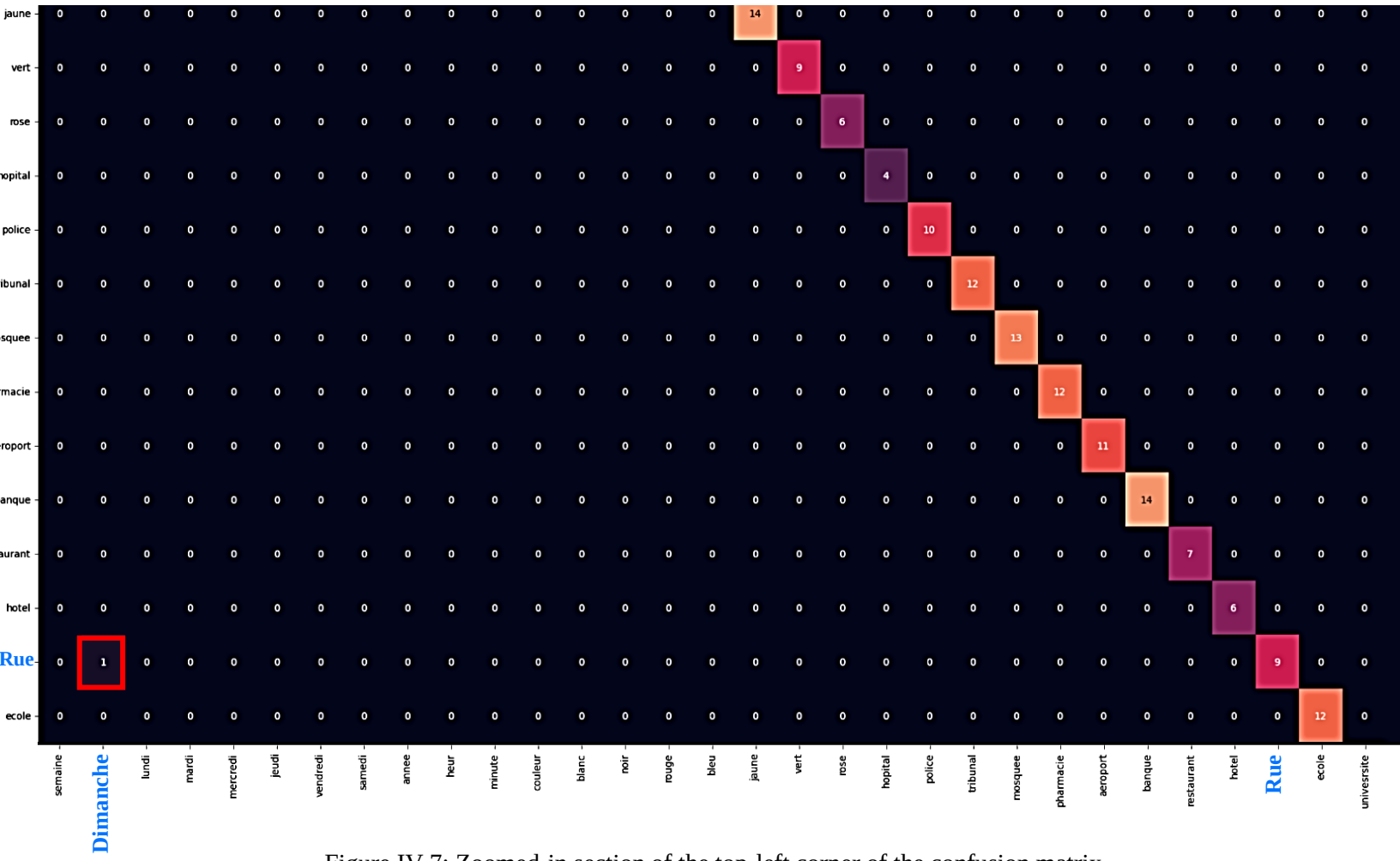


Figure IV.7: Zoomed-in section of the top-left corner of the confusion matrix

In this particular scenario, we observe that out of the instances representing the word "rue," nine of them have been correctly classified as "Rue." However, there is one instance that has been mistakenly classified as "Dimanche " when it should have been classified as " Rue". This misclassification is highlighted with red borders in the zoomed-in version of the confusion matrix. It is worth noting that in this matrix, the x-axis (horizontal) represents the predicted classes, while the y-axis (vertical) represents the actual classes.

IV.4.3 Test Models on SIGAL Dataset

In this section, we present the training results of different models on the SIGAL dataset, which was specifically created for this project. The models were trained using a subset of the dataset, and the reported metrics are based on the **training data** refer to Table IV.6 for mor details.

Table IV.6: Performance Comparison of Different Models on SIGAL Dataset

Model	Average time per iteration	Total model training time	Accuracy	Loss	Validation Accuracy	Validation Loss
ConvLSTM	52sec	22min	80.48%	0.6930	73.48%	0.9395
Autoencoder	160sec	67min	**100.00%**	**0.00005**	**98,99%**	**0.0015**
LSTM	**22sec**	**10min**	92.28%	0.2259	60.56%	2.3433
2DCNN-LSTM	55sec	23min	71.48%	0.9300	65.00%	0.9395

These results indicate the performance of the models during the training phase. The reported metrics include the loss and accuracy values. It is important to note that these metrics are based on the training data and may not necessarily reflect the models' performance on unseen test data.

IV.4.4 Comparative analysis

IV.4.4.1 different datasets on our chosen model

In this section, we conducted tests on different datasets consisting of various languages to evaluate the performance of our chosen model in recognizing sign language gestures across different linguistic contexts., we were able to assess its robustness and generalization capabilities. The test datasets included samples from different sign languages, representing the diversity and complexity of gestures across different

linguistic communities. By evaluating our chosen model on these datasets, we aimed to assess its ability to accurately classify and understand sign language gestures beyond a single language or dialect.

Table IV.7: Performance Comparison of Different Datasets with our model only

Dataset name	Accuracy	Loss	Validation Accuracy	Validation Loss	Training Time
LSA64 [111]	95,10%	0.00076	90.06%	0.0019	38.70 min
ASl_vd20 [112]	100%	0.00019	62.26%	0.0282	12 min
SIGAL	**100%**	**0.00005**	**98,99%**	**0.0015**	**80min**

Table IV.7, we present the comparative performance of different datasets on the training data, using 30 iterations. It is important to note that these results are based on the training phase of the models and may not necessarily reflect their performance on unseen test data. The accuracy, loss, validation accuracy, and validation loss metrics provide insights into the models' performance during training. The training time column indicates the time taken to train the models on each dataset. It is worth mentioning that these results are specific to the training data and the number of iterations used in this analysis

IV.4.4.2 LSA64 comparison

LSA64 is a dataset specifically curated for Argentinian Sign Language (LSA), comprising 3,200 videos of 10 non-expert subjects performing 64 different sign types. The dataset covers a wide range of commonly used signs in LSA, including verbs and nouns. The recordings were conducted in both outdoor and indoor environments, providing variation in lighting conditions. Subjects wore fluorescent gloves to facilitate hand segmentation in the images. LSA64 serves as a valuable resource for research and development in automatic sign recognition and understanding, contributing to advancements in sign language technology. [102]

Table IV.8, we present a comparison of various classifiers and their corresponding features, accuracy, and publication years. While our autoencoder model achieved an accuracy of 98.67%, it is important to note that direct comparison with other models may not be straightforward. Other classifiers in the table might have achieved higher accuracies, but they could have faced challenges such as overfitting or required longer processing times to provide results.

Table IV.8: Performance Comparison of LSA64 dataset

Classifier	Feature	Accuracy	Year	Reference
HMM+GMM	Radon Transform + Hand segmentation	95.95%	2016	[92]
Custom Bayesian	Radon Transform + Hand segmentation	97%	2016	[93]
SVM	Cumulative Shape Difference	85%	2018	[94]
LSTM	Skeletal + LDS Histograms	98.09%	2018	[95]
LSTM	Skeletal + RGB + Optical Flow	97.94%	2018	[96]
LSTM + DSC	**Skeletal + RGB + Optical Flow**	**99.84%**	2018	[97]
LSTM	RGB + Optical Flow	99.37%	2018	[98]
LSTM	Inception CNN + Hand segmentation	95.2%	2018	[99]
Bi-directional LSTM	Inception CNN	96%	2018	[100]
LSTM + MEMP 3D CNN	3D CNN	99.06%	2019	[101]
Kernel Extreme Learning Machines	MobileNet CNN + Skeletal + Motion Template Fusion	97.81%	2020	[102]
LSTM	3D CNN	98.5%	2021	[103]
autoencoder	RGB Cropped hand frames + hand landmarks	98.67%	**2023**	Our work

IV.4.4.3 ASl_vd20 comparison

ASL_vd20 is a subset of the ASLLVD-Skeleton database, comprising 20 selected sign gestures from American Sign Language (ASL). The dataset focuses on capturing skeletal data of sign language movements and provides a valuable resource for research and development in ASL recognition and understanding. Each sign gesture is represented by a small video segment, allowing for targeted analysis and exploration of ASL gestures within a concise subset. [108]

Table IV.9 presents a concise comparison of classifiers and their associated features. Notably, the auto-encoder-based approach achieved an accuracy of 66.36%. Interestingly, all of the classifiers in the table demonstrate accuracy percentages below 70%. This lower accuracy is likely due to the limited size of the dataset, which may not include a sufficient number of videos for robust training and evaluation purposes.

Table IV.9: Performance Comparison of ASl_vd20

Classifier	Feature	Accuracy	Year	Reference
ST-GCN	skeleton + key-points	61.04%	2019	[104]
Basic 3DGCN	skeleton + hand landmarks	62.5	2022	[105]
Enhanced 3DGCN	skeleton + hand landmarks	**68.75**	2022	[105]
autoencoder	RGB Cropped hand frames + hand landmarks	66.36%	**2023**	Our work

IV.6 Discussion

The performance of a classifier involves a **trade-off** between accuracy and other factors such as **generalization, robustness, and real-time responsiveness**. Our autoencoder model strikes a balance in these aspects, offering a reasonable accuracy while being robust and responsive. It allows for efficient communication and interpretation of sign language gestures in real-world scenarios, making it suitable for practical applications.

Therefore, although our model may not have achieved the highest accuracy in these comparisons, it demonstrates competitive performance while addressing important considerations for real-time sign language recognition systems.

IV.7 The graphical interface

SignAtlas graphical interface developed for the computer software includes various features to enhance user experience and functionality. The interface consists of a home page that provides a brief description of the software along with our logo for visual identification. Additionally, there is a button labeled "Webcam" that directs users to another interface specifically designed for recording gestures. In this interface, users can start recording gestures, and the results are displayed in both written and sound formats.

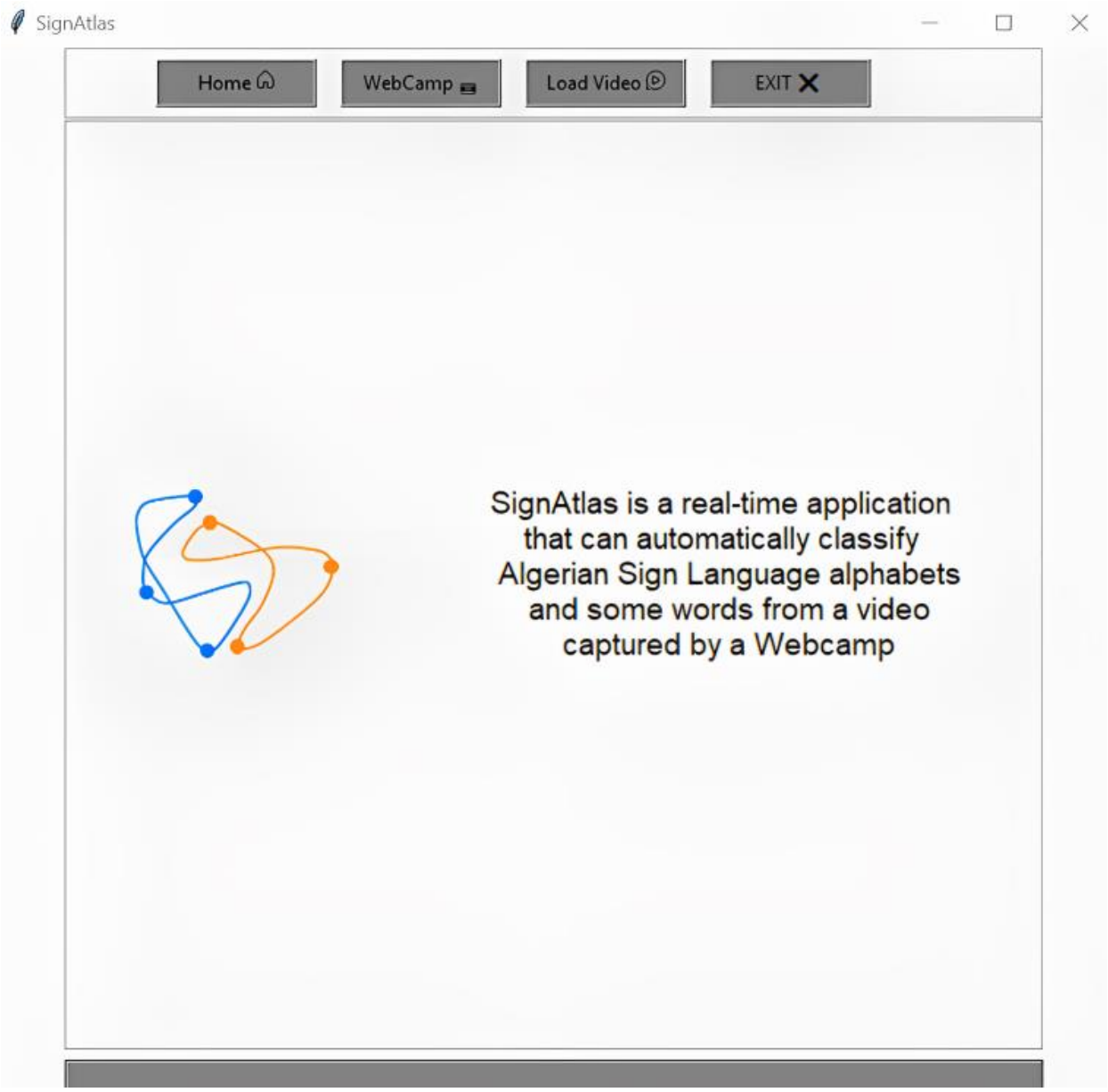


Figure IV.8: graphical interface of the home page of the computer software

Furthermore, the graphical interface offers a "Load Video" option, enabling users to upload pre-recorded videos. The software then analyzes the uploaded video and predicts the corresponding gestures. This feature allows for flexibility in analyzing gestures from different sources.

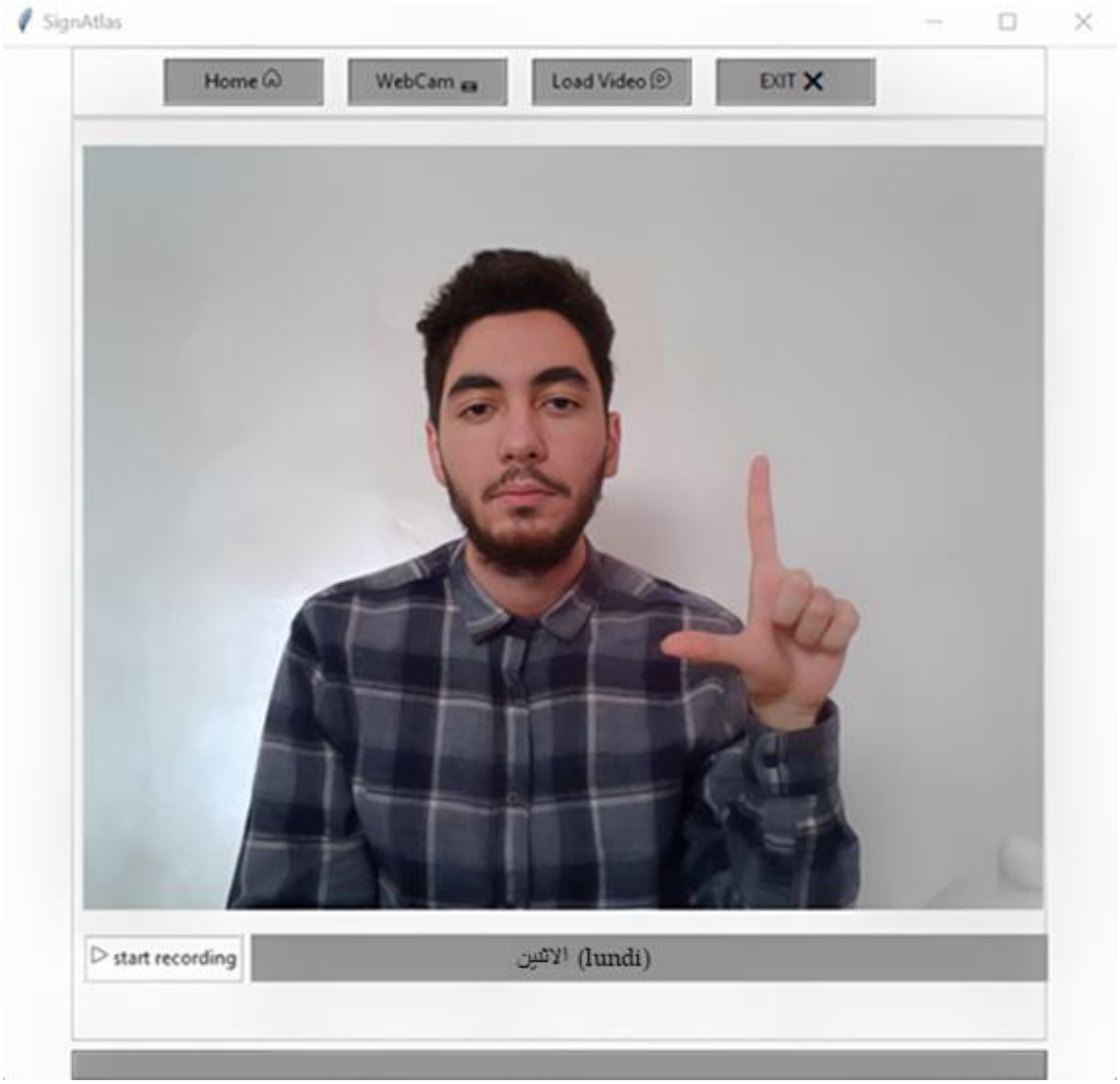


Figure IV.9: graphical interface of the webcam interface (early version)

IV.8 Conclusion

In this chapter, we have implemented and evaluated our approach for recognizing Algerian sign language gestures. We discussed the development environment, hardware specifications, and software requirements. We compared different preprocessing variants and tuned parameters to optimize our model's performance. The classification report and Confusion Matrix provided insights into the model's accuracy and areas for

improvement. We conducted experiments on the SIGAL dataset using various test models and assessed our chosen model's performance on different datasets. Lastly, we showcased the graphical interface of our application on both PC and mobile platforms. Our approach achieved a 99% success rate with an average precision of 98,99%. Overall, this chapter presents a comprehensive exploration of our approach's development, analysis of results, and evaluation of its performance.

Chapter V

Business Model Canvas

V.1 Introduction

In this chapter, we present the Business Model Canvas (BMC) for SignAtlas, our startup aiming to revolutionize communication for the deaf community. The BMC provides a strategic overview of our business model, highlighting key elements such as value proposition, customer segments, channels, and revenue streams. Through the nine essential sections of the BMC, including Key Partners, Key Activities, Key Resources, and more, we outline our strategy to deliver a communication solution that enhances accessibility and inclusion. For a detailed breakdown, please refer to "Table V.1: Business Model Canvas for SignAtlas".

V.2 What is a business model?

V.2.1 Definition

A business model describes the rationale of how an organization creates, delivers and captures value. It can be described through 9 building blocks: Customer Segments, Value Propositions, Channels, Customer Relationships, Revenue Streams, Key Resources, Key Activities, Key Partnerships & Cost Structure. [109]

V.2.1 Originality of the BMC

The Business Model Canvas (BMC), originally developed by Alexander Osterwalder and Yves Pigneur in 2005, emerged as a groundbreaking framework in their book 'Business Model Generation.' Since its introduction, the BMC has gained widespread recognition and has become a staple tool in the field of strategic management and entrepreneurship. [109]

V.2.2 Why use the Business Model Canvas?

1.Create A Shared Language

This concept can become a shared language that allows you to easily describe and manipulate business models to create new strategic alternatives. Without such a shared language it is difficult to systematically challenge assumptions about one’s business and innovate successfully.

2.Simple, Visual and Practical

The canvas is perfect for any good discussion, meeting, or workshop on business model innovation and creates a shared language. We need a concept that everybody understands: one that facilitates description and discussion. We need to start from the same point and talk about the same thing. The challenge is that

the concept must be simple, relevant, and intuitively understandable, while not oversimplifying the complexities of how enterprises function.

3.Discover Opportunities

By going through the process of listing the different parts of your business on the canvas, you begin to visualize and understand the different relationships between the nine building blocks that make up the tool.

4.Iterate Quickly

Whether you're using sticky notes on a real-life Business Model Canvas or you're designing your business model on the Strategyzer app, you can iterate on your designs very quickly. The tool enables you to prototype a first version and simply keep iterating until you've tested enough ideas to find product-market fit.

V.3 The 9 building blocks of the Business Model Canvas

We believe a business model can best be described through nine basic building blocks that show the logic of how a company intends to deliver value and make money. The nine blocks cover the three main areas of a business: desirability, viability and feasibility. The business model is like a blueprint for a strategy to be implemented through organizational structures, processes, and systems. Let's take a look into the three different sections:



Figure V.1: BMC divided sections [109]

V.3.1 Desirability

Value Proposition:

The Value Proposition is the unique bundle of products and services that meet customer needs and differentiate one company from another. It solves problems and satisfies customer requirements, offering a range of benefits. Some Value Propositions are innovative and disruptive, while others enhance existing market offers with additional features.

Customer Segments:

The Customer Segments Building Block identifies the target groups an enterprise aims to serve. Customers are crucial for the survival of any business. To enhance customer satisfaction, companies may group them based on shared needs, behaviors, or attributes. A business model may focus on one or multiple Customer Segments. An organization must consciously decide which segments to serve and which to disregard, allowing for a business model tailored to specific customer needs.

Channels:

Channels are the means through which a company engages and connects with its Customer Segments. They are essential touch points that shape the customer experience. Channels fulfill various roles, such as creating awareness, facilitating product evaluation, enabling purchases, delivering the Value Proposition, and providing post-purchase support.

Customer Relationships:

The Customer Relationships Building Block outlines the relationships a company builds with its Customer Segments. The desired type of relationship should be clearly defined for each segment, which can vary from personal to automated interactions. Customer relationships are driven by motivations such as acquisition, retention, and boosting sales through up-selling.

V.3.2 Viability

Revenue Streams:

The Revenue Streams Building Block represents the cash generated from each Customer Segment, taking into account subtracted costs for earnings. Revenue Streams are the arteries of a business model, flowing from the value customers are willing to pay for. By understanding customer willingness to pay, a company can generate one or more Revenue Streams. Pricing mechanisms may vary, such as fixed prices, bargaining, auctions, market-dependent, volume-dependent, or yield management. There are two types of Revenue Streams: transaction revenues from one-time

payments and recurring revenues from ongoing payments for Value Proposition delivery or post-purchase support.

Cost Structure:

The Cost Structure encompasses all operating costs within a business model. It identifies the key costs associated with creating and delivering value, maintaining Customer Relationships, and generating revenue. After defining Key Resources, Key Activities, and Key Partnerships, costs can be calculated.

V.3.2 Feasibility

Key Resources:

The Key Resources Building Block outlines the essential assets necessary for a business model to function. These resources enable the creation and offering of a Value Proposition, market reach, customer relationship maintenance, and revenue generation. The specific type of business determines the required Key Resources, such as capital-intensive production facilities for a microchip manufacturer or a focus on human resources for a microchip designer. Key Resources can be physical, financial, intellectual, or human in nature, owned or leased, or obtained through key partnerships.

Key Activities:

The Key Activities Building Block outlines the critical actions a company must take for its business model to succeed. Key Activities are the essential tasks required for creating and offering a Value Proposition, reaching markets, maintaining Customer Relationships, and generating revenues. Similar to Key Resources, Key Activities vary depending on the type of business model. For example, software development is a Key Activity for Microsoft, supply chain management for Dell, and problem-solving for McKinsey consultancy.

Partnerships:

The Key Partnerships Building Block outlines the network of suppliers and partners that contribute to the success of a business model. Partnerships are crucial and increasingly integral to business models, serving purposes such as optimization, risk reduction, and resource acquisition. Four types of partnerships can be distinguished:

- Strategic alliances between non-competitors
- Coopetition: strategic partnerships between competitors
- Joint ventures for new business development

- Buyer-supplier relationships for reliable supplies.

V.4 SignAtlas

After familiarizing ourselves with the definition of each building block and understanding the Business Model Canvas (BMC) concept, we now shift our focus to our own BMC model for SignAtlas. This particular model represents our participation in the startup ecosystem under the esteemed Medea Incubator. By utilizing the BMC framework, we have crafted a comprehensive and dynamic representation of our business model.

We will now delve into each section of our BMC, highlighting the key components that drive SignAtlas forward. For a detailed visual representation, please refer to Table V.1

Key Partners: Deaf community organizations, local government agencies, and technology providers.

Key Activities: Developing the mobile and computer application, creating and maintaining a sign language database, user testing, and continuous improvement.

Key Resources: Skilled software developers and machine learning experts, sign language experts, access to necessary technological tools and infrastructure, funding for research, development, and marketing efforts.

Value Proposition: Communication solution for deaf and non-sign language speakers, enhancing accessibility and inclusion.

Customer Relationships: Continuous engagement, customer support, partnerships with deaf community organizations.

Customer Segments: Deaf individuals, non-deaf individuals, educational institutions, healthcare providers, legal professionals.

Channels: Mobile app stores, desktop version, online platforms, collaborations with relevant organizations.

Cost Structure: Development and maintenance costs, data storage, marketing expenses, salaries, operational costs.

Revenue Streams: Freemium model with subscription-based premium features, potential partnerships for revenue sharing.

Table V.1: Business Model Canvas for SignAtlas

Key Partners

- Deaf community organizations for collaboration and support.
- Local government agencies for potential funding and resources.
- Technology providers for necessary software and hardware components.

Key Activities

- Developing the mobile and computer application for recognizing Algerian Sign Language.
- Creating and maintaining a comprehensive sign language database.
- Conducting user testing and feedback collection.
- Continuous improvement of the application based on user needs and advancements in technology.

Key Resources

- Skilled software developers and machine learning experts.
- Sign language experts for accurate interpretation and translation.
- Access to necessary technological tools and infrastructure.
- Funding for research, development, and marketing efforts.

Value proposition

- Providing a solution for communication barriers between the deaf community and non-sign language speakers.
- Enabling easier and more efficient communication through sign language recognition and live transcription.
- Empowering the deaf community by enhancing their accessibility and inclusion in various aspects of life.

Customer Relationships

- Continuous engagement with the deaf community for feedback and improvement.
- Providing customer support for technical issues and application usage.
- Building partnerships with organizations serving the deaf community for mutual benefit.

Channels

- Mobile application stores (Google Play Store, Apple App Store) for distribution
- Desktop Version: Enhancing communication in police stations, courtrooms
- Online platforms and social media for marketing and awareness
- Collaborating with relevant organizations and institutions to reach target users.

Customer Segments

- Deaf individuals who rely on sign language for communication.
- Non-deaf individuals who want to communicate effectively with the deaf community.
- Educational institutions, healthcare providers, and legal professionals dealing with deaf individuals.

Cost Structure

- Development and maintenance costs for the mobile application
- Data storage and server costs for the sign language database
- Marketing and promotional expenses
- Salaries and operational costs

Revenue Streams

- Freemium model with basic features available for free and additional premium features for a subscription fee
- Potential partnerships and collaborations for revenue sharing

V.5 Conclusion

The Business Model Canvas presented in this chapter provides a comprehensive overview of SignAtlas, our startup committed to improving communication for the Algerian deaf community. Through our mobile application and desktop version, we aim to empower individuals by enabling effective communication and promoting inclusivity. The BMC serves as a strategic guide, outlining our value proposition, target customer segments, revenue streams, and key activities. By addressing identified challenges and leveraging innovative technologies, SignAtlas seeks to make a meaningful impact on the lives of deaf individuals and foster a more inclusive society.

For more information on SignAtlas, including some use cases, it benefits, and a SWOT analysis, please refer to the annex chapter.

General Conclusion

In conclusion, this document has made significant progress in addressing the challenges of recognizing Algerian Sign Language (LSA) and has contributed to the development of effective LSA recognition systems. Through our research, we have achieved notable milestones and gained valuable insights.

One of the major accomplishments of this document is the successful development of a real-time application that accurately interprets LSA gestures. Our work involved creating five variants and extensively testing each one to determine the most effective approach. We specifically focused on frame extraction from videos, allowing us to extract significant frames for model input, ensuring real-time processing and response.

By employing machine learning techniques, particularly an autoencoder with attention, we achieved a balance between accuracy and response time, resulting in a recognition system with high reliability and efficient performance. Our rigorous evaluation and comparison of the different variants allowed us to select the optimal approach for LSA recognition.

The extensive dataset collection, involving visits to deaf schools across Algeria and collaboration with the deaf community, has been instrumental in training and evaluating our recognition models. This diverse dataset, combined with the selected frame extraction method, ensured the robustness and scalability of our system.

The involvement of the deaf community throughout the research process has been invaluable. Their expertise, feedback, and cultural insights have guided our work, ensuring the accuracy and cultural authenticity of the collected data. By actively involving them, we have created a recognition system that aligns with their needs and preferences, ultimately enhancing communication and accessibility for the deaf community.

Looking ahead, there are several perspectives to consider for further research:

- Dataset Expansion: To further improve the accuracy and generalizability of LSA recognition systems, expanding the dataset is crucial. Collecting additional diverse samples, including a wider range of LSA signs, gestures, and expressions, would enhance the system's performance and adaptability.

- Integration of Facial Expression and Emotion Recognition: To provide a more comprehensive understanding of sign language communication, incorporating facial expression and emotion recognition as additional features would enrich the interpretation of LSA gestures. This can enhance the system's ability to capture nuanced expressions and improve overall accuracy.
- Continuous Sign Language Recognition: Building upon our work with isolated gestures, exploring continuous sign language recognition is essential. Developing models that can interpret complete sign language sentences and conversations would enable more natural and fluid communication for the deaf community, extending the system's usability and impact.

In summary, this document has contributed to the advancement of LSA recognition systems by developing a real-time application with optimized frame extraction and machine learning techniques. Our collaboration with the deaf community and careful evaluation of different variants ensured the accuracy, reliability, and cultural authenticity of our system. As we move forward, expanding the dataset, incorporating facial expression recognition, and exploring continuous sign language recognition will further enhance the effectiveness and inclusivity of LSA recognition systems.

Bibliography

- [1] A. C. Hamill and C. H. Stein, “Culture and empowerment in the deaf community: An analysis of internet weblogs,” Journal of Community and Applied Social Psychology, vol. 21, pp. 388–406, 2011.
- [2] D. M. Perlmutter, “What is Sign Language”. Linguistic Society of America, Accessed: Feb. 23, 2023. [Online]. Available: What is Sign Language? | Linguistic Society of America
- [3] G. Simons, “Welcome to the 25th edition”. Ethnologue, Accessed: Feb. 05, 2023. [Offline]. Available: https://www.ethnologue.com/ethnoblog/gary-simons/welcome-25th-edition
- [4] P. A. Judéaux, “French Sign Language: a language in its own right”. TradOnline, Accessed: Feb. 23, 2023. [Online]. Available: French Sign Language: a language in its own right - Tradonline
- [5] “Second International Congress on Education of the Deaf”. Wikipedia, Accessed: Feb. 23, 2023. [Online]. Available: Second International Congress on Education of the Deaf - Wikipedia
- [6] T. Humphries, P. Kushalnagar, G. Mathur, D. J. Napoli, C. Padden, C. Rathmann, and S. Smith, “Sign Language Is Best for Deaf Children”. School of Social Service Administration Magazine, Accessed: Feb. 23, 2022. [Online]. Available: Sign Language Is Best for Deaf Children | Crown Family School of Social Work, Policy, and Practice (uchicago.edu)
- [7] World Federation of the Deaf. “21st International Congress on the Education of the Deaf (ICED) in July 2010 in Vancouver”. World Federation of the Deaf, Accessed: Feb. 23, 2022. [Online]. Available: 21st International Congress on the Education of the Deaf (ICED) in July 2010 in Vancouver, Canada - WFD (wfdeaf.org)
- [8] FrancoSourd. "Accueil." FrancoSourd, Accessed: Feb. 23, 2023. [Online]. Available: https://www.francosourd.com/

- [9] Nekkaa, Foudil. "Détection automatique de la main: Application à la reconnaissance de la langue des signes arabe." Spécialité: Systèmes Distribués et Méthodes Formelles (SDMF), Université Abdelhamid Mehri-Constantine 2, 2014/2015.
- [10] Julien, Thomet. Une vue d'ensemble de la reconnaissance de gestes, Département d'informatique Université de Fribourg, vol.8.
- [11] Bossard, Bruno. Problèmes posés par la reconnaissance de gestes en Langue des Signes. LIMSI-CNRS - Université Paris XI, Orsay. Soutenu 24–27 juin 2002.10 p.
- [12] LAKHFIF, Abdelaziz. Un Environnement de Traduction Automatique du Texte Arabe vers la Langue des Signes Algérienne (LSA). Magistère en Informatique. Université Badji Mokhtar-Annaba. Soutenu le Juillet 2009.134p.
- [13] G. J. Grimes, "Digital data entry glove interface device," in US Patent US4414537A, 1983.
- [14] I. Adeyanju, O. Bello, and M. Adegboye, "Machine learning methods for sign language recognition: A critical review and analysis," Intelligent Systems with Applications, vol. 12, p. 200 056, 2021, ISSN: 2667-3053. DOI: https://doi.org/ 10.1016/j.iswa.2021.200056. [Online]. Available: Machine learning methods for sign language recognition: A critical review and analysis - ScienceDirect
- [15] M. Al-Qurishi, T. Khalid, and R. Souissi, "Deep learning for sign language recognition: Current techniques, benchmarks, and open issues," IEEE Access, vol. 9, pp. 126 917–126 951, 2021. DOI: 10.1109/ACCESS.2021.3110912.
- [16] D. Mitchell, "ASL Sign Language Dictionary". SignASL, Accessed: Feb. 29, 2023. [Online]. Available: American Sign Language ASL Video Dictionary - nice to meet you (signasl.org)
- [17] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudrealt, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, C. Vogler, and M. Morris, "Sign language recognition, generation, and translation: An interdisciplinary perspective," in The 21st International ACM SIGACCESS Conference on Computers and Accessibility, Aug. 2019, pp. 16–31.
- [18] Z. Zhang, J. Pu, L. Zhuang, W. Zhou, and H. Li, "Continuous sign language recognition via reinforcement learning," in 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 285–289. DOI: 10.1109/ICIP.2019.8802972.
- [19] N. Aloysius, G. M, and P. Nedungadi, "Incorporating relative position information in transformer-based sign language recognition and translation," IEEE Access, vol. 9, pp. 145 929–145 942, 2021. DOI: 10.1109/ACCESS.2021.3122921.
- [20] J. Hou, X.-Y. Li, P. Zhu, Z. Wang, Y. Wang, J. Qian, and P. Yang, "Signspeaker: A real-time, high-precision smartwatch-based sign language translator," in The 25th Annual International

Conference on Mobile Computing and Networking, ser. MobiCom '19, New York, NY, USA: Association for Computing Machinery, 2019, ISBN: 9781450361699. [Online]. Available: https://doi.org/10.1145/3300061. 3300117.

- [21] A. Mittal, P. Kumar, P. P. Roy, R. Balasubramanian, and B. B. Chaudhuri, "A modified lstm model for continuous sign language recognition using leap motion," IEEE Sensors Journal, vol. 19, no. 16, pp. 7056–7063, 2019. DOI: 10.1109/ JSEN.2019.2909837.
- [22] S. Tang, D. Guo, R. Hong, and M. Wang, "Graph-based multimodal sequential embedding for sign language translation," IEEE Transactions on Multimedia, pp. 1–1, 2021. DOI: 10.1109/TMM.2021.3117124.
- [23] A. Duarte, S. Palaskar, L. Ventura, D. Ghadiyaram, K. DeHaan, F. Metze, J. Torres, and X. Giro-i-Nieto, "How2sign: A large-scale multimodal dataset for continuous american sign language," in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 2734–2743. DOI: 10.1109/CVPR46437. 2021.00276.
- [24] intelligence artificielle - larousse. https://www.larousse.fr/encyclopedie/ divers/intelligence_artificielle/187257.
- [25] Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning – crayon data. https://shorturl.at/dyEFJ
- [26] What is machine learning ? https://www.cognizantsoftvision.com/blog/ what-is-machine-learning/.
- [27] Neural networks | a beginners guide - geeksforgeeks. https://www.geeksforgeeks. org/neural-networks-a-beginners-guide/.
- [28] Karim Aderghal. Classification of multimodal mri images using deep learning : Application to the diagnosis of alzheimer's disease., 2021
- [29] Ahlem Ferchichi. Propagation et réduction des incertitudes dans les modèles de changement d'occupation des sols, 2017. https://www.researchgate.net/ publication/322263283.
- [30] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining. concepts and techniques, 3rd edition (the morgan kaufmann series in data management systems), 2011.
- [31] Categorical crossentropy loss function | peltarion platform. https://peltarion. com/knowledge-center/documentation/modeling-view/build-an-ai-model/ loss-functions/categorical-crossentropy.
- [32] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, Ping Tak, and Peter Tang. On large-batch training for deep learning : Generalization gap and sharp minima, 2016.

- [33] 4.7. forward propagation, backward propagation, and computational graphs — dive into deep learning 0.17.5 documentation. https://d2l.ai/chapter_ multilayer-perceptrons/backprop.html.
- [34] Abdelouaf, Hicham. "Deaf Education in Algeria: Is it a Sustainable Approach? تعليم الصم في الجزائر: نهج مستدام؟" Sociology Review Volume:05 / N°:02(2021), p 417-429, Universitat Internacional de Catalunya (Spain), hicham@uic.es. Received: 30/07/2021 Accepted:30/08/2021 Published:27./10/2021. Accessed: Mar. 31, 2023. [Online]. Available: https://www.asjp.cerist.dz/en/downArticle/432/5/2/166788
- [35] Lanesman, S. (2013). Algerian Jewish Sign language: Its emergence and survival (Master). Ishara Press. https://doi.org/10.26530/OAPEN_588714
- [36] Lanesman, S., & Meir, I. (2012). Algerian Jewish Sign Language: A sociolinguistic sketch. In U. Zeshan & C. de Vos (Eds.), Sign Languages in Village Communities (pp. 361-364). Berlin, Boston: DE GRUYTER.
- [37] Lanesman, S., & Meir, I. (2012b). The survival of Algerian Jewish Sign Language alongside Israeli Sign Language in Israel. Anthropological and linguistic insights. Retrieved from https://www.academia.edu/10678229/The_survival_of_Algerian_Jewish_Sign_Language_alongside_Israeli_Sign_Language_in_Israel.
- [38] Djama, Amal. (2016). Les points communs entre la Algerian Sign Language (LSA) - dialecte de Laghouat, Sud de l'Algérie - et la Langue des Signes Française (LSF). Dossier, licence SCL « Acquisition et dysfonctionnement » (SCL F14), Licence 3, AMU, Faculté ALLSHS d'Aix-en-Provence. 5pp.
- [39] Mansour, M. S. (2007). Langage et surdité, descriptive de la langue des signes des sourds Oranais (Magistère), Université d'Oran Es-Sénia. 124pp.
- [40] Abdelouafi, H., (2018). Teaching sign language to the deaf children in Adrar, Algeria. [Master's thesis, University of Ahmed Draia Adrar].
- [41] S. Khan, M. Sajjad, T. Hussain, A. Ullah, and A. S. Imran, "A review on traditional machine learning and deep learning models for wbcs classification in blood smear images," IEEE Access, vol. 9, pp. 10 657–10 673, 2021. DOI: 10.1109/ ACCESS.2020.3048172.
- [42] M. M. Adnan, M. S. M. Rahim, A. Rehman, Z. Mehmood, T. Saba, and R. A. Naqvi, "Automatic image annotation based on deep learning models: A systematic review and future challenges," IEEE Access, vol. 9, pp. 50 253–50 264, 2021. DOI: 10.1109/ACCESS.2021.3068897.
- [43] A. Wu, Y. Han, Y. Yang, Q. Hu, and F. Wu, "Convolutional reconstruction-tosequence for video captioning," IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 11, pp. 4299–4308, 2020. DOI: 10 . 1109 / TCSVT . 2019.2956593.

- [44] K. Khurana and U. Deshpande, “Video question-answering techniques, benchmark datasets and evaluation metrics leveraging video captioning: A comprehensive survey,” IEEE Access, vol. 9, pp. 43 799–43 823, 2021. DOI: 10 . 1109 / ACCESS.2021.3058248.
- [45] “Multilayer perceptron”. Wikipedia, Accessed: Apr. 01, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Multilayer_perceptron.
- [46] “Recurrent neural network”. Wikipedia, Accessed: Apr. 01, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network.
- [47] R. Siriak, I. Skarga-Bandurova, and Y. Boltov, “Deep convolutional network with long short-term memory layers for dynamic gesture recognition,” in 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), vol. 1, 2019, pp. 158–162. DOI: 10.1109/IDAACS.2019.8924381.
- [48] “Sigmoid function”. Wikipedia, Accessed: Apr. 01, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Sigmoid_function.
- [49] “Hyperbolic function”. Wikipedia, Accessed: Apr. 01, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Hyperbolic_functions.
- [50] X. Han, F. Lu, J. Yin, G. Tian, and J. Liu, “Sign language recognition based on r(2+1)d with spatial-temporal-channel attention,” IEEE Transactions on HumanMachine Systems, vol. 52, no. 4, pp. 687–698, 2022. DOI: 10.1109/THMS.2022. 3144000.
- [51] P. Xie, M. Zhao, and X. Hu, “Pisltrc: Position-informed sign language transformer with content-aware convolution,” IEEE Transactions on Multimedia, vol. 24, pp. 3908–3919, 2022. DOI: 10.1109/TMM.2021.3109665.
- [52] F. B. Slimane and M. Bouguessa, “Context matters: Self-attention for sign language recognition,” in 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 7884–7891. DOI: 10.1109/ICPR48806.2021.9412916.
- [53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, vol. 1, 2017, pp. 6000–6010.
- [54] Ballard, D. H. (1987). Modular learning in neural networks. Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 101-108.
- [55] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798-1828.
- [56] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

- [57] J. Huang, W. Zhou, H. Li, and W. Li, "Sign language recognition using 3d convolutional neural networks," in 2015 IEEE International Conference on Multimedia and Expo (ICME), 2015, pp. 1–6. DOI: 10.1109/ICME.2015.7177428.
- [58] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015, pp. 136–140. DOI: 10.1109/ ACPR.2015.7486481.
- [59] O. Koller, H. Ney, and R. Bowden, "Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3793–3802. DOI: 10.1109/CVPR.2016.412.
- [60] S. Y. Kim, H. G. Han, J. W. Kim, S. Lee, and T. W. Kim, "A hand gesture recognition sensor using reflected impulses," IEEE Sensors Journal, vol. 17, no. 10, pp. 2975–2976, 2017. DOI: 10.1109/JSEN.2017.2679220.
- [61] S. Shin, Y. Baek, J. Lee, Y. Eun, and S. H. Son, "Korean sign language recognition using emg and imu sensors based on group-dependent nn models," in IEEE Symposium Series on Computational Intelligence (SSCI), 2017, pp. 1–7. DOI: 10 . 1109/SSCI.2017.8280908.
- [62] O. Mercanoglu Sincan, A. O. Tur, and H. Yalim Keles, "Isolated sign language recognition with multi-scale features using lstm," in 27th Signal Processing and Communications Applications Conference (SIU), 2019, pp. 1–4. DOI: 10.1109/SIU. 2019.8806467.
- [63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for largescale image recognition," CoRR, vol. abs/1409.1556, 2015.
- [64] E. K. Kumar, P. V. V. Kishore, M. T. K. Kumar, D. A. Kumar, and A. S. C. S. Sastry, "Three-dimensional sign language recognition with angular velocity maps and connived feature resnet," IEEE Signal Processing Letters, vol. 25, no. 12, pp. 1860– 1864, 2018. DOI: 10.1109/LSP.2018.2877891.
- [65] Y. Liao, P. Xiong, W. Min, W. Min, and J. Lu, "Dynamic sign language recognition based on video sequence with blstm-3d residual networks," IEEE Access, vol. 7, pp. 38 044–38 054, 2019. DOI: 10.1109/ACCESS.2019.2904749.
- [66] K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in IEEE International Conference on Big Data (Big Data), 2018, pp. 4896–4899. DOI: 10.1109/BigData.2018.8622141.
- [67] A. Das, S. Gawde, K. Suratwala, and D. Kalbande, "Sign language recognition using deep learning on custom processed static gesture images," in International Conference on Smart City and Emerging Technology (ICSCET), 2018, pp. 1–6. DOI: 10.1109/ICSCET.2018.8537248.

- [68] S.-Z. Li, B. Yu, W. Wu, S.-Z. Su, and R.-R. Ji, "Feature learning based on sae–pca network for human gesture recognition in rgbd images," Neurocomputing, vol. 151, pp. 565–573, 2015, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom. 2014.06.086.
- [69] H. B. Nguyen and H. N. Do, "Deep learning for American sign language fingerspelling recognition system," in 26th International Conference on Telecommunications (ICT), 2019, pp. 314–318. DOI: 10.1109/ICT.2019.8798856.
- [70] O. Koller, H. Ney, and R. Bowden, "Deep learning of mouth shapes for sign language," in IEEE International Conference on Computer Vision Workshop (ICCVW), 2015, pp. 477–483. DOI: 10.1109/ICCVW.2015.69.
- [71] O. Koller, N. C. Camgoz, H. Ney, and R. Bowden, "Weakly supervised learning with multi-stream cnn-lstm-hmms to discover sequential parallelism in sign language videos," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 9, pp. 2306–2320, 2020. DOI: 10.1109/TPAMI.2019.2911077.
- [72] T. Liu, W. Zhou, and H. Li, "Sign language recognition with long short-term memory," in IEEE International Conference on Image Processing (ICIP), 2016, pp. 2871– 2875. DOI: 10.1109/ICIP.2016.7532884.
- [73] D. Avola, M. Bernardi, L. Cinque, G. L. Foresti, and C. Massaroni, "Exploiting recurrent neural networks and leap motion controller for the recognition of sign language and semaphoric hand gestures," IEEE Transactions on Multimedia, vol. 21, no. 1, pp. 234–245, 2019. DOI: 10.1109/TMM.2018.2856094.
- [74] O. M. Sincan and H. Y. Keles, "Autsl: A large scale multi-modal turkish sign language dataset and baseline methods," IEEE Access, vol. 8, pp. 181 340–181 355, 2020. DOI: 10.1109/ACCESS.2020.3028072.
- [75] Q. Xiao, M. Qin, P. Guo, and Y. Zhao, "Multimodal fusion based on lstm and a couple conditional hidden markov model for chinese sign language recognition," IEEE Access, vol. 7, pp. 112 258–112 268, 2019. DOI: 10.1109/ACCESS. 2019.2925654.
- [76] B. Xu, S. Huang, and Z. Ye, "Application of tensor train decomposition in s2vt model for sign language recognition," IEEE Access, vol. 9, pp. 35 646–35 653, 2021. DOI: 10.1109/ACCESS.2021.3059660.
- [77] W. Li, H. Pu, and R. Wang, "Sign language recognition based on computer vision," in IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2021, pp. 919–922. DOI: 10.1109/ICAICA52286.2021.9498024

- [78] S. He, “Research of a sign language translation system based on deep learning,” in 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), 2019, pp. 392–396. DOI: 10.1109/AIAM48774.2019.00083
- [79] I. Gruber, Z. Kr ˇnoul, M. Hrúz, J. Kanis, and M. Bohá ˇcek, “Mutual support of data modalities in the task of sign language recognition,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021, pp. 3419– 3428. DOI: 10.1109/CVPRW53098.2021.00381.
- [80] P. Das, T. Ahmed, and M. F. Ali, “Static hand gesture recognition for american sign language using deep convolutional neural network,” in IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1762–1765. DOI: 10.1109/TENSYMP50017. 2020.9230772.
- [89] S. Kumar Santhosh, S. R. Kayshap, S. R. Lakka, and A. Arya, “Signed english recognition with spatio-temporal graphs and language modelling,” in IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1–7. DOI: 10.1109/I2CT54291.2022.9824226.
- [82] P. Likhar and R. G. N, “Indian sign language translation using deep learning,” in IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC), 2021, pp. 1– 4. DOI: 10.1109/R10-HTC53172.2021.9641599.
- [83] S. Kim, C. J. Kim, H.-M. Park, Y. Jeong, J. Y. Jang, and H. Jung, “Robust keypoint normalization method for korean sign language translation using transformer,” in International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 1303–1305. DOI: 10.1109/ICTC49870.2020.9289551.
- [84] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10 . 18653 / v1 / N19-1423. [Online]. Available: https://aclanthology.org/N19-1423
- [85] N. Cihan Camgöz, O. Koller, S. Hadfield, and R. Bowden, “Sign language transformers: Joint end-to-end sign language recognition and translation,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10 020–10 030. DOI: 10.1109/CVPR42600.2020.01004.
- [86] Numpy. https://numpy.org/.
- [87] Home - opencv. https://opencv.org/.
- [88] Matplotlib — visualization with python. https://matplotlib.org/.
- [89] Keras : the python deep learning api. https://keras.io/.

- [90] Tensorflow. https://www.tensorflow.org/.
- [91] Google Mediapipe https://google.github.io/mediapipe/
- [92]. Ronchetti, F., Quiroga, F., Estrebou, C.A., Lanzarini, L.C., Rosete, A.: Lsa64: an argentinian sign language dataset. In: XXII Congreso Argentino de Ciencias de la Computaci´on (CACIC 2016) (2016)
- [93] Ronchetti, F., Quiroga, F., Estrebou, C., Lanzarini, L., Rosete, A.: sign languague recognition without frame-sequencing constraints: a proof of concept on the argentinian sign language. In: Montes-y-G´omez, M., Escalante, H.J., Segura, A., de Dios Murillo, J. (eds.) IBERAMIA 2016. LNCS (LNAI), vol. 10022, pp. 338–349. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47955-2 28
- [94] Rodr´ıguez, J., Mart´ınez, F.: Towards on-line sign language recognition using cumulative SD-VLAD descriptors. In: Serrano C., J.E., Mart´ınez-Santos, J.C. (eds.) CCC 2018. CCIS, vol. 885, pp. 371–385. Springer, Cham (2018). https://doi.org/ 10.1007/978-3-319-98998-3 29
- [95] Konstantinidis, D., Dimitropoulos, K., Daras, P.: Sign language recognition based on hand and body skeletal data. In: 2018–3DTV-Conference: The True VisionCapture, Transmission and Display of 3D Video (3DTV-CON), pp. 1–4. IEEE (2018)
- [96] Konstantinidis, D., Dimitropoulos, K., Daras, P.: A deep learning approach for analyzing video and skeletal features in sign language recognition. In: 2018 IEEE International Conference on Imaging Systems and Techniques (IST), pp. 1–6. IEEE (2018)
- [94] Masood, S., Srivastava, A., Thuwal, H.C., Ahmad, M.: Real-time sign language gesture (word) recognition from video sequences using CNN and RNN. In: Bhateja, V., Coello Coello, C.A., Satapathy, S.C., Pattnaik, P.K. (eds.) Intelligent Engineering Informatics, pp. 623–632. Springer Singapore, Singapore (2018)
- [98] Shah, J.A., et al.: Deepsign: a deep-learning architecture for sign language. Master's thesis, University of Texas at Arlington (2018)
- [99] Zhang, X., Li, X.: Dynamic gesture recognition based on MEMP network. Future Internet 11, 91 (2019). https://doi.org/10.3390/fi11040091
- [100] Imran, J., Raman, B.: Deep motion templates and extreme learning machine for sign language recognition. Vis. Comput. 36(6), 1233–1246 (2020)
- [101] Elsayed, E.K., Fathy, D.R.: Semantic deep learning to translate dynamic sign language. Int. J. Intell. Eng. Syst. 14 (2021). https://doi.org/10.22266/ijies2021.0228. 3

- [102] Ronchetti, F., Quiroga, F., Estrebou, C., Lanzarini, L., & Rosete, A. (2016). LSA64: A Dataset of Argentinian Sign Language. XX II Congreso Argentino de Ciencias de la Computación (CACIC).
- [103] DaiASLLRP: American Sign Language Lexicon Video Dataset," Boston University, Accessed: Apr. 01, 2023. [Online]. Available: https://www.bu.edu/asllrp/av/dai-asllvd.html.
- [104] Amorim, C.C.D.; Macêdo, D.; Zanchettin, C. Spatial-temporal graph convolutional networks for sign language recognition. In Proceedings of the International Conference on Artificial Neural Networks, Munich, Germany, 17–19 September 2019.
- [105] Al-Hammadi, M., Bencherif, M. A., Alsulaiman, M., Muhammad, G., Mekhtiche, M. A., Abdul, W., Alohali, Y. A., Alrayes, T. S., Mathkour, H., Faisal, M., Algabri, M., Altaheri, H., Alfakih, T., & Ghaleb, H. (2022). Spatial Attention-Based 3D Graph Convolutional Neural Network for Sign Language Recognition. Sensors, 22(12), 4558. doi:10.3390/s22124558
- [106] Welcome to python.org. https://www.python.org/.
- [107] Visual studio code - code editing. redefined. https://code.visualstudio.com/.
- [108] Cleison Correia de Amorim, David Macêdo, and Cleber Zanchettin. "Spatial-Temporal Graph Convolutional Networks for Sign Language Recognition." CoRR, abs/1901.11164, 2019. [Online]. Available: http://arxiv.org/abs/1901.11164
- [109] "Business Models." Strategyzer. Available at: https://www.strategyzer.com/expertise/business-models.

Chapter Appendix

1.Mediapipe

MediaPipe Solutions provides a suite of libraries and tools for you to quickly apply artificial intelligence (AI) and machine learning (ML) techniques in your applications. You can plug these solutions into your applications immediately, customize them to your needs, and use them across multiple development platforms. MediaPipe Solutions is part of the MediaPipe open-source project, so you can further customize the solutions code to meet your application needs. The MediaPipe Solutions are available across multiple platforms. Each solution includes one or more models, and you can customize models for some solutions as well. The following list shows what solutions are available for each supported platform and if you can use Model Maker to customize the model:

Solution	Android	Web	Python	Customize model
Object detection	●	●	●	●
Image classification	●	●	●	●
Image segmentation	●	●	●	
Interactive segmentation	●	●	●	
Hand landmark detection	●	●	●	
Gesture recognition	●	●	●	●
Image embedding	●	●	●	
Face detection	●	●	●	
Face landmark detection	●	●	●	
Pose landmark detection	●	●	●	
Text classification	●	●	●	●
Text embedding	●	●	●	
Language detector	●	●	●	
Audio classification	●	●	●	

Figure 1: The MediaPipe Solutions

For our project, we primarily focused on implementing the Hand Landmarker task using Mediapipe solutions. Utilizing the powerful features offered by Mediapipe, we were able to accurately detect and track hand landmarks in real-time video and image inputs. (The ones highlighted with red borders are the one been utilized in this project)

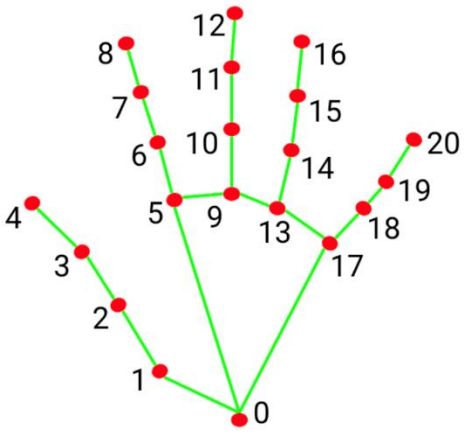
1.1 Hand landmarks detection

The MediaPipe Hand Landmarker task lets you detect the landmarks of the hands in an image. You can use this Task to localize key points of the hands and render visual effects over the hands. This task operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs hand landmarks in image coordinates, hand landmarks in world coordinates and handedness (left/right hand) of multiple detected hands.

1.2 Hand landmarks Models

The Hand Landmarker uses a model bundle with two packaged models: a palm detection model and a hand landmarks detection model. You need a model bundle that contains both these models to run this task.

The hand landmark model bundle detects the keypoint localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 2: The hand landmark of 21 hand-knuckle

The Palm detection model locates hands within the input image, and the hand landmarks detection model identifies specific hand landmarks on the cropped hand image defined by the palm detection model.

Since running the palm detection model is time consuming, when in video or live stream running mode, Hand Landmarker uses the bounding box defined by the hand landmarks model in one frame to localize the region of hands for subsequent frames. Hand Landmarker only re-triggers the palm detection model if the hand landmarks model no longer identifies the presence of hands or fails to track the hands within the frame. This reduces the number of times Hand Landmarker tiggers the palm detection model

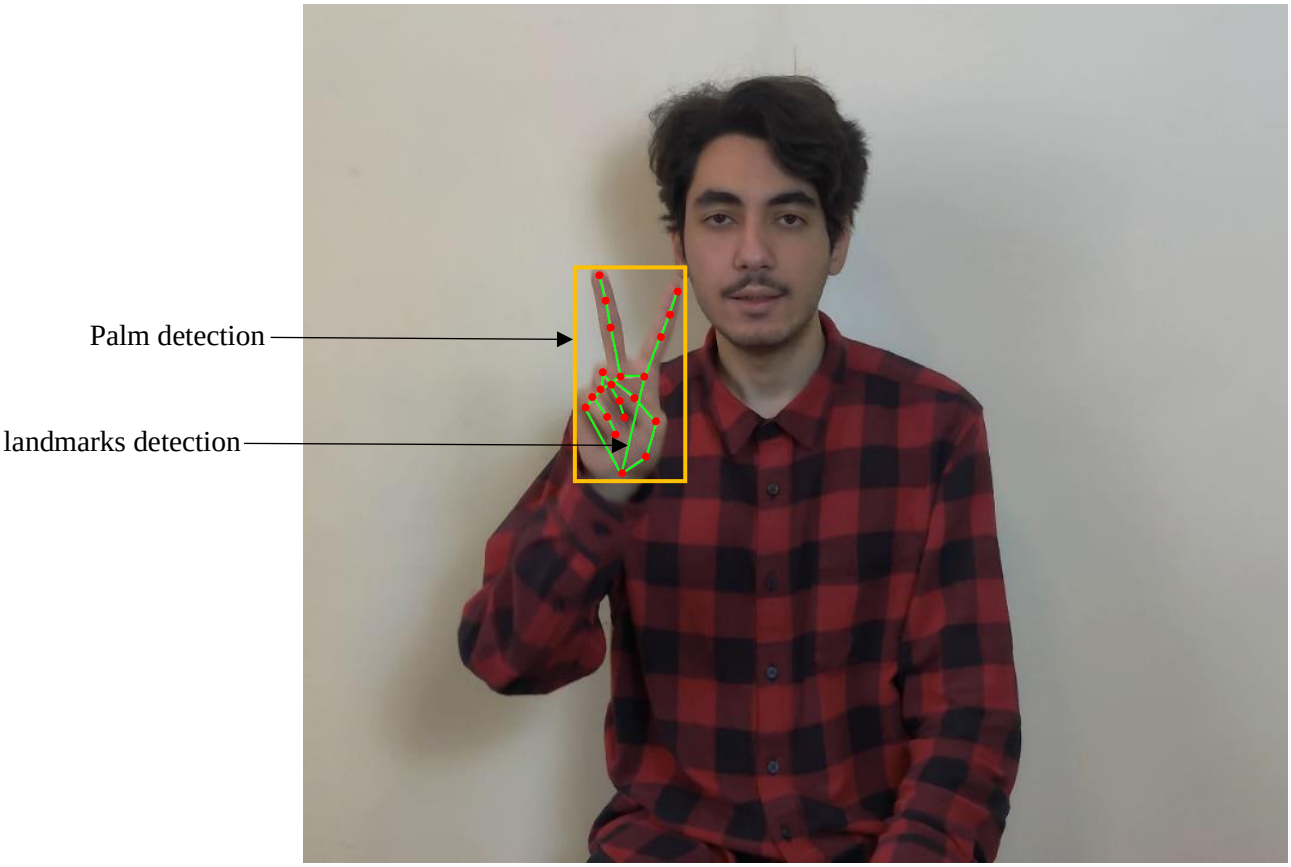


Figure 3: Example of detection palm with 21 Landmark on SIGAL dataset

While Mediapipe provided reliable hand tracking capabilities, it's important to note that in some cases, not all 21 hand landmarks were detected. Occasionally, the detection may be limited to a subset of the landmarks or miss certain hand poses or orientations.

2. Details of the dataset SIGAL

2.1 Handshapes

Figures 6 and 7 showcase the distinct handshapes of the right and left hand, respectively, for each sign class. Numerous handshapes exhibit repetition, although their 2D projection may vary due to hand rotation.

2.2 Positions

In Figure 5, the mean initial and final positions for each hand, along with the covariance, are depicted. Although some signs can be distinguished based on their positions, there is substantial overlap observed in the majority of cases.

2.3 Trajectories

Person 6's sample trajectories for each sign are illustrated in Figure 4. There is a noticeable amount of overlap in the movements of both one-handed and two-handed signs. Noteworthy examples of one-handed signs with similar trajectories include signs 2, 3, 48, 40, and 15. Similarly, two-handed signs like signs 52, 53, and 54 also exhibit comparable movement patterns.

2.4 Amount of movement

Figure 8 shows the amount of movement for each hand, measured with coordination (x,y) of the central landmark in the trajectory of the hand. The movement in the left hand is significantly smaller than that of the right hand in many signs, consistent with the fact that the right hand is the dominant one for all the signers.

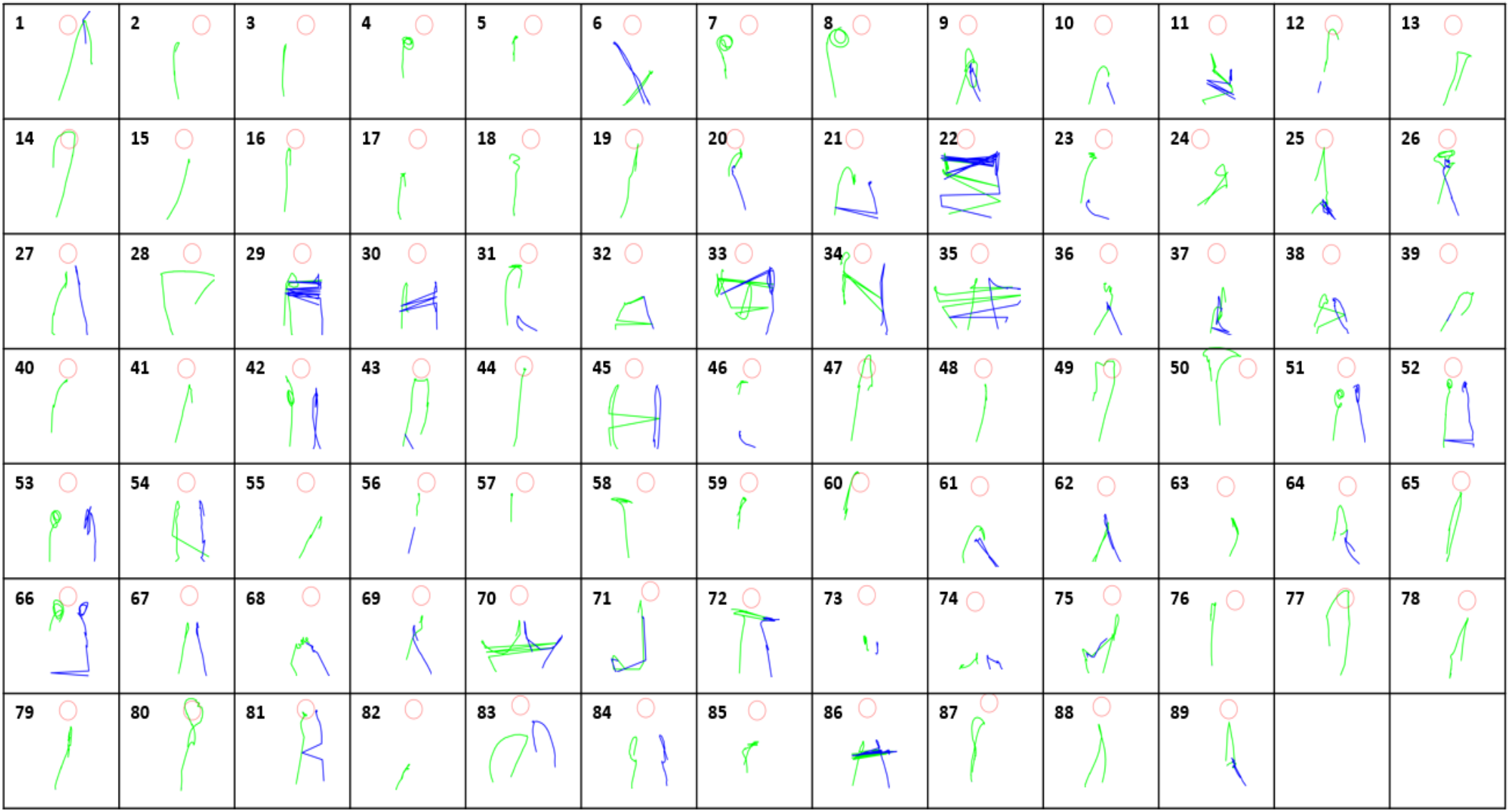


Figure 4: Sample trajectories for each sign in SIGAL. The left-hand trajectory is shown in light green, the right-hand one in blue, and the head position as a red circle.

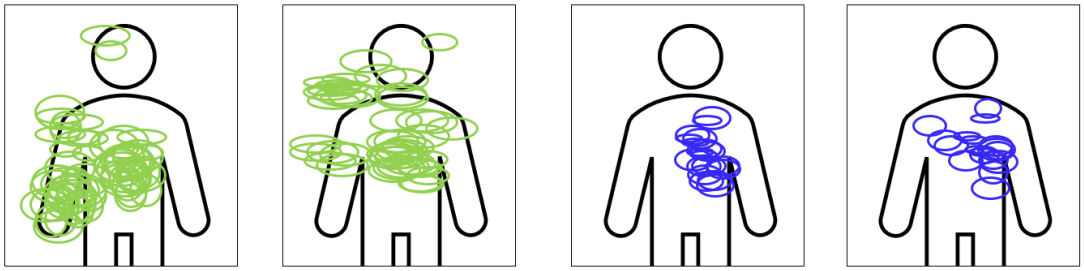


Figure 5: Means for the initial and final positions of the right hand for each sign (a and b), and also for the left hand (c and d).



Figure 6: Images of segmented hands as captured in the SIGAL database. Each image shows the initial handshape of the right hand for each sign in the dataset.

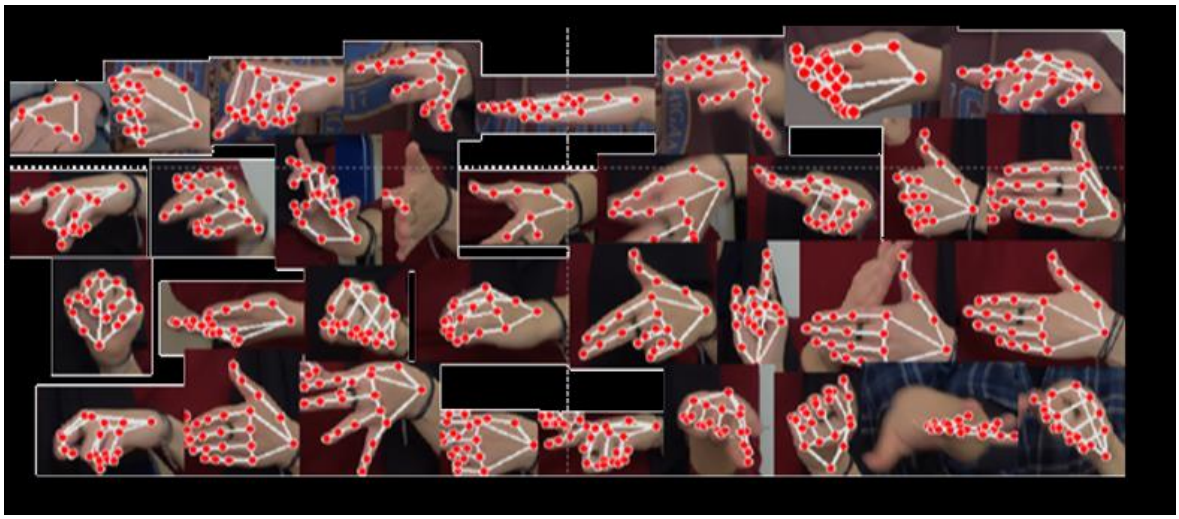


Figure 7: Images of segmented hands as captured in the SIGAL database. Each image shows the initial handshape of the left hand for the two-handed signs of the dataset.

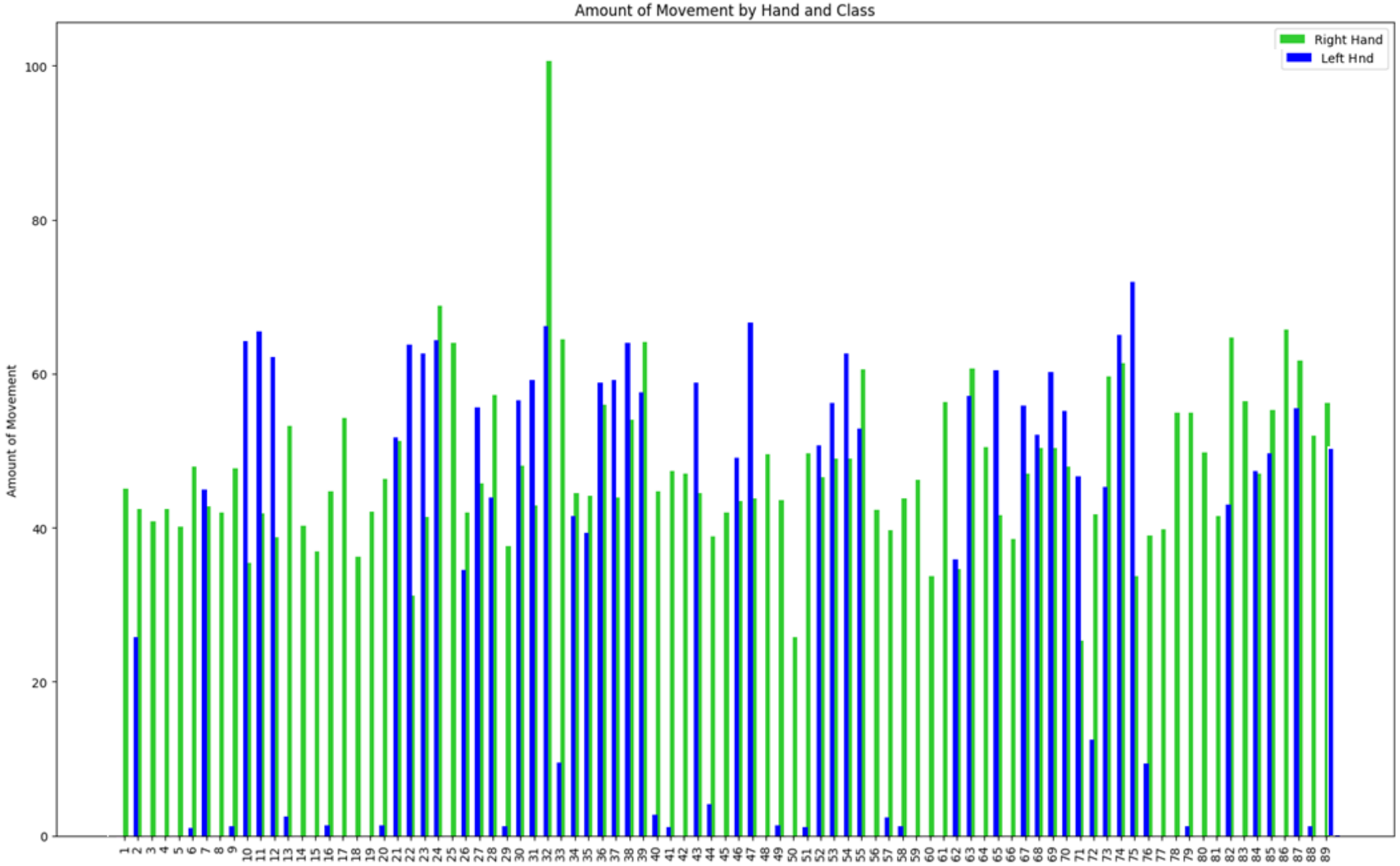


Figure 8: Amount of movement for each class of sign. Green bars show the amount of movement of the right hand, and blue bars the movement of the left hand (in two handed signs).

3. Architecture of the developed models

In this section, we present the architectures of the developed models used in our research. These models were designed to address the specific objectives and requirements of our study, as well as to facilitate a comprehensive comparison between different approaches. The following figures provide detailed architectures of each model, including the ConvLSTM, 2DCNN-LSTM, and LSTM architectures.

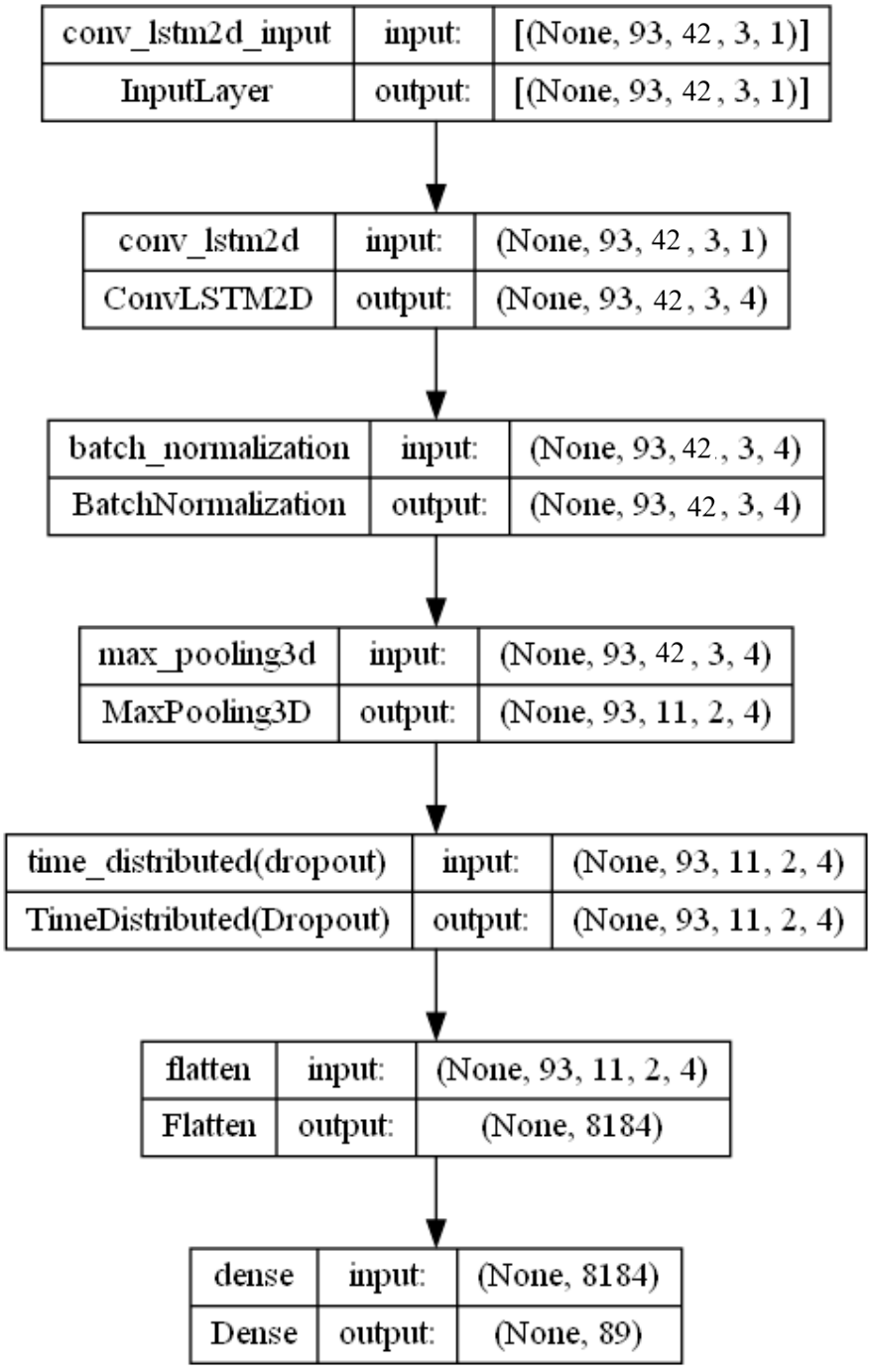


Figure 9: detailed architecture of CONV_ LSTM developed model

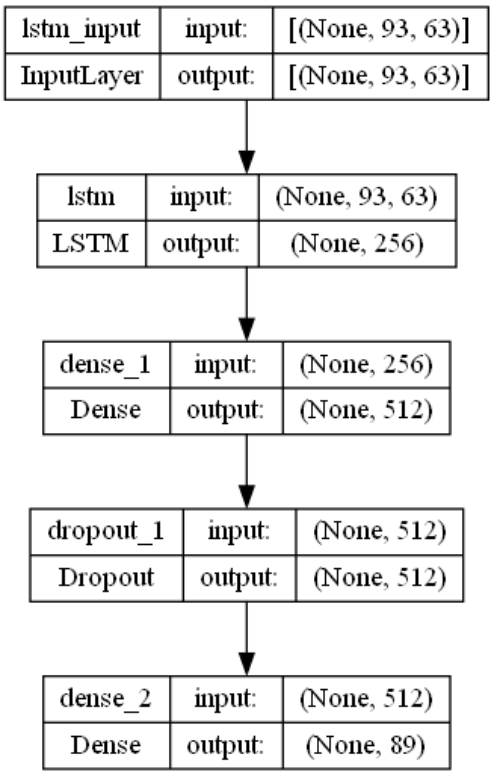


Figure 10: detailed architecture of LSTM developed model

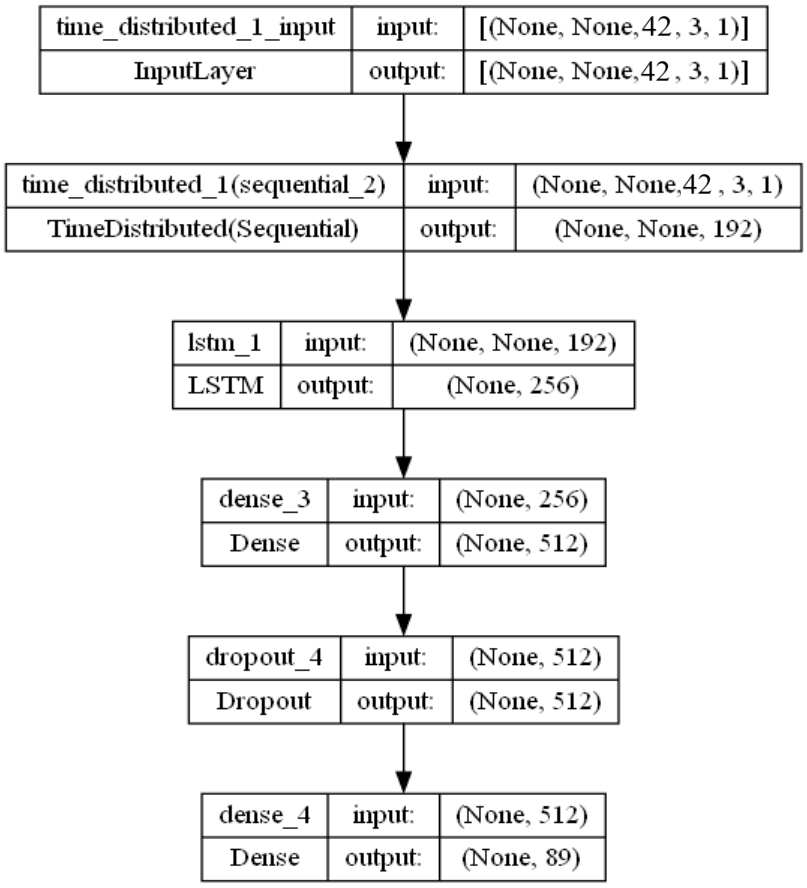


Figure 11: detailed architecture of 2DCNN-LSTM developed model

Use cases

Daily communication

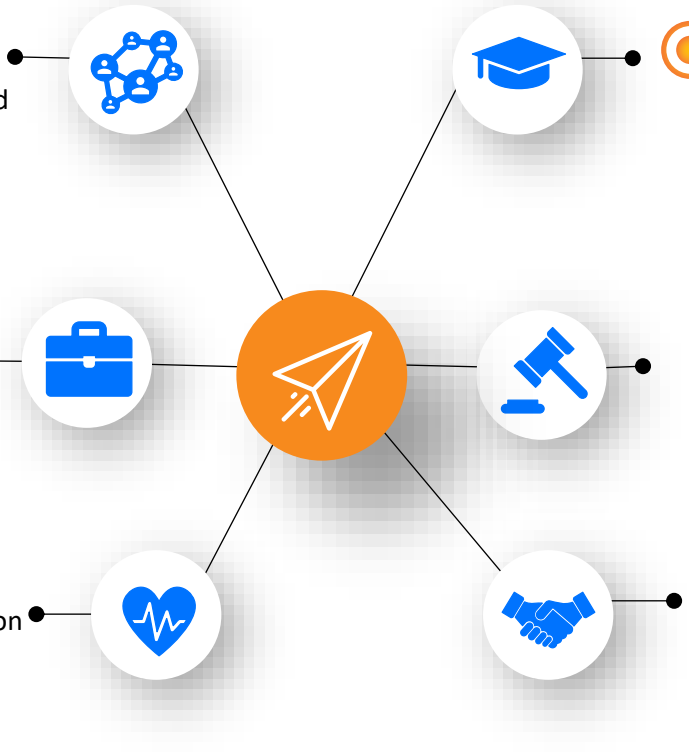
Communicate easily with your loved ones using sign language and live transcription

Career development

Create more opportunities by having a deaf–friendly job environment

Health care

Engage in a direct communication with your doctor



Education

enhance communication and coherence between deaf students and teachers during classes

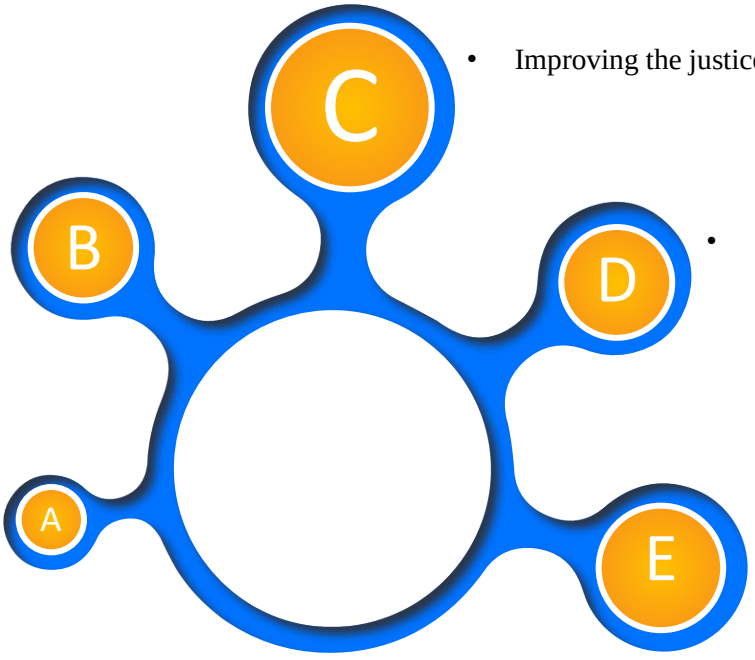
court trials

Facilitating communication between deaf individuals and legal professionals during court trials

police department

Assisting law enforcement officers in communicating with deaf individuals

Benefits of our solution



- Improving the justice system for the deaf
- Enhancing accessibility in education
- Supporting inclusive workplaces
- Encouraging a Unified Academic Sign Language
- Empowering the deaf community and remove the segregation

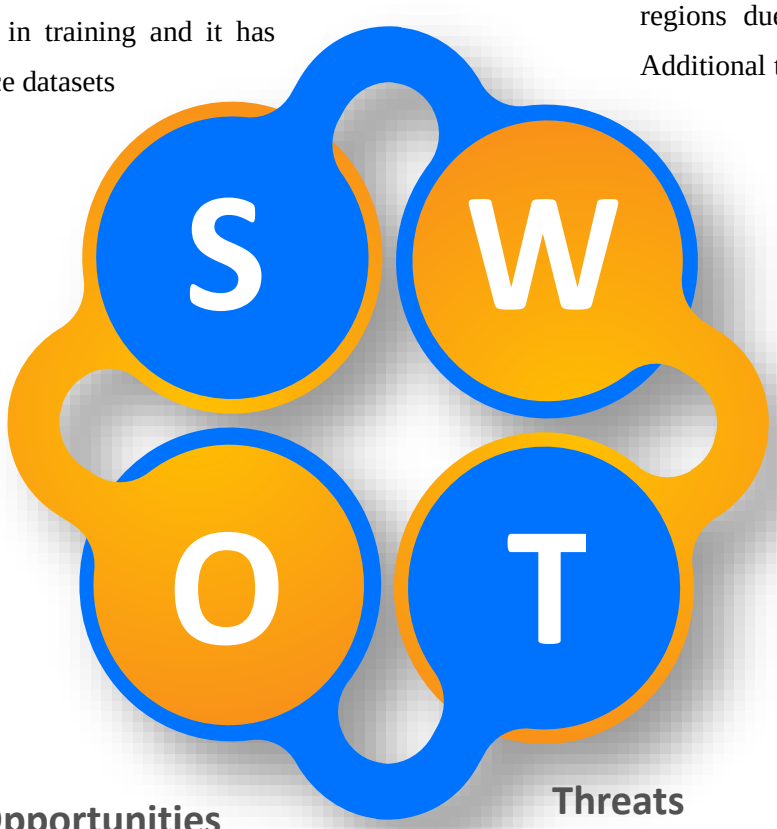
SWOT Analysis for SignAtlas

Strengths

- Our solution leverages a unique machine learning algorithm (AI) that can interpret both sign language and alphabet in real time, enabling seamless communication between deaf and hearing individuals.
- We have also developed our own proprietary dataset, which has been instrumental in training and it has better quality than old existence datasets

Weaknesses

- The performance of our application is sensitive to environmental conditions such as lighting and distance, which may affect its accuracy.
- Our solution may face challenges in interpreting signs accurately from different regions due to variations in sign language. Additional training data may be required more



Opportunities

- We are the **first to address this issue using the latest technology available**, which presents us with significant opportunities to make a positive impact.

Threats

- Using SignAtlas in courtrooms or police departments may face regulatory challenges due to legal and policy restrictions. Additional approvals and certifications may be required before it can be adopted.