People's Democratic Republic of Algeria
Ministry of Higher Education for Scientific Research
University 8 May 45 –Guelma-
Faculty of Mathematics, Computer Science and Sciences of Matter
Department of Computer Science

Master Thesis

Specialty: Computer science
Option: Computer system

Theme

A Deep Learning Framework for Bilingual Sign Language Interpretation

---

Presented by: Aya Chihaoui and Amel Bouchama

Jury Members

Chairman	Pr. Zineddine Kouahla
Supervisor	Dr. Hassina Bouressace
Examiner	Dr. Nabil Berrehouma
Representative POLE-PRO	Dr. Wahiba Abassi

June 2025

Acknowledgements

Praise be to God who has granted us the courage, strength, and patience to complete this humble work.

We would like to express our heartfelt gratitude to ***Dr. Boursas Hassina****, our supervisor, for her continuous guidance throughout the course of this project. Her valuable knowledge, insightful advice, and unwavering support greatly contributed to the achievement and improvement of our work.*

Our sincere thanks also go to the members of the jury for the honor they have bestowed upon us by accepting to evaluate this work.

We extend our deepest appreciation to all the teachers who have accompanied and supported us during our university journey. In particular, we wish to thank our Head of Department, ***Mr. KOUAHLA Zineddine****, for his dedication.*

We are also grateful to everyone who contributed, directly or indirectly, to the realization of this project—our friends, our colleagues for their help, and the entire ***class of 2025****.*

Dedications

I dedicate this humble work to:

By Amel Bouchama

*To my beloved parents, **Bouchama Amar** and **Benhamida Fatma**, the pillars of my life — for your unconditional love, endless prayers, boundless patience, and countless sacrifices. This work would not have been possible without your unwavering support.*

*To all my dear family members, my siblings **Lina**, **Bassma**, **Salma**, **Islem**, and my aunt **Zaineb**, for your comforting presence and constant encouragement. And to my fiancé, **Djaber**, for always believing in me and standing by my side.*

*To my dear friend and partner, **Chihaoui Aya**, thank you for your support, dedication, and for sharing every step of this journey with me. Your presence made this experience truly special.*

By Chihaoui Aya

*To my dearest parents, **Azedine Chihaoui** and **Abla Amari**, For your boundless tenderness, wise guidance, and unwavering moral support throughout my academic journey. Your belief in me has always been a driving force and a source of strength.*

*To my cherished family—my beloved grandfather **Khoudhir**, and my siblings **Alaa**, **Zaki**, and **Taki**, especially **Amira** Thank you for encouragement and presence during the most challenging times.*

*To my closest friends **Amani**, **Ilham**, **Hadil**, **Rahma**,**Djihan** and **Ines**, Your constant support, kindness, and uplifting spirits have meant the world to me*

*my partner and dear friend, **Amel Bouchama**, Thank you for your dedication, valuable collaboration, and the genuine friendship that made this journey both meaningful and enjoyable..*

*We would like to extend our sincere thanks to our dear friend **Ines Dergoum** for her constant presence, moral support, and continuous encouragement throughout this journey. Her kind words, her time, and her positive spirit have been a true source of inspiration and comfort. Thank you from the bottom of our hearts for your precious friendship.*

Abstract

Sign language is one of the oldest and most widely used forms of communication, primarily used by deaf or mute individuals. However, these individuals often face significant communication challenges due to the general lack of awareness about sign language and the limited number of qualified interpreters. Specifically in the Arabic world, we face limited options that are mostly based on traditional methods. To address this issue, we applied different approaches based on neural networks capable of recognizing both Arabic and English Sign Language. Our system directly analyzes a person's hands, which can be from an image or real-time input, using classification models to accurately predict the correct sign category. We employed three different approaches: Convolutional Neural Networks (CNN), CNN combined with Long Short-Term Memory networks (CNN-LSTM), and the You Only Look Once (YOLO) method, to achieve high recognition accuracy. We integrated text grammar correction and a 3D avatar for smart presentation. Our system also covers the reverse operation that ensures good communication between the speaker and the signer, where the system takes text inputs and generates the corresponding signs. Our integrated approach demonstrates that sign language recognition is both accurate and responsive in real-time scenarios, offering a valuable solution to bridge communication gaps for the hearing-impaired community.

Keywords: Arabic Sign Language, ArSL, CNN, CNN-LSTM, ASL, YOLO.

Résumé

La langue des signes est l'une des formes de communication les plus anciennes et les plus largement utilisées, principalement par les personnes sourdes ou muettes. Cependant, ces personnes sont souvent confrontées à d'importants défis de communication en raison du manque général de sensibilisation à la langue des signes et du nombre limité d'interprètes qualifiés. Plus particulièrement dans le monde arabe, nous faisons face à des options limitées qui reposent principalement sur des méthodes traditionnelles. Pour résoudre ce problème, nous avons appliqué différentes approches basées sur les réseaux de neurones, capables de reconnaître la langue des signes en arabe et en anglais. Notre système analyse directement les mains de la personne, à partir d'une image ou d'un flux en temps réel, en utilisant des modèles de classification pour prédire avec précision la catégorie du signe. Nous avons utilisé trois approches différentes : les réseaux de neurones convolutifs (CNN), les CNN combinés avec les réseaux de mémoire à long terme (CNN-LSTM), et la méthode You Only Look Once (YOLO), afin d'atteindre une haute précision de reconnaissance. Nous avons intégré une correction grammaticale du texte ainsi qu'un avatar 3D pour une présentation intelligente. Notre système couvre également l'opération inverse, garantissant une bonne communication entre le locuteur et le signant, où le système reçoit un texte en entrée et génère les signes correspondants. Notre approche intégrée démontre que la reconnaissance de la langue des signes peut être à la fois précise et réactive en temps réel, offrant une solution précieuse pour combler les lacunes de communication au sein de la communauté malentendante.

Mots-clés : Langue des signes arabe, ArSL, CNN, CNN-LSTM, ASL, YOLO.

List of content

List of Figures

List of Tables

List of abbreviations

ArSL – Arabic Sign Language Recognition

SVM – Support Vector Machines

PCA – Principal Component Analysis

KNN – K-Nearest Neighbors

CNN – Convolutional Neural Networks

ML – Machine Learning

LSTM – Long Short-Term Memory

GAN – Generative Adversarial Networks

HMM – Hidden Markov Model

ANN – Artificial Neural Network

ASL – American Sign Language

YOLO –You Only Look Once

General Introduction

Spoken language is considered the primary means of communication between individuals, with its dialects varying from one region to another and from one community to another. However, some individuals suffer from difficulties in speaking or hearing, which hinders their ability to use this method and makes communication a complex and challenging process. In this context, sign language emerges as an effective alternative that enables such individuals to express their thoughts and emotions through hand gestures without relying on speech or hearing. Sign language is a precise visual system that conveys meaning using hand movements only, making it an ideal substitute for communication in cases where spoken language is not possible.

Based on this need, the aim of this project is to develop an automated system capable of recognizing hand signs in both Arabic and English sign languages and translating them into understandable spoken-language equivalents. This work focuses on hand gestures, without relying on facial expressions or body movements, by analyzing visual clips that contain sequences of hand movements representing words or phrases. The main challenge lies in the linguistic diversity and variation in how individuals perform the signs, in addition to input quality and varying recording conditions. Our system based on different approaches for ensuring optimal results where we use different deep learning models (CNN, hybrid and YOLO). CNNs are deep learning systems that assign trainable weights to different parts of the model, enabling them to learn from images and effectively distinguish between signs. While CNNs are highly efficient in extracting spatial features from individual frames, they do not account for the temporal sequence of gestures in cases of continuous signing. Therefore, we integrated CNN with LSTM, a type of recurrent neural network capable of learning long-term dependencies. The CNN-LSTM model first extracts spatial features using CNN layers, then feeds these features into LSTM layers to capture the temporal dynamics of gesture sequences. This architecture is particularly effective for continuous sign language recognition, where the order and flow of gestures are crucial. Additionally, we used YOLO for real-time object detection, which allows for fast and accurate localization and classification of hand signs in a video stream.

This project is divided into four main chapters, each focusing on a specific technical or scientific aspect of the project:

Chapter One: Discusses general concepts related to sign language, providing an overview of its different systems and recognition methods.

Chapter Two: Presents the most prominent recent methods and studies in the field of sign language recognition, with a special focus on Arabic and English.

Chapter Three: Explains the stages of designing the proposed system for recognizing hand signs in both languages.

Chapter Four: Covers the practical implementation of the system, presents the obtained results, and includes a comparative analysis with previous work in the same field.

Chapter 1 Sign Language Recognition and Interpretation

1.1. Introduction

Effective communication is essential to ensuring inclusivity in diverse communities, with sign language interpretation regarded as one of the vital means of participation and understanding for individuals with hearing impairments. As technology develops and artificial intelligence is incorporated, researchers are constantly looking for new ways to improve the precision and effectiveness of sign language interpreting systems. The basic ideas of sign language interpretation are covered in detail in this chapter, along with methods for detection, identification, and classification. It also looks into different technologies that help enhance the interpretation process, guaranteeing more efficient and accessible communication for the hard-of-hearing and deaf communities in both languages English and Arabic.

1.2.Sign Language

Sign language (SL) is regarded as the most structured and ordered of the different gesture categories in the communicative hand/arm gesture taxonomies. An essential communication tool for the deaf and hearing challenged community is sign language. Hearing-impaired people communicate by employing signs in visual space rather than spoken words and sound patterns. SL deals with non-manual signals that convey semantic meaning through a variety of body postures and face expressions in addition to hand and arm gestures [WK21].

Pattern matching, computer vision, natural language processing, and linguistics are all involved in the joint study field of sign language recognition. Its goal is to develop a variety of techniques and algorithms to recognize and interpret signs that have already been produced. Systems based on Human Computer Interaction (HCI) that recognize sign language are intended to facilitate effective and interesting communication. These systems use data

collection, SL technology, SL testing, and SL linguistics in a multidisciplinary manner. To help hearing-impaired people learn new ideas and information and manage their emotional behavior, such a system can be implemented in public services like hotels, railroads, resorts, banks, offices, etc. [GAS09].

1.2.1. Symbols in Sign Language

In the 1970s, linguistic research on sign language began [BKA15]. It comprises linguistic information, such as various letters and symbols. All of the sign parameters, such as hand forms, movement, location, and palm orientation, can be represented by sign language symbols. Figure 1-1 displays the classification of SL symbols. Single-handed and double-handed signs are the two categories into which SL symbols fall. These signs are further classified into two categories: one-handed and two-handed signs.

A. One-Handed Signs

One dominant hand is used to indicate one-handed signs. Any static gesture or a motion-based gesture can be used to depict it.

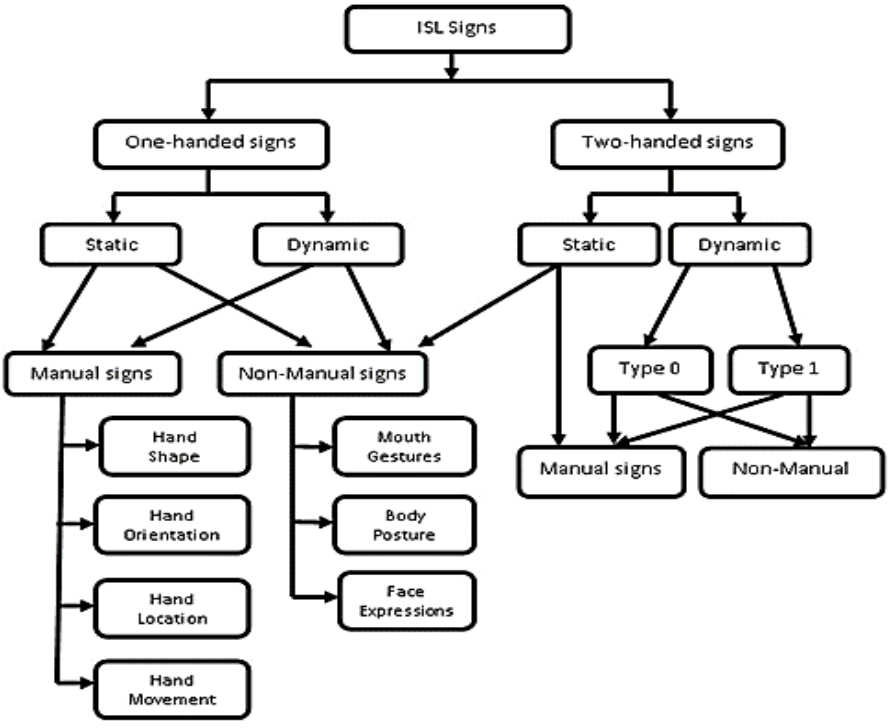

Figure 1-1 A hierarchical classification of signs. [DK13]

B. Two-Handed Signs

When signing, both the dominant and non-dominant signs are utilized to signify two-handed signs. Type 0 and type 1 signs are further classifications for these. Figure 1-2

illustrates that both hands are active in the Type 0 sign, while Figure 1-3 shows that the dominant hand is more active than the nondominant hand in the Type 1 sign [WK21].

Figure 1-2 Two handed type 0 sign (both the hands are active). [AA23]

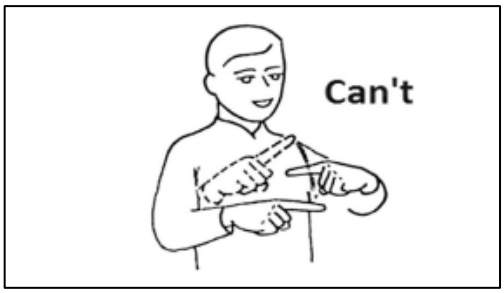

Figure 1-3 Two handed type 1 sign (only dominant hand is active). [AA23]

Both manual and non-manual components make up sign language; Body postures, mouth gestures, and facial expressions are employed in non-manual signs, while just the hands are used in manual signs. Figure 1-4 display static manual signs that are single-handed and non-manual, respectively.

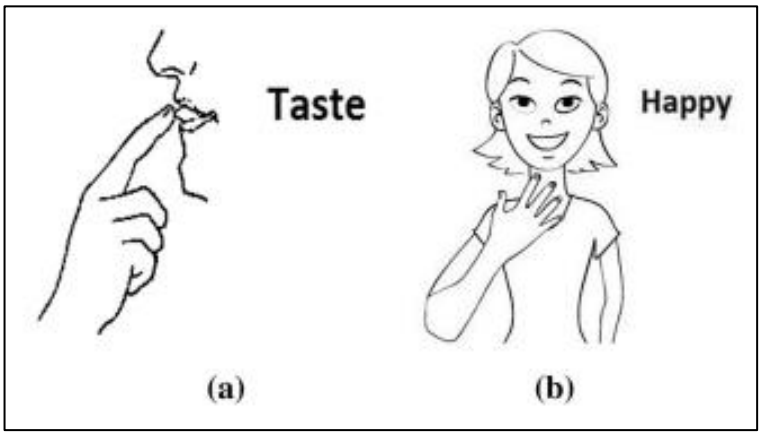

Figure 1-4 A sample of static manual signs. (a) One-handed static manual sign, (b) Non-manual sign [WK21].

Figure 1-5 illustrates the five components of signs in sign languages, which include hand movement, hand shape, hand location, hand orientation, and non-manual features. These components collectively define the structure and meaning of signs, playing a crucial role in sign language communication. [AUD16].

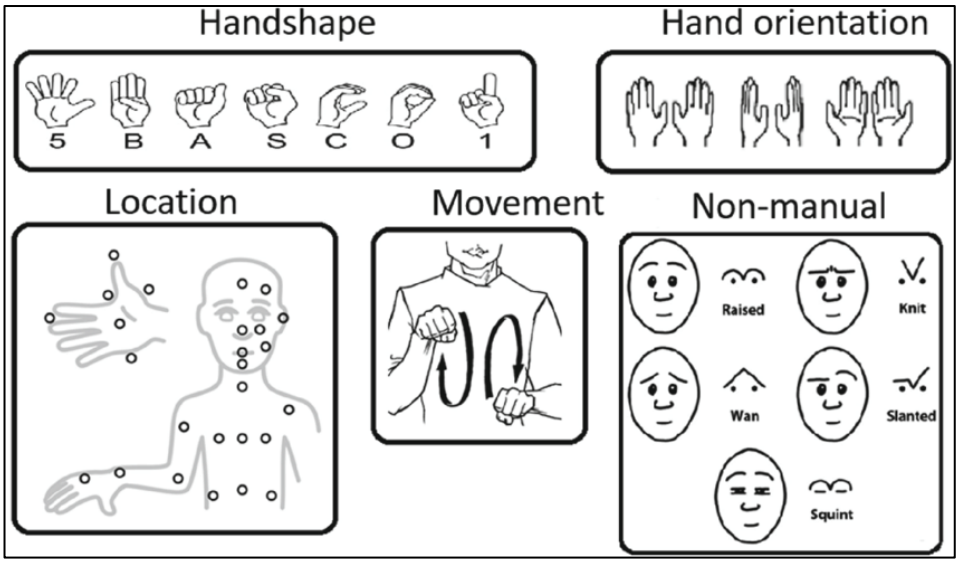

Figure 1-5 Components of Signs in Sign Languages. [PBK20]

1.3. Phases of Sign Language Processing: From Detection to Translation

The following terms are phases that used to recognize sign language, and each phase advances the process of comprehending and converting sign language into speech or writing. The phases and their descriptions are as follows:

1.3.1. Sign Language Detection

The system's objective in this initial phase is to determine whether or not the image or video contains sign language content. Stated differently, it assesses whether the content is relevant to sign language without requiring knowledge of the particular signs being used [BC19]. (See Figure 1-6).

Figure 1-6 Example of detection of the word 'الله ' in Arabic sign gesture [HL23].

1.3.2. Sign Language Identification

The next phase after determining whether sign language is present is to determine whether particular sign language is being used. American Sign Language (ASL), British Sign Language (BSL), Arabic Sign Language (ArSL), or any other type of sign language can be recognized by the system [DK10].

1.3.3. Sign Language Recognition

This next stage involves analyzing the gestures themselves and translating them into readable text or speech. It includes interpreting hand movements, finger positions, facial expressions, and body postures, then converting them into understandable text or speech [CH11]. (See **Figure 1-7**).

Figure 1-7 Example of recognition of the word ' الله ' in Arabic sign gesture [HL23].

1.3.4. Sign Language Translation

Sign language translation refers to the process of converting linguistic content between sign language and spoken or written language. As shown in Figure 1-8, this field includes systems that translate natural language to sign language and vice versa. These systems often rely on earlier steps like gesture detection and recognition, handled through computer vision, specialized sensors, or wearable devices. Translated output may be presented using avatars or animations to visually render signs, as discussed in the next section. Additionally, various applications support communication and education for the deaf community by integrating translation capabilities.

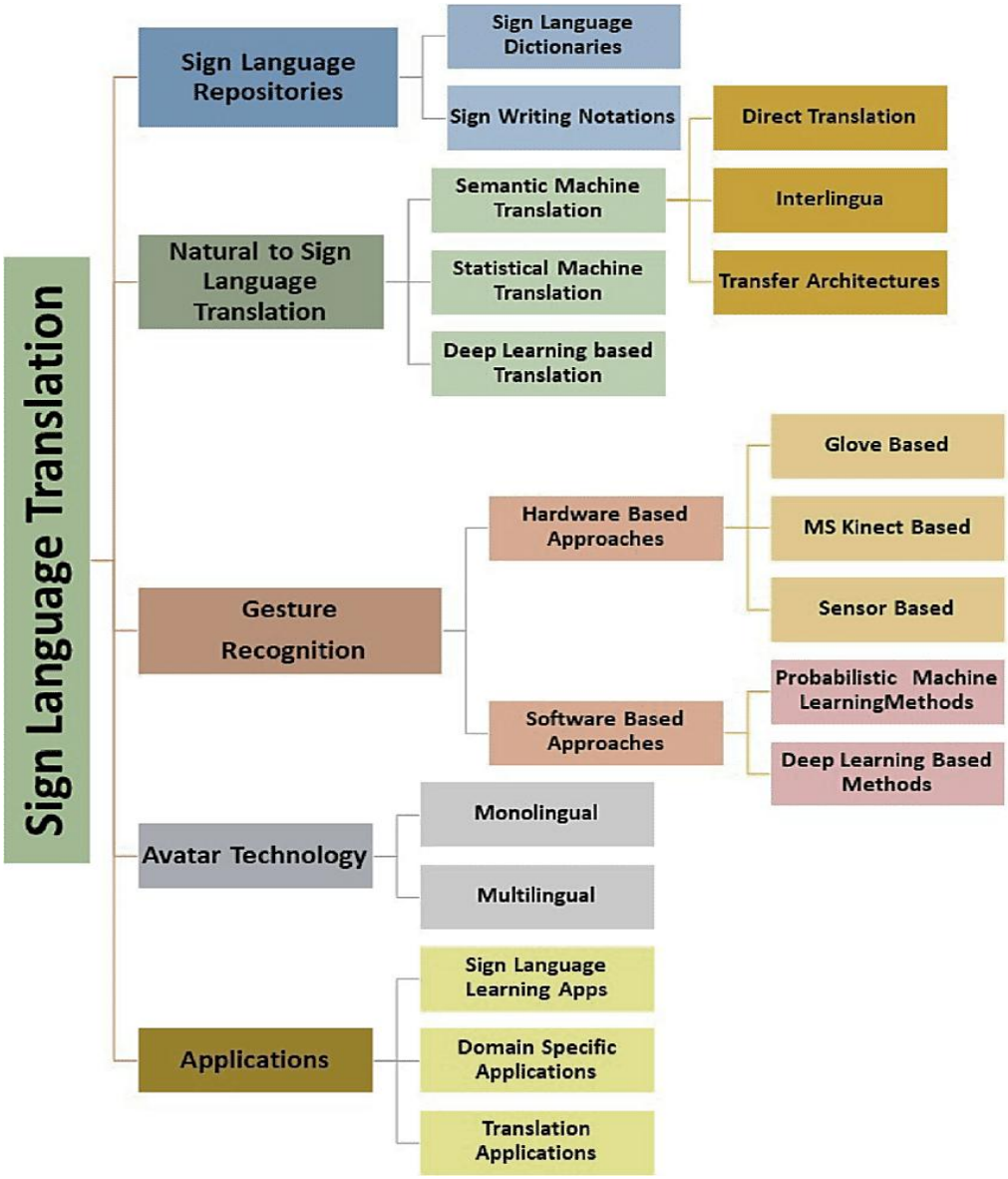

Figure 1-8 Taxonomy of sign language translation system components [FRS21].

Datasets with signs and the text that goes with them are used to train sign language translation algorithms. A letter, number, or word can all be represented by a sign. The sign-to-speech process is shown in **Figure 1-9**, where videos or pictures of signs are taken, pre-processed, and categorized to produce text that is subsequently translated into voice.

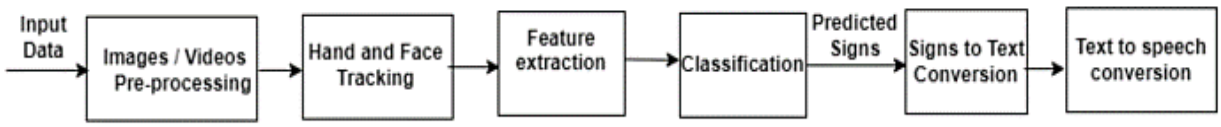

Figure 1-9 Sign language to speech translation process. [NAJ25]

The opposite procedure is depicted in **Figure 1-10**, where speech is first transformed into text using programs such as Google's Speech API, and then processed and translated into matching image or avatar representations.

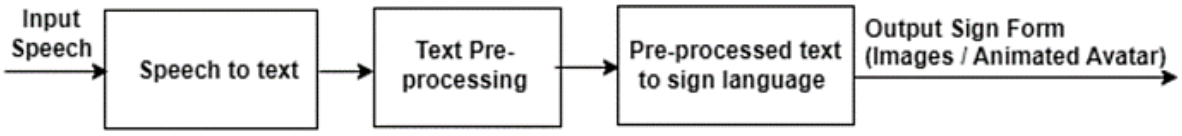

Figure 1-10 Speech to sign language translation process. [NAJ25]

1.3.5. Sign Language Interpretation

Unlike translation, which typically involves converting words from one language to another in a direct and often literal manner, interpretation of sign language encompasses a deeper, more nuanced process. It involves understanding not just the lexical content of the signs, but also their contextual meaning, cultural significance, and emotional tone. Sign language interpretation requires real-time cognitive processing to convey intent, mood, and non-verbal cues such as facial expressions and body language. This distinction is crucial in multilingual and multicultural settings, where literal translation may lead to miscommunication, but skilled interpretation ensures that the true meaning and purpose of the message are faithfully conveyed.

1.4. Features of Sign Language

Like to spoken languages, there are numerous distinct sign languages that differ from one place to another; sign language is not a universal language. American Sign Language (ASL) and Arabic Sign Language (ArSL), for instance, each have distinct signals and grammatical norms. Every sign language has a unique system for representing words and concepts, despite certain commonalities. Sentence structure, facial emotions, and spatial location for communication are some examples of these variations. To make sign language translation systems more precise and effective, it is imperative to comprehend these variances. [ABF05]

Every country has its own distinct syntax and semantics for sign languages. The alphabet systems of the main sign languages American (ASL) and Arabic as well as their variations are displayed in **Figures 1-11, 1-12.**

Figure 1-11 Arabic Alphabet in Sign Language. [SAN20]

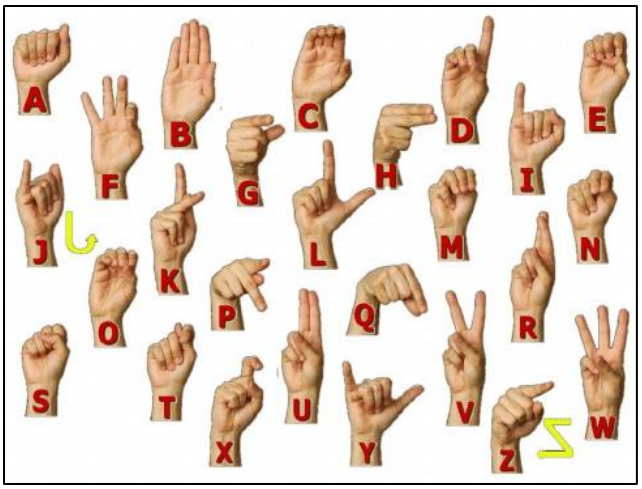

Figure 1-12 Figure 1 12 Signs of the ASL sign language. [ASL]

1.4.1. American Sign Language (ASL)

ASL, or American Sign Language, is a visual language that is widely used in portions of Canada and the United States. It uses hand signals, body language, and facial expressions to convey meaning and has its own grammar and syntax distinct American. One of the sign languages that has been studied the most is ASL, which is crucial for deaf communication and education. Accessibility for the deaf community has greatly increased because to technological developments in ASL recognition. [EKH06]

1.4.2. Arabic Sign Language (ArSL)

The main means of communication for the deaf community in Arab nations is Arabic Sign Language, or ArSL. It uses hand gestures, facial emotions, and body movements and has separate grammatical patterns from spoken Arabic. Despite regional variations, ArSL has some characteristics in common. There isn't as much research on ArSL as there is on ASL. To improve accessibility and communication, there is now more interest in creating ArSL recognition systems thanks to developments in computer vision and artificial intelligence [MDL14].

1.5. Sign Language Recognition Methods

A gesture is any significant physical movement of the hands, arms, fingers, or other body parts [GW] intended to transmit meaning or information for interaction with the environment

[MA07]. For gesture recognition to work, hand movements must be accurately interpreted as meaningful commands. There are two popular methods for human-computer interaction (HCI) interpretation systems [MS09].

1.5.1. Glove-Based Methods

These techniques use optical or mechanical sensors. Connected to a glove that uses electrical signals generated from finger flexions to ascertain hand position [MG01]. With this approach, one or more data-glove instruments that have various measurements for the hand's joint angles and degree of freedom (DOF)—which include information on the hand's location and orientation utilized for hand tracking—collect the data [LJ99]. The naturalness of user-computer contact will be hampered by this method's requirement that a glove be worn and a cumbersome gadget with numerous cables attached to the computer [MA07].

A. **CyberGlove:** The Stanford University-developed CyberGlove is a glove with eighteen or twenty-two sensors that track wrist and finger movements. It is extensively utilized in animation, virtual prototyping, and gesture recognition [MH13].
B. **The Humanglove** is an Italian glove that tracks finger and wrist movements using 20 sensors. It is perfect for robotics and simulations because of its calibration system, which shows a virtual hand [CB07], [DH16].
C. **5DT Data Glove:** Measures finger bending using optical-fiber flex sensors. It is appropriate for virtual reality and sign language interpretation because it supports wireless applications and medical settings [CF03], [CW02].
D. **Pinch Glove:** Does not require calibration; uses electrical contacts on the fingertip to detect gestures [BW01]. With over 1,000 gestures recognized, it is a valuable tool for human-computer interaction [LV99].
E. **Didjiglove:** A cutting-edge glove with capacitive bend sensors made especially for 3D modeling and animation programs like Maya and 3DS Max [CF03].
F. **The StrinGlove** is a Japanese glove that measures wrist motion and finger joint angles using 24 inductive sensors and 9 magnetic sensors. It has a modular sensor replacement design and is made of washable cloth [KT04].

Figure 1-15 presents examples of Glove-Based Methods, showcasing various glove systems used for hand motion sensing and gesture recognition. Below is a breakdown of some key glove-based systems:

Figure 1-13 Examples of Glove systems: (a) CyberGlove, (b) Humanglove, (c) Pinch Glove (Image courtesy of Fakespace Systems), (d) Didjiglove, (e) Fingernail Sensor [Ghu14], (f) AcceleGlove [Her02], (g) Upper limb garment prototype [Par06], and (h) Sensing glove [LS T05].

1.5.2. Vision-Based Approaches

These methods are predicated on how an individual perceives their surroundings. These techniques often involve utilizing a camera or cameras to capture the input image [SM11]. The gestures should be chosen based on their pertinent meaning in order to build the database for the gesture system, and each gesture may have multiple samples [HM10] to improve system accuracy. (See **Figure 1-16**)

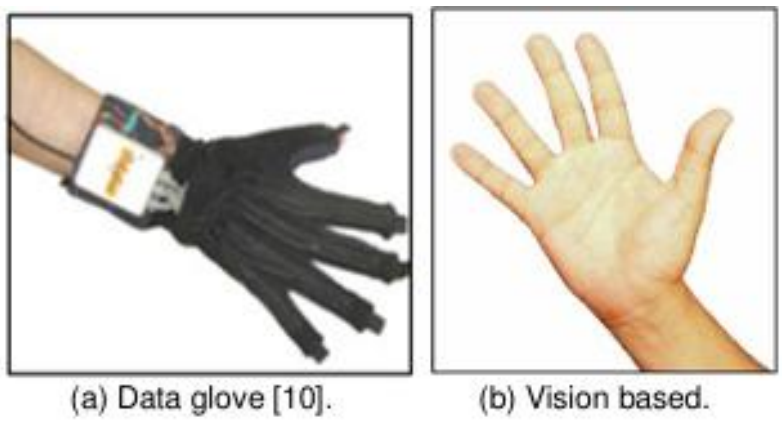

Figure 1-14 Examples of data glove and vision based. [IKK12]

There are two types of vision-based hand gesture recognition techniques: appearance-based techniques and 3D model-based techniques.

A. Appearance Based Approaches

These methods compare features taken from the input camera or video input with features taken from the visual appearance of the input image model of the hand.

B. 3D Model Based Methods

These methods rely on the hand's kinematic degrees of freedom. These techniques attempt to create 2D projections from 3D hand models and infer various hand properties from the input image, such as palm position and joint angles [GAS09].

1.6. Machine Learning Techniques for Sign Language Processing

Both supervised and unsupervised machine learning are used in sign language recognition (SLR). Although supervised learning is expensive and time-consuming, it uses labeled datasets for categorization.

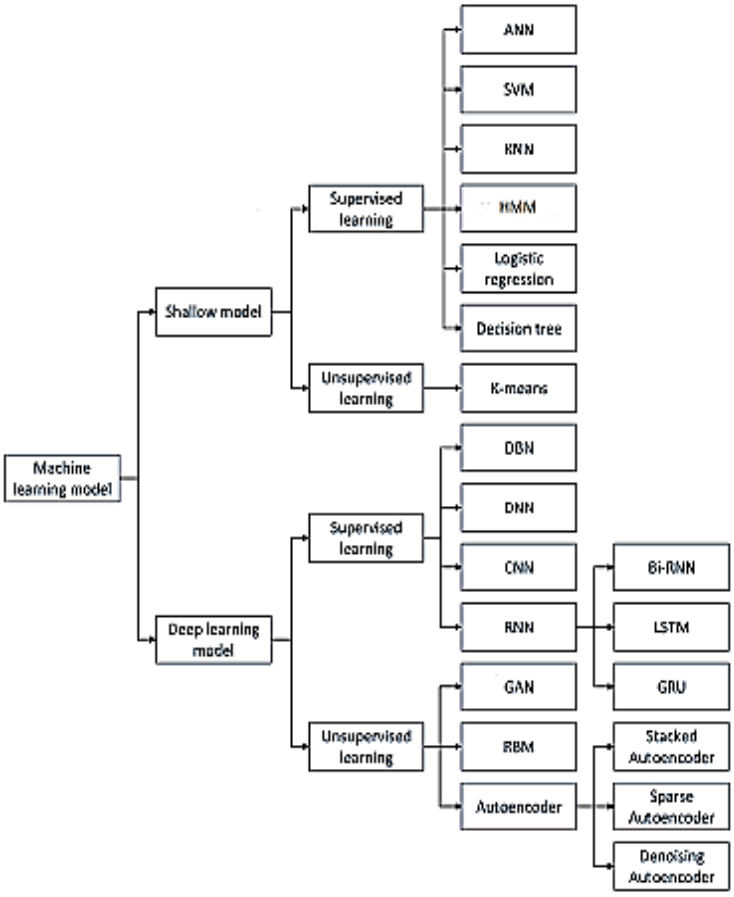

Figure 1-15 Taxonomy of machine learning models [LL19].

Unsupervised learning makes training simpler but frequently less accurate by identifying patterns in unlabeled data. Generally speaking, supervised models do better in recognition [GE19]. Common machine learning techniques used in SLR are depicted in **Figure 1-17.**

1.6.1. Shallow Models

Artificial neural networks (ANNs), support vector machines (SVMs), K-nearest neighbors (KNN), naïve Bayes, logistic regression (LR), decision trees, clustering, and hybrid approaches are examples of traditional machine learning models for SLR. Over time, these models have been thoroughly examined and improved. They are employed to solve real-world issues in SLR systems in addition to increasing recognition accuracy.

A. Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are inspired by the structure of the human brain. They consist of an input layer, multiple hidden layers, and an output layer, with fully connected units between adjacent layers. ANNs have a strong ability to model nonlinear relationships, making them effective for recognizing complex sign language gestures. However, due to their large number of parameters and deep architecture, training ANNs requires significant computational resources and time.

B. K-Nearest Neighbor (KNN)

Sign language motions are categorized by the KNN algorithm according to how similar they are to nearby examples. The manifold hypothesis states that a sign is likely to belong to the same class if the majority of its closest neighbors do. The model's performance is greatly impacted by the choice of k. A smaller k indicates a more intricate model with a greater chance of over fitting. Large $k=>A$ simpler model with a reduced capacity to differentiate between comparable actions. Because of its ease of use and capacity to categorize new signals using pre-existing examples, KNN is useful for sign language recognition; yet, when working with big datasets, it can be computationally costly.

C. Hidden Markov Model (HMM)

Statistical models known as Hidden Markov Models (HMMs) are used to depict systems in which observable outputs can be utilized to infer underlying states that are not readily observable. (See Figure 1-18). They have been extensively used in many domains, including gesture detection, and are especially good at simulating temporal sequences. To capture the sequential nature of sign language motions, HMMs are used to simulate the temporal dynamics of hand gestures in the context of sign language recognition. Accurate sign language recognition and interpretation are made possible by HMMs' ability to deduce the most likely sequence of underlying states from the series of observed motions.

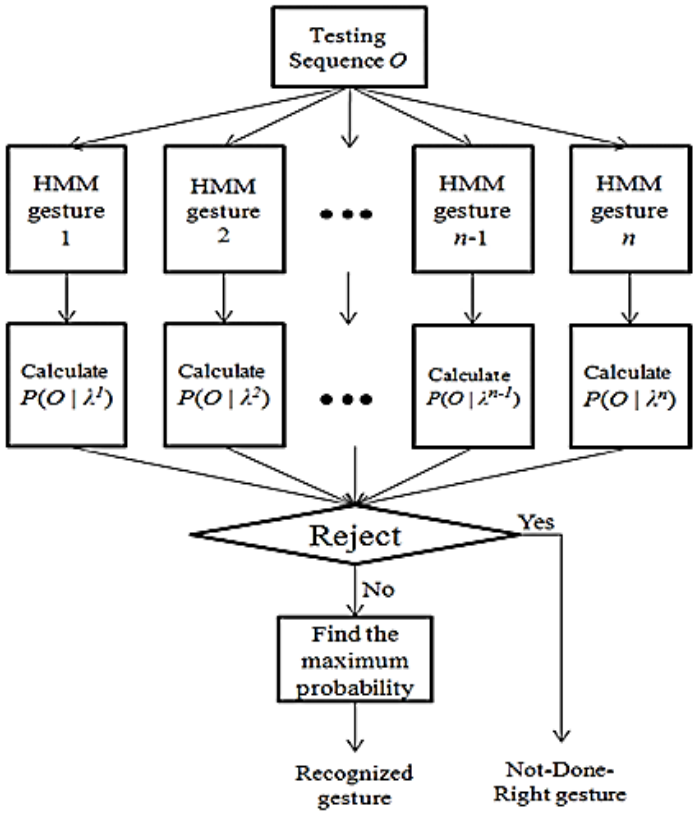

Figure 1-16 Hand gesture recognition architecture based on an HMM model. [LCJ14]

D. Support Vector Machines (SVM)

The goal of a Support Vector Machine (SVM) classifier is to determine the best border between classes. The structural risk minimization (SRM) concept, which SVM adheres to in contrast to other classifiers, improves its capacity for generalization in machine learning applications. With their high efficiency in differentiating between different movements, SVMs have been successfully used in hand gesture identification. They achieve reliable and precise categorization by locating a hyper plane that maximizes the margin between various gesture classifications. An example of an SVM model architecture for hand gesture recognition is shown in the **Figure 1-19**.

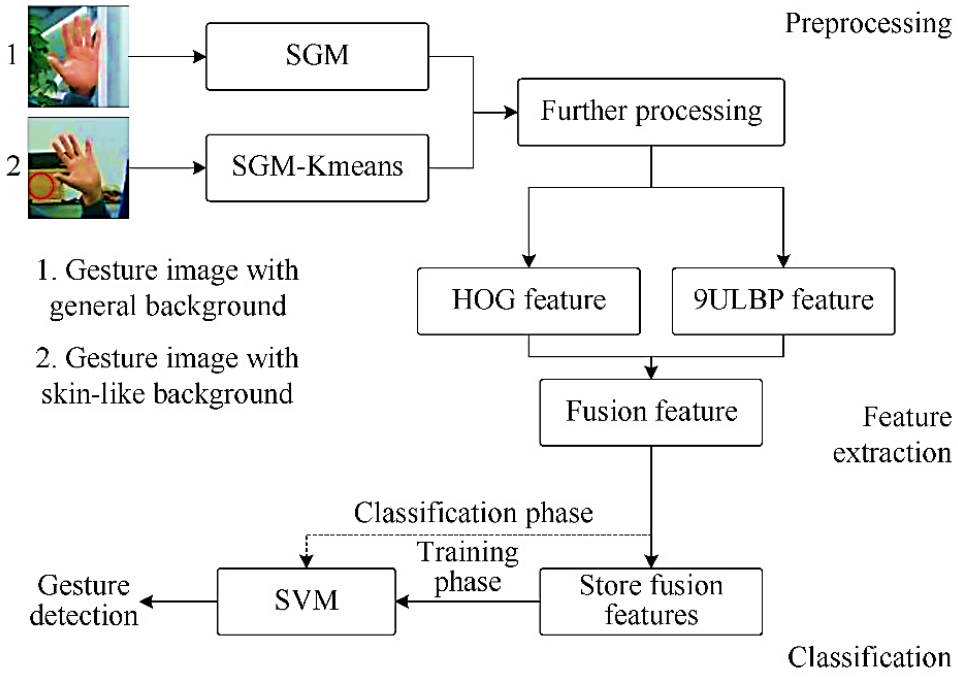

Figure 1-17 The architecture of hand gesture recognition using a SVM model. [LHL22]

1.6.2. Deep Learning Approaches

In sign language recognition (SLR), deep learning is essential because it can automatically extract significant features from unprocessed input (such pictures or videos) without the need for human feature extraction. In SLR, there are two primary types of deep learning models:

A. Supervised models

In sign language recognition (SLR), supervised learning plays a key role by training models on labeled datasets to recognize specific gestures or signs. These models learn to associate input data (such as images or videos) with their corresponding sign language labels, for an accurate classification results. Among the most widely used supervised deep learning models in SLR are CNNs, RNNs, and LSTMs.

a) Convolutional Neural Networks (CNNs)

CNNs are effective in sign language recognition (SLR) as they mimic the human visual system (HVS). They consist of convolutional layers (for feature extraction) and pooling layers (for generalization) [RAZ14], [KRI12]. Since CNNs process 2D data, sign language images or video frames are converted into matrices for recognition, making them highly suitable for SLR tasks. Figure 1-20 displays a CNN-based hand gesture recognition model. The procedure entails:

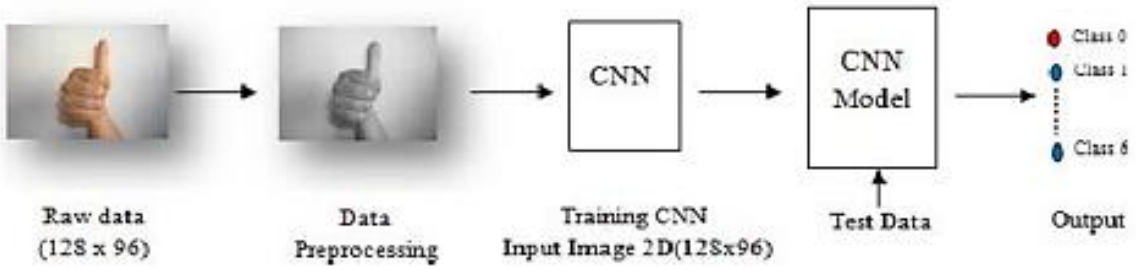

Figure 1-18 Hand gesture recognition using an adapted convolutional neural network [ACT18]

- **Raw Data:** Taking a picture of a hand gesture.
- **Preprocessing:** Using normalization and filtering to improve the quality of images.
- **CNN training:** Key feature extraction for categorization.
- **Testing:** Assessing the accuracy of the model with fresh data.
- Putting the gesture into pre-established categories is the output.
-

b) Recurrent Neural Networks (RNNs)

Because they are made for sequential input, recurrent neural networks (RNNs) are helpful for sign language recognition (SLR), particularly in applications that use videos [LAW97], [GRA13], and [GJ14]. RNNs take into account both recent and historical frames in order to collect contextual information because sign language depends on temporal dependencies. Nevertheless, conventional RNNs have trouble with vanishing or bursting gradients, which restricts how lengthy sequences they can process. Advanced versions such as Gated Recurrent Units (GRU) [CHU14] and Long Short-Term Memory (LSTM) [GJ14] are frequently employed in SLR to get around this.

c) Long Short-Term Memory Networks (LSTMs)

Because they can capture temporal relationships in gesture sequences, LSTM networks—a specific type of Recurrent Neural Networks (RNNs) —are very successful at recognizing sign language. Because LSTMs handle vanishing and exploding gradient concerns, they are better suited to managing lengthy and intricate gesture patterns than typical RNNs. LSTM's main benefits in sign language recognition are as follows:

- **Processing Time Series Data:** LSTM accurately recognizes dynamic hand movements by effectively tracking the temporal progression of gestures.

- **Modeling Long-Term Dependencies:** LSTM improves gesture classification by retaining pertinent information over long sequences thanks to its forget and input gates.
- **Contextual Gesture Understanding:** LSTM makes it easier to recognize continuous sign language phrases by maintaining context across gestures. [WU24]

Figure 1-21 uses the first LSTM layer [TB07] for a time series of features extracted from gesture data to exploit the long-term dependencies and encode them into a sequence the same as the length of the input gesture with the specific feature size.

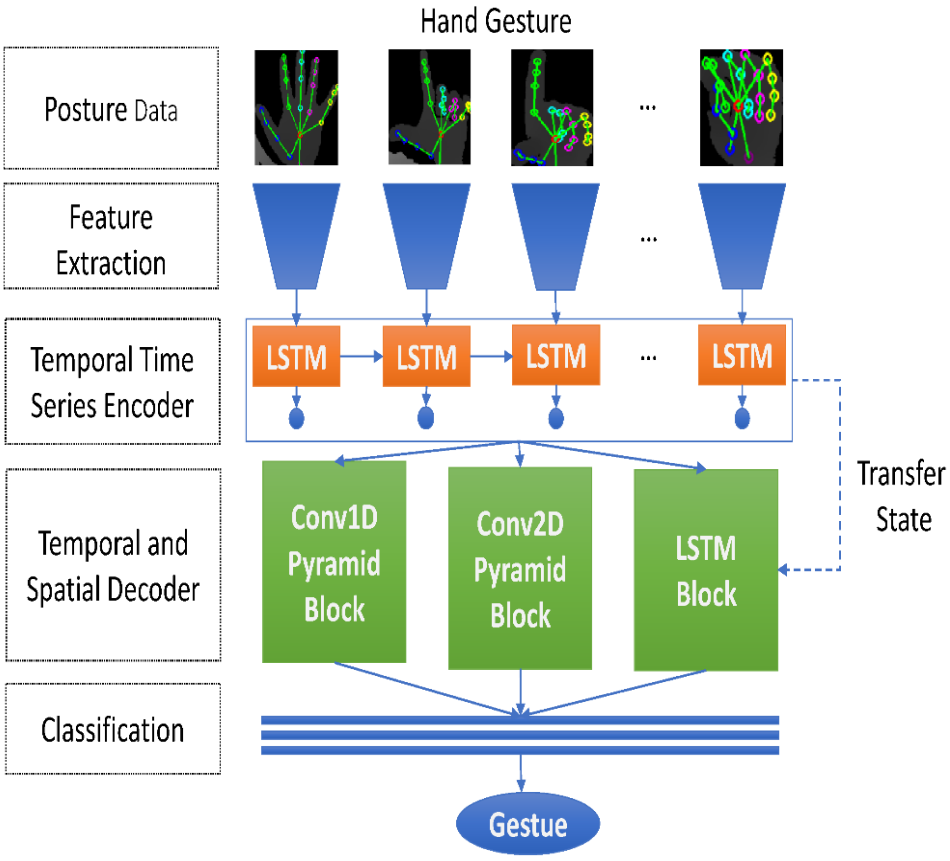

Figure 1-19 Multi-level feature LSTM architecture [DKYL20].

B. Unsupervised models

In the field of SL, the unsupervised models are not be developed much compared supervised models, where the most well-known model is Generative Adversarial Networks (GANs). GANs improve sign language datasets and increase the reliability of sign recognition models. They are also employed in data augmentation, which makes it possible to train models more effectively even when there is a shortage of real-world data. [SD25]

1.7. Applications for sign language recognition

Technology that recognizes sign language is essential for improving deaf people's learning, accessibility, and communication [DSD08]. Here are a few important uses:

1.7.1. System for Translating Sign Language

This application translates sign language into text or spoken language in real time using hand motion recognition **Figure 1-22**. It is an essential tool for promoting communication between those who are hard of hearing or deaf and those who are not familiar with sign language. Wearable sensor arrays and wireless circuitry for sensitivity and fast reaction are examples of advanced systems [ZCL20].

Figure 1-20 Example for sign language translation system [ZCL20].

1.7.2. System for training sign language

This is skilled in tracking and deciphering user hand gestures. These tools help students improve their sign language skills by giving them immediate feedback **Figure 1-23**. Designed to be interactive, they provide comprehensive courses that cover a wide range of grammar and vocabulary, along with an in-depth exploration of the Deaf community's cultural fabric, to enable a comprehensive and interesting educational experience. [MOT18]

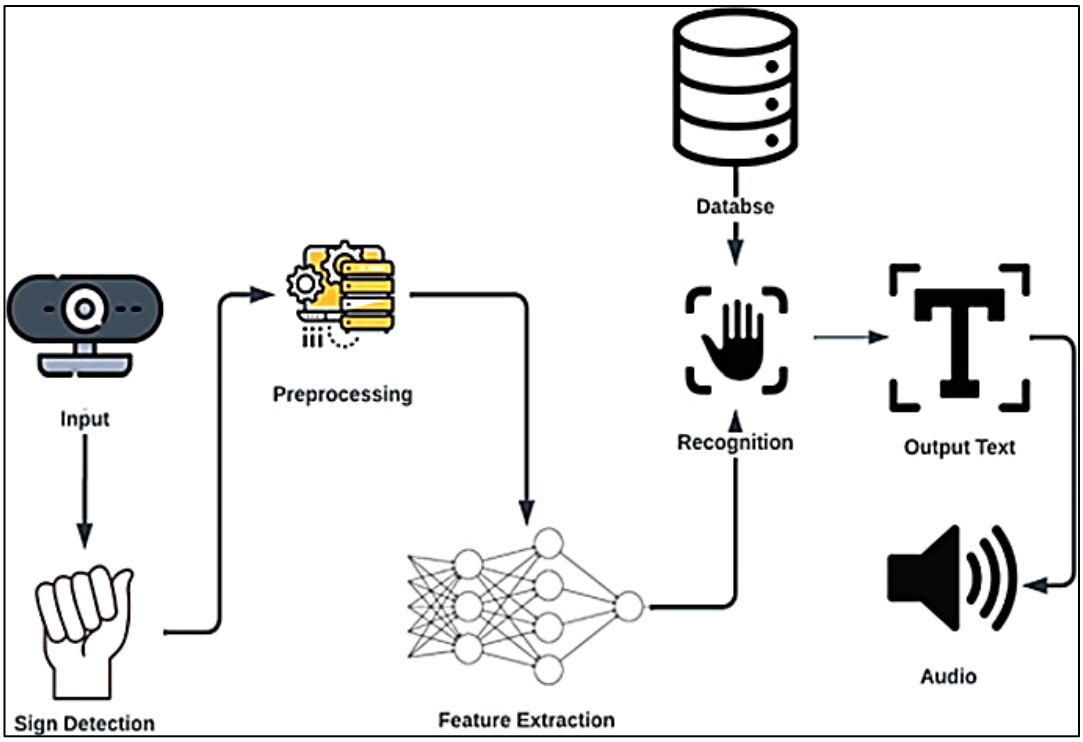

Figure 1-21 Overview of sign language training system [MOT18].

1.8. Open challenges in sign language Translation

Every unique facet of sign language translation covered in this paper has a number of intriguing difficulties. This section outlines some of the exciting avenues for future study in each of the several facets of sign language translation as indicated by its taxonomy [FR21].

1.8.1. Sign language repositories

Despite they existence, sign language dictionaries need to be improved in three crucial areas. Since many developing nations still lack structured systems and only a small number of sign languages, like ASL, have standardized dictionaries, standardization is crucial. With recent developments like Wehrmeyer's notation system [WEH19] supporting phonetic representation and lexicography, sign writing notations should be incorporated to facilitate avatar-based gesture production. Furthermore, a system for adding new terms is required to guarantee that new terms - especially in the field of technology - are included in an organized manner.

1.8.2. Recognition of Gestures

There are various obstacles in the way of gesture recognition. The nature of gestures presents challenges since it is still difficult to distinguish between single- and double-handed

actions, gestures, and non-manual characteristics, particularly when the gestures are identical (See **Figure 1-24**). Variability is increased by subjects making gestures because of variations in skill, speed, hand size, and other elements. Cost and calibration problems plague hardware-based methods, which need to be affordable and flexible enough to accommodate users' anatomical variations. Additionally, sensor placement affects cost and accuracy, resulting in effectiveness trade-offs.

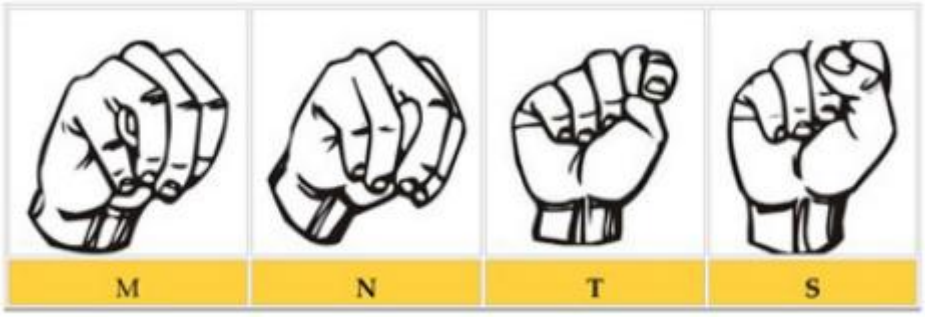

Figure 1-22 Gestures for different ASL alphabets with almost similar hand shapes [AZS18]

1.8.3. Technology for Avatars

Avatar technology needs to be improved in a few ways. With variations for kids and adults, realistic avatars should closely resemble people. They should also improve non-manual clues to improve semantic correctness. Since avatars must connect several sign executions from motion-capture libraries, smoother gesture transitions are required. Another difficulty is gesture modulation since it can be challenging to record motion data when altering gestures to convey adjectives or adverbs (for example, "a bumpy plane ride" in ASL). Last but not least, the majority of motion-capture data is proprietary, which restricts the development of avatar technology due to the lack of publicly available datasets.

1.8.4. General Challenges and Directions

For sign language translation systems to progress, a number of issues need to be resolved. Due to the lack of written sign language, the corpora for sign language are much smaller than those for spoken languages, which makes the lack of datasets still a serious problem. Deaf people and sign language specialists must be involved in the creation of more databases. The deaf community can be involved in crowdsourcing with editorial control to create parallel sign language corpora, provide gestures for new words, and assess avatar and translation

systems. In order to enable cross-language sign translation and promote communication amongst deaf groups around the world, multilingual translation applications are also required.

1.9. Future Research Directions

There are currently no systems tackling big vocabulary recognition based on language dictionaries, and most sign language recognition research focuses on letters, numbers, or tiny word and sentence groups. There is currently little research on internet services for the deaf and mute, and no solutions have been proposed to help with online banking or shopping. Furthermore, despite their significance, social networking sites designed specifically for deaf-mute people have not been covered. Finally, there aren't many suggestions for self-education resources, like programs that let deaf people read books, articles, and other materials online.

1.10. Conclusion

Making the world more inclusive for people with hearing disabilities requires recognizing and developing sign language interpretation. We can empower deaf people and bridge the gap between sign language users and the general public by promoting accessibility and communication. The next chapter will discuss current research on Arabic and English sign language interpretation, including initiatives to improve understanding and communication.

Chapter 2 State-of-the-Art

2.1.Introduction

Recent advancements in computer vision and artificial intelligence have sparked renewed interest in sign language recognition, particularly for ArSL. However, the development of reliable recognition models is hindered by a lack of publicly available datasets. This chapter reviews existing research on sign language recognition, focusing on both Arabic and English languages, and discusses the methodologies used, challenges faced, and comparisons between the two languages.

2.2.Basic methodology

This section outlines the application of vision-based techniques for identifying motions in Sign Language, where sign inputs are extracted from either image or video frames. **Figure 2-1** illustrates the fundamental steps in the recognition process, detailing the transition from raw data collection to motion categorization.

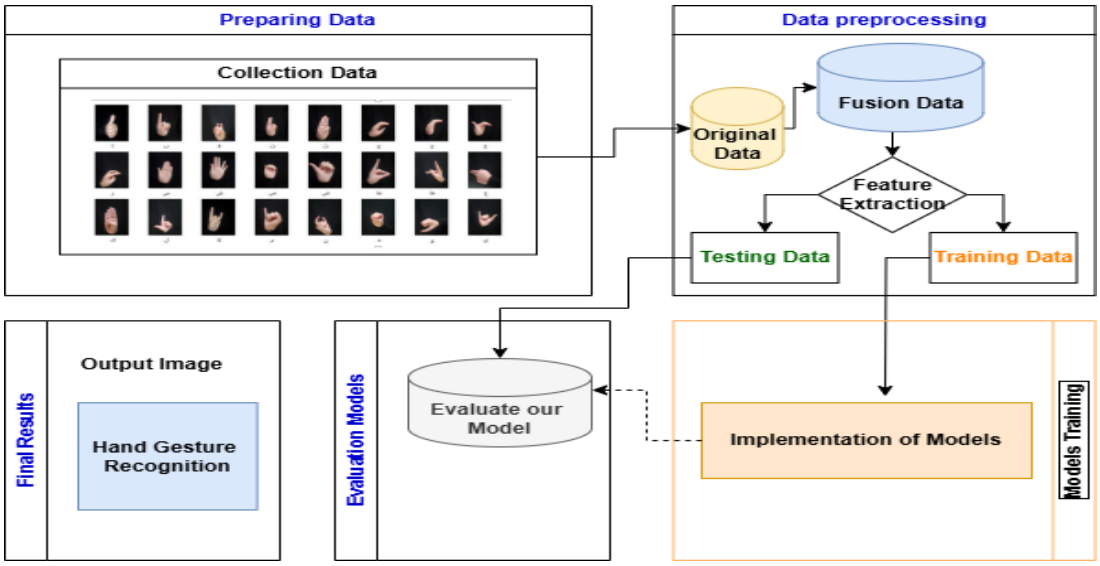

Figure 2-1 Architecture of the proposed method for hand gesture recognition.

As shown in **Figure 2-1**, there are five primary stages that need to be followed, starting from data collection and preparation, then data preprocessing and feature extraction, followed by model implementation and training, model evaluation, and finally gesture detection.

2.3. Image Processing/ Statistical Modelling based approaches

These methodologies use a variety of image processing techniques, including classifiers, fuzzy logic, and rule-based algorithms, to recognize gestures in both ArSL and ASL. By addressing issues such as hand segmentation, sequence identification, and gesture variability, these methods aim to improve recognition accuracy. In the following, we present recent works that exploit these techniques in both Arabic and American Sign Language.

2.3.1. Techniques for ArSLR

Omar Al-Jarrah et al. [AJH01] developed an ArSLR system that uses neuro-fuzzy techniques to automatically translate manual letter movements without the need for gloves or visual markers. The system is built on a first-order Sugeno-type fuzzy inference model, with training intended to give an output value of 1 for successfully recognized motions and 0 otherwise. The system accurately detected all 30 Arabic manual alphabets, with an overall recognition rate of 93.55%. Specific gestures demonstrated improved identification rates of 90.00%, 86.66%, and 95.00%, respectively. Despite its effectiveness, the system encountered difficulties discriminating between visually similar movements and balancing performance gains with model complexity.

Simillarly, Arora and Roy [AR17] used the fuzzy C-means clustering algorithm and neutrosophic logic to create an ArSLR system. A dataset from the Al-Amal Institute in Damietta for Deaf Students [EE20]. With an overall accuracy of 91%, the suggested system used a categorization strategy based on fuzzy logic. Despite its success, the study found a number of challenges. It was difficult to structure sign language sentences for translation and recognition since there was a dearth of linguistic study on ArSL. Furthermore, dataset building was made more difficult by the lack of defined techniques for characterizing sign language features.

Ahmed et al. [AAM20] developed a system to identify isolated dynamic ArSL motions and translate them into Arabic text, addressing gesture variability and dynamic recognition. Their model, trained on 100 distinct dynamic ArSL signs from 1,500 video clips using a

Weighted Euclidian Distance metric for gesture similarity, achieved a recognition accuracy of 95.8%.

A. A. M. Riad et al. [RSH14] created an ArSLR system that uses color-based hand localization to compensate for differences in hand size and form. The technology includes a hand region description technique that detects border points in images and extracts geometric information for gesture identification. A rule-based classifier is used to categorize hand motions based on the extracted features. The system successfully recognizes static ArSL gestures by combining a vision-based geometric model and a rule-based classification technique. However, the model is prone to false positives and inaccurate classifications in some circumstances. Despite this limitation, the system achieved a high recognition accuracy of 95.3% when evaluated on a dataset of seven ArSL terms.

2.3.2. Techniques for ASLR

Sharmila Gaikwad et al. [GSS19] proposed an image processing and machine learning system for recognizing American Sign Language (ASL) motions. The system works by taking photos through a webcam and then comparing them to pre-stored images of ASL characters using the Scale-Invariant Feature Transform technique. The SIFT technique is used to extract important elements from the input image that can withstand scaling, rotation, and translation, making it perfect for gesture identification. The technology recognizes ASL motions and provides text translations, which are subsequently transformed to voice. The system is intended to improve communication for those with hearing problems by providing a new way for them to connect with technology. However, the system may struggle to handle different hand shapes and poses, and the lack of a sufficient ASL dataset limits further improvements.

Halder and Tayade et al. [HT21] developed a real-time ASL recognition system using Mediapipe for Support Vector Machine (SVM) classification and hand landmark detection. 21 important hand landmarks were retrieved using their method, which then transformed them into x and y coordinates for feature-based classification. Their model demonstrated remarkable accuracy of 99.15% for ASL alphabets after being trained on 156,000 ASL alphabet images and 1,400 ASL number images. The Mediapipe + SVM architecture offered a lightweight and computationally efficient solution in contrast to deep learning models like Convolutional Neural Networks (CNNs) and Artificial Neural Networks (ANNs), which made it perfect for real-time mobile deployment.

Nurhadi et al. [NWS24] developed a system to recognize ASL fingerspelling by combining global feature descriptors and machine learning methods. The suggested approach used Color Histogram, Hu Moments, and Haralick Texture to extract features, allowing for a more robust representation of hand gestures. The collected characteristics were then classified using Logistic Regression and Random Forest models to determine their usefulness in gesture detection. The trial findings showed that Random Forest outperformed Logistic Regression, with an accuracy rate of 86% versus 48% for the latter. Classes B, F, H, I, K, Y, and Z had the highest classification precision, whilst class U had the lowest performance. However, the system struggled to discriminate motions with visually similar elements and was sensitive to differences in contrast and noise.

2.4. Classic Machine Learning-Based Approaches

In these methodologies, Sign Language were recognized using conventional machine learning methods such as SVM, Decision Trees, and KNN. To improve accuracy, feature extraction and noise reduction were essential. The usefulness of several classifiers and methods, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA are emphasized in the section.

2.4.1. Existing Approaches for ArSLR

A.M. Ahmed et al. [AAM21] developed the ATASAT System (Automatic Translation of Arabic Sign Language to Arabic Text), which converts ArSL motions into written Arabic text utilizing advanced picture and pattern recognition algorithms. The system uses many classification algorithms, including C4.5, Naïve Bayes, KNN, Multilayer Perceptron (MLP), Sequential Minimal Optimization (SMO), and Voting Feature Intervals (VFI), to accurately identify hand movements based on their distinctive properties. The system interprets videos and collected photos, evaluating hand shapes and movements to generate Arabic text.

Mosab A. Hassan et al. [HSA24] developed an automatic ArSLR system that combines machine learning, dimensionality reduction, and image processing. Their method uses PCA for dimensionality reduction, LDA for feature extraction, and KNN for classification. The system was tested using ArSL letter images and achieved 86.4% accuracy when trained on 90% of the dataset. KNN outperformed AdaBoost, Decision Tree (DT), and Naïve Bayes. While the approach significantly reduces feature dimensions while maintaining accuracy, its performance is heavily influenced by input data quality and hyper parameter settings.

Mahmoud M. Khattab et al. [KZM24] used image processing and statistical modeling to identify gestures in ArSL. During the hand detection phase, two techniques were evaluated: Sobel and Threshold, with the latter achieving the highest detection accuracy of 98.64%. Four classifiers were tested: C4.5, Naive Bayes, KNN, and MLP. KNN had the highest accuracy, ranging from 88.38% to 99.62%. The study also looked at the influence of hidden neurons on MLP accuracy, and discovered that while additional neurons improved accuracy, it also increased processing time. The results demonstrated the importance of dataset quality and classifier selection in obtaining peak performance in ArSL identification.

Altememe et al. [ATE22] created ArSL identification using machine learning approaches to improve communication in the Deaf and Hard of Hearing (DHH) community. The study extracted features from hand gesture photos using LDA, which transformed them into a lower-dimensional space while keeping crucial visual qualities. The study used various machine learning techniques for classification. The dataset, which included 32 Arabic letters, controlled for differences in background, illumination, hand shape, and skin tone. Experimental results indicate that k-NN had the highest accuracy (~87%), followed by Random Forest (~79%), while DT, NB, and SGD performed poorly (~48-53%). The study identified problems such as lighting changes and misclassification of visually identical. The data imply that k-NN is the most successful classifier for ArSL recognition, while more work is needed to increase real-time adaptation.

R.M. Mohammed et al. [MK21] conducted a comprehensive assessment of ArSL translation systems, focusing on important advances in the area. The study looked at many approaches to sign identification, including frequency-based transformations like the Fourier Transform, Hartley Transform, and Log-Gabor filters for feature extraction. Researchers have used sensor-based technology such as Kinect and Leap Motion controllers to improve hand movement tracking. SVM, KNN, ANN, and Logistic Regression are some of the classification approaches used in various systems.

A. M. Zakariya et al. [ZJ19] created an ArSL detection system for smartphone platforms that uses a client-server architecture. In this system, the smartphone collects and sends sign gestures to a server for processing and categorization before displaying the recognized gesture to the user. The system incorporates image processing algorithms for background detection and feature extraction, which use binary pixel representation to improve accuracy. The study used a SVM model trained on a collection of 200 photos per gesture for ten different ArSL

signs. The trial findings showed an accuracy of 92.5%, indicating that the system is excellent at recognizing these motions. However, the system can only recognize ten movements and is restricted by smartphone processing capabilities, which may have an impact on real-time performance.

2.4.2. Existing Approaches for ASLR

Rajeev Goel et al. [GBG25] developed an Automatic SLR system tailored to ASL. Their method overcomes the curse of dimensionality, a major issue in ASL recognition, by combining Histogram of Oriented Gradients (HOG), Autoencoders, and Grey Wolf Optimization (GWO).in the first step, HOG collects important visual information from ASL movements, including hand shapes and motion patterns. The method beat standard techniques such as PCA-IGWO and KPCA-IGWO, with 98.4% accuracy and an F1-score of 96.59%.

Halder and Tayade et al. [HT21] developed a real-time ASLR system using Media pipe for SVM classification and hand landmark detection. Important hand landmarks were retrieved using their method, which then transformed them into x and y coordinates for feature-based classification. Their model demonstrated remarkable accuracy of 99.15% for ASL alphabets after being trained on 156,000 ASL alphabet images and 1,400 ASL number images.

Nurhadi et al. [NWS24] developed a system to recognize ASL fingerspelling by combining global feature descriptors and machine learning methods. The suggested approach used Color Histogram, Hu Moments, and Haralick Texture to extract features. The collected characteristics were then classified using Logistic Regression and Random Forest models to determine their usefulness in gesture detection. The trial findings showed that Random Forest outperformed Logistic Regression, with an accuracy rate of 86% versus 48% for the latter. However, the system struggled to discriminate motions with visually similar elements and was sensitive to differences in contrast and noise.

Bala et al. [BHI22] developed an ASL recognition system that uses classic machine learning classifiers to improve alphabet categorization accuracy. The system was trained using the Sign Language MNIST dataset. CNN led the other models, obtaining 100% accuracy, followed by Random Forest (98.63%) and SVM (97.92%).

Mitra et al. [MRM22] proposed an ASL identification system. Their technology analyzes video-based hand motions, identifies essential elements, and classifies them using machine

learning models to produce text and voice output. The paper evaluates multiple ASL classification algorithms for Bengali SL, including SVM-based models (95% accuracy) and deep CNNs (96.33% accuracy). The system combines real-time video processing, segmentation, and feature extraction, comparing observed features to pre-trained models to categorize indications. Experimental results indicate 64.70% accuracy for certain ASL characters, with errors due to environmental noise and hand positioning inconsistencies.

Kaur and Garg [KG24] developed a machine learning-based system for real-time recognition of ASL from video streams. The system preprocesses video frames and extracts key hand traits before applying machine learning methods such as Naïve Bayes, Random Forest, SVM, KNN, Logistic Regression, and Decision Trees. Each model was tested for accuracy and efficacy in classifying ASL hand motions, and Decision Tree outperformed the rest, obtaining an astonishing 99.79%.

Miah et al. [MTA25] developed a vision-based system for detecting ASL utilizing a basic camera and the MediaPipe Hands algorithm to track hand joint movements in RGB images. They employed two feature extraction techniques: distance-based, which determines the distance between hand joint locations, and angle-based, which computes the angles between vectors and 3D axes. They used three different datasets to categorize the indicators with SVM and Light Gradient Boosting Machine (GBM). Their algorithm scored 99.39% accuracy on the Massey dataset, 87.60% on the ASL Alphabet dataset, and 98.45% on the Finger Spelling A dataset.

2.5.Deep learning-based approaches

In recent years, deep learning techniques have transformed sign language recognition by allowing for more accurate and efficient gesture classification. Unlike traditional methods that rely on handcrafted features, deep learning models use automated feature extraction, spatiotemporal modeling, and sequence prediction. This section looks at several deep learning-based approaches, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Transformers, and hybrid architectures. The evaluated studies concentrate on Arabic Sign Language (ArSL) and American Sign Language (ASL), emphasizing advances, obstacles, and future research objectives.

2.5.1. Existing Approches for ArSLR

Ahmad M. J. AL Moustafa et al. [AMJ24] proposed an Automatic ArSLR system, where they employs CNNs for feature extraction and PCA for dimensionality reduction, resulting in increased computing efficiency while keeping important visual features. Trained on a dataset of 600 ArSL gestures, the model recognized individual movements with 91% accuracy. Despite these promising findings, problems remain, such as the need for larger datasets, gesture variability, and the lack of grammatical structures in Arabic sign language.

Herbaz et al. [HEB25] developed an advanced deep learning SLR system for ArSL. Two new datasets were presented in their work [RW18]. The researchers used the VGG16, VGG19, and ResNet50 models for classification, normalization, and detection. Two training regimes were used to assess the system: one without fine-tuning and another with improvements for fine-tuning. With VGG16 reaching 99.05%, VGG19 reaching 99.99%, and ResNet50 reaching 98.50%, the results showed great identification accuracy.

Ahli et al. [AHL23] developed a deep learning-based model for SL translation that uses CNNs and RNNs to extract spatiotemporal information from sign language movies and translate them to text. The system was tested against several standard classifiers, including K-NN, Decision Tree, Boosted Decision Tree, and SVM. In classification, the K-NN achieved 85.80% accuracy on validation and 83.10% on test data. The SVM model performed well overall, with an F1-score of 79.30% on validation and 79.20% on test data, whereas Boosted Decision Trees achieved 85% validation accuracy and 83.10% test accuracy. Among all models, the CNN-RNN technique beat traditional classifiers, with validation accuracy of 86.94% and test accuracy of 80.01%.

Oulad-Naoui et al. [ONB24] developed an LSTM-based system for identifying dynamic ArSL that incorporates deep learning approaches. In order to improve recognition accuracy, they used a CNN-LSTM-SelfMLP architecture, which merged a multi-layer perceptron (SelfMLP) into CNN-LSTM models. Using MobileNetV2 and ResNet18 as CNN backbones for feature extraction, the study investigated six distinct models. The suggested method successfully captured the sequential character of ArSL gestures by utilizing LSTM networks for temporal modeling.

Al-Hammadi et al. [AMH20] developed a 3DCNN for spatiotemporal feature learning to create a hand gesture identification system for sign language. In the first technique, a 3DCNN

extracted features from whole video samples for classification using a SoftMax layer. In the second approach, the 3DCNN was trained to extract features from different parts of the video in order to improve temporal dependency. The system demonstrated recognition rates of 84.38%, 34.9%, and 70% in signer-independent mode and 98.12%, 100%, and 76.67% in signer-dependent mode when tested on three gesture datasets.

Ali Alani et al. [AC21] developed the ArSL-CNN model, a deep learning framework based on CNN for recognizing ArSL movements. The model was trained and tested on the ArSL2018 dataset, which included 54,049 images representing 32 ArSL motions. The system had training and testing accuracy of 98.80% and 96.59%, respectively. To address data imbalance difficulties, the scientists used the Synthetic Minority Over-Sampling Technique (SMOTE), which increased the model's accuracy to 97.29%. Despite this improvement, the model's performance remained influenced by the quality and quantity of training data, making generalization to unseen motions difficult.

Rawf et al. [RMA23] developed a hybrid strategy to detect and classify Arabic-script-based sign language motions, integrating 2D CNN with transfer learning. The system was trained and tested on the ASSL2022 dataset, which was created from two publicly available datasets—the ASL alphabet dataset and ArSL2018—and contained tagged Arabic-script images for 40 sign language classes. The study showed that Arabic hand gestures may be recognized with about 100% accuracy. However, problems such as dataset size, diversity, generalization to novel gestures, and hardware dependency may have an impact on real-world deployment.

Saleh Aly et al. [AA20] developed a comprehensive Arabic Sign Language (ArSL) recognition system that addresses three major challenges: hand segmentation, hand shape representation, and gesture sequence identification. The system used DeepLabv3+ to segment hands, Convolutional Self-Organizing Maps (CSOM) to extract hand shape features, and Bi-directional Long Short-Term Memory (Bi-LSTM) to recognize gesture sequences. The model was trained using a dataset of 23 distinct ArSL word signs, each performed by three different persons. The experimental findings demonstrated a recognition accuracy of 89.59%, illustrating the usefulness of using deep learning models into dynamic sign language recognition.

Moustafa et al. [MAM24] developed a hybrid strategy that uses Mediapipe for hand landmark detection and a CNN model to improve Arabic Sign Language (ArSL) recognition.

Their model was trained using a dataset of 7,000 photos, which included 28 distinct ArSL motions, and achieved a validation accuracy of 97.1%. However, the study identified categorization issues caused by strong visual similarities between specific characters, such as "Beh," "Teh," and "Theh," which had an impact on model performance. Variations in hand position and signing techniques created additional obstacles, underlining the need for increased robustness and adaptability in real-world situations.

2.5.2. Existing Approaches for ASLR

Miah et al. [MTA25] integrated Graph Convolutional Networks (GCN) with CNNs and Multi-Head Self-Attention (MHSA) to develop a deep learning framework for ASLR. By using SLIC-based superpixel segmentation to improve spatial feature extraction, their model, Graph Meets Attention and CNN (GmTC), increases the effectiveness of gesture recognition. The suggested model demonstrated an accuracy of 99.46% on ASL-10 and 99.60% on ASL-20. However, the lack of real-time optimization, and NLP integration for translation, are some of the system's drawbacks despite its high accuracy.

Sharmila Gaikwad et al. [GSS19] proposed an image processing and machine learning system for recognizing ASL motions. The system works by taking photos through a webcam and then comparing them to pre-stored images of ASL characters using the Scale-Invariant Feature Transform technique. The technology recognizes ASL motions and provides text translations, which are subsequently transformed to voice. The suggested method uses CNNs for feature learning and classification, resulting in an effective framework for ASL recognition. However, the system may struggle to handle different hand shapes and poses, and the lack of a sufficient ASL dataset limits further improvements.

Lee et al. [LK99] developed a real-time ASL recognition system employing a LSTM network and KNN. The system used a Leap Motion Controller to extract information like sphere radius, finger angles, and distances for accurate gesture classification. After training on 2,600 ASL samples, the system attained an average accuracy of 99.44% for 26 ASL alphabets and 91.82% accuracy in 5-fold cross-validation. This study demonstrates the feasibility of combining motion sensors and deep learning to provide real-time ASL detection and learning applications.

Elakkiya et al. [NRE22] proposed a comprehensive sign language system that includes Generative Adversarial Networks (GANs), 3D-CNNs, and LSTMs for gesture identification, translation, and video creation. The system used MediaPipe and a CNN + Bi-LSTM model to extract poses and generate text, with a classification accuracy of more than 95%. Evaluation criteria including BLEU (38.06), SSIM (0.921), and PSNR (29.73) demonstrate the model's good accuracy and visual quality.

Ahmadi et al. [AMA24] proposed a deep hybrid model for sign language recognition that combines a Custom CNN and a Temporal TCNN. The system was assessed using publicly available benchmark datasets that included both British Sign Language (BSL) and American Sign Language (ASL). The CNN-TCN model uses a structured pipeline that includes data collection, preprocessing (labeling, normalization, and frame extraction), feature extraction with CNN, and sequence modeling with TCN. The experimental findings showed high performance, with accuracy, precision, recall, and F1 scores of 95.31%, 94.03%, 93.33%, and 93.56%, respectively, demonstrating the efficiency of CNN-TCN in recognizing isolated characters and numbers.

Paul et al. [PWP24] developed an ASL recognition system that uses deep learning approaches to improve real-time gesture classification. Their model used a ResNet-based CNN trained on a dataset of 26 ASL hand signs. The system was optimized with the Adam optimizer, resulting in a classification accuracy of 89.07%. Furthermore, Paul et al. [PWP24] proposed a sequence-based ASL recognition system that uses Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). The dataset included three basic ASL gestures: "hello," "I am hungry," and "thanks." The evaluation findings showed that LSTM outperformed GRU, with an accuracy of 94.3% versus 79.3% for GRU.

Baihan et al. [BAA24] developed a hybrid deep learning model called CNNSa-LSTM that combines CNN, Self-Attention (SA), and LSTM to improve sign language recognition. The system uses VGG16 for spatial feature extraction and optical flow for motion feature analysis, attaining 98.7% accuracy on sign language datasets. The proposed approach performed well across various measures, with a sensitivity of 98.2%, precision of 98.5%, word error rate (WER) of 0.131, and sign error rate (SER) of 0.114.

Camgoz et al. [CHK20] developed a Transformer-based architecture for end-to-end ASL recognition and translation. Without the requirement for explicit timestamp annotations, their

methodology combines Connectionist Temporal Classification (CTC) loss to jointly maximize both recognition and translation tasks.

Said et al. [SMT24] developed a deep learning framework for continuous ASL detection and translation based on Adaptive Transformer (ADTR). To improve translation accuracy, their model added Adaptive Masking (AM), Local Clip Self-Attention (LCSA), and Adaptive Fusion (AF). Their method allows for direct translation from ASL video sequences to spoken language text because it does not rely on gloss annotations like traditional SLT models do.

.

2.6.Conclusion

Sign language recognition research in both Arabic and American Sign Language (ArSL and ASL) has achieved great advances, but significant gaps remain. ArSL research is currently sparse, with each using distinct approaches, making direct comparisons impossible. The lack of consistent, large-scale datasets complicates the creation of robust models. In contrast, ASL research benefits from well-established datasets, which enable more consistent evaluations and increases in recognition accuracy. Deep learning-based techniques, including CNNs, RNNs, and Transformer models, have demonstrated encouraging results in ArSL and ASL recognition, with high accuracy in gesture classification and phrase translation. Traditional machine learning algorithms, such as SVM and KNN, remain useful for certain tasks, like static gesture identification. However, variances in dataset quality, gesture complexity, and signer differences continue to provide issues for both languages. Future research should concentrate on dataset expansion, real-time recognition improvements, and the use of multimodal approaches to improve accuracy and practicality in real-world circumstances. Standardization of ArSL datasets and procedures will be critical in closing the gap between ArSL and ASL recognition systems.

Chapter 3 Conception

3.1.Introduction

In this chapter, we provide a deep learning method for identifying Arabic and English (American) sign languages. For precise and effective hand sign identification and recognition, we used a variety of technological methods, such as Convolutional Neural Networks (CNN) separately and YOLOv8. Strict preprocessing procedures were taken to guarantee high-quality findings before applying these algorithms to datasets that included signs from both Arabic and American sign languages. In order to promote inclusive and seamless communication, this chapter offers a systematic approach to recording and deciphering gestures from Arabic and American sign languages. The integration of contemporary methods to promote intercommoned linguistic and cultural understanding is underlined.

3.2.Challenges and Objectives of the System

The primary objective of this project is to develop a deep learning-based system capable of accurately detecting and classifying Arabic and English Sign Language letters from images and real-time video streams, as well as recognizing the extracted letters correctly. By leveraging advanced deep learning techniques, the system aims to identify and extract the necessary letter information and convert it into readable text output. **Figure 3.1** illustrates the general architecture of the proposed system.

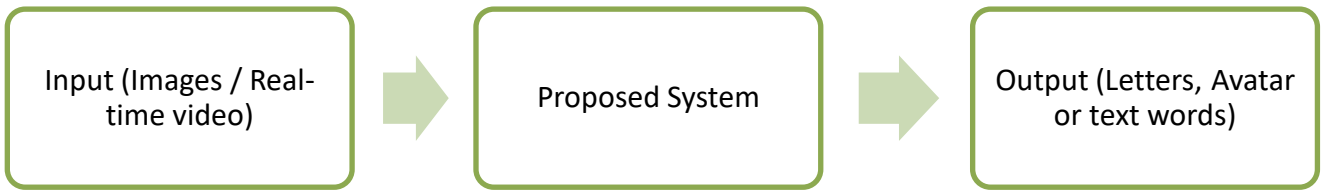

Figure 3-1 General architecture of our proposed system.

3.3. Overview of Utilized Datasets

In order to create a deep learning-based system for recognizing sign language, we looked for the best datasets that met our methodology's needs. Our project mostly concentrated on identifying individual letters in both Arabic Sign Language (ArSL) and American Sign Language (ASL), as opposed to whole words or phrases. In addition, to further specify Arabic sign language for the Maghrebi region, we created a new dataset called the 'Maghrebi Arabic Dialect Sign Language Dataset' (MASLD).

3.3.1. Arabic Sign Language (ASL) Alphabet Dataset

In this project, two Arabic Sign Language datasets were used in order to enhance the model’s performance and improve its ability to generalize across different conditions in both versions (image and real-time).

A. ArSL21L Dataset

The ArSL21L dataset [AS21] is a comprehensive and recent dataset designed for Arabic Sign Language letter recognition. It contains 14,202 high-resolution images covering 32 Arabic sign language letters (see Figure 3-2). The images are captured with a wide variety of backgrounds, lighting conditions, and signer appearances, enhancing the dataset's diversity. Each image is annotated with bounding boxes in the PASCAL VOC format, enabling both classification and detection tasks. The dataset size is approximately 803 MB and is divided into training and testing splits, commonly with 80% for training and 20% for testing. This dataset is suitable for training advanced deep learning models including convolutional neural networks and object detection frameworks.

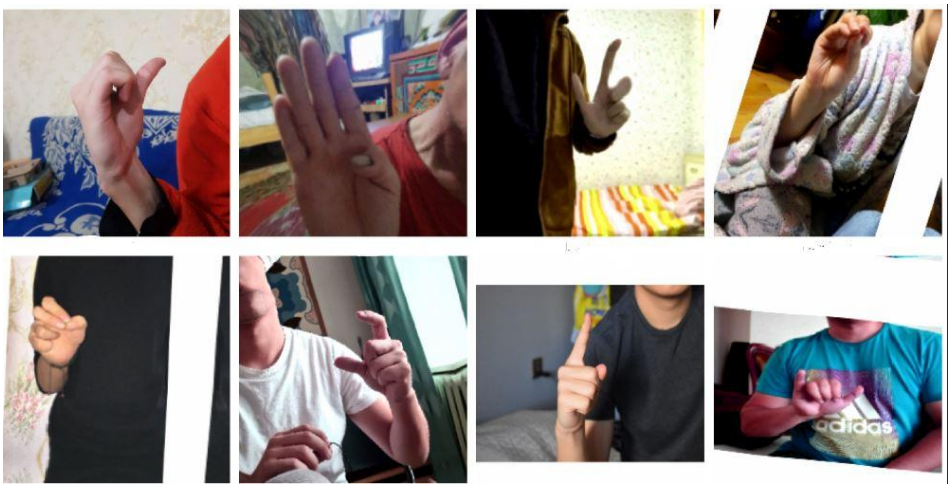

Figure 3-2 A sample of the dataset showing some Arabic sign language alphabet letters.

B. ArSL2018 Dataset

The ArSL2018 dataset [AS18] consists of static images representing the 32 letters of the Arabic Sign Language alphabet. It was collected under more controlled conditions with consistent background and lighting to provide clear visibility of the signs. The dataset size is smaller compared to more recent collections, focusing primarily on classification tasks rather than detection. Images are standardized in size, facilitating model training and evaluation. The dataset is widely used in early-stage research on Arabic Sign Language recognition and serves as a reliable resource for benchmarking classification models (see Figure 3-3).

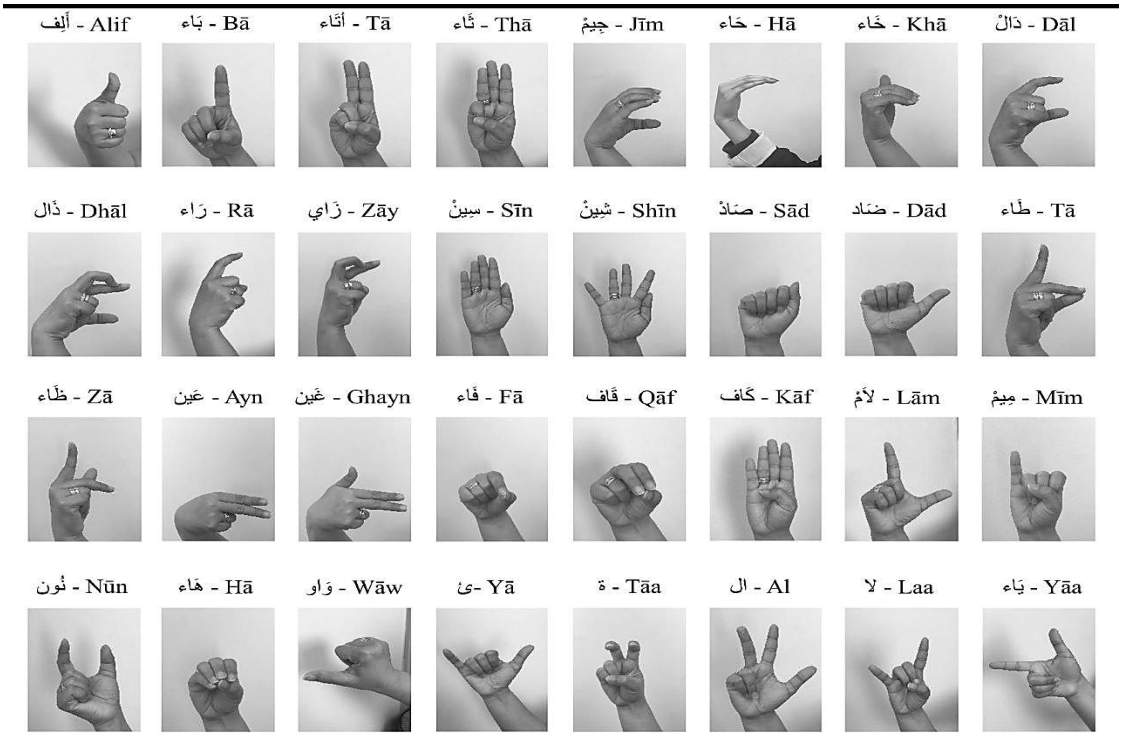

Figure 3-3 A sample of the dataset showing some Arabic sign language alphabet letters. [AS18].

3.3.2. American Sign Language (ASL) Alphabet Dataset

In English sign language, specifically American Sign Language (ASL), two datasets were used to train and evaluate a deep learning model for recognizing ASL alphabet signs.

A. ASL Dataset

The ASL dataset [MSK24] consists of static images representing the 26 letters of the American Sign Language alphabet. The images were collected in natural and varied conditions with diverse backgrounds, lighting, and hand orientations, providing a more realistic representation of real-world usage. Compared to more standardized datasets, this collection is smaller in size and includes some variability that makes it suitable for detection-oriented and robustness-focused tasks. Although not originally intended for classification, the dataset can also be adapted for early experimentation in recognition systems. It serves as a useful supplementary resource for evaluating model generalization in unconstrained environments.

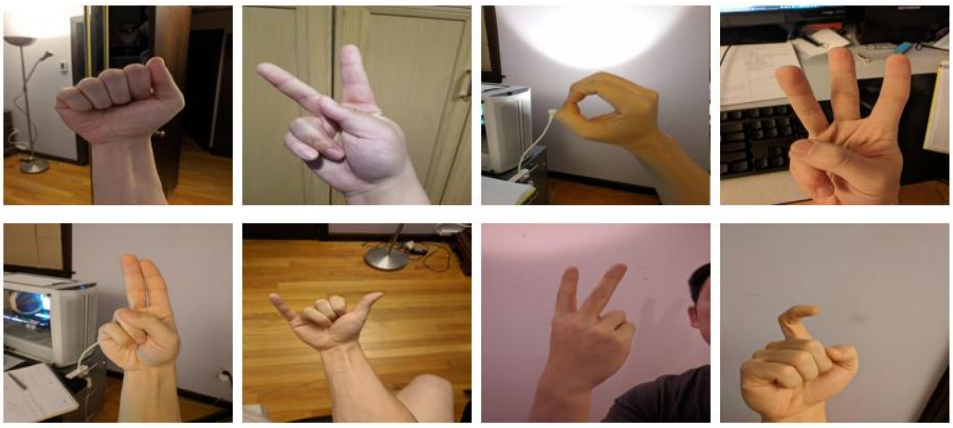

Figure 3-4 A sample of the dataset ASL showing some sign language alphabet letters.

B. The ASL Alphabet dataset

The ASL Alphabet dataset [GA20] consists of static images representing the 26 letters of the American Sign Language alphabet, along with additional signs such as "Nothing", "Space", and "Delete" (see Figure 3-5).

Figure 3-5 Example of ASL Alphabet Signs Represented by Hand Gestures.

The dataset was collected under highly controlled conditions with uniform backgrounds, consistent lighting, and standardized hand positions to ensure clarity and uniformity across samples. With over 87,000 high-resolution images, the dataset is considerably large and primarily designed for classification tasks. All images are uniformly sized (200x200 pixels), which simplifies preprocessing and facilitates training deep learning models. This dataset is widely used in academic research and practical applications related to ASL recognition and serves as a strong benchmark for evaluating classification models.

3.4. System Architecture Suggestion

This section presents the basic phases of sign language recognition for translating the provided sign language into Arabic and English words or sentences. The proposed system consists of the following phases:

Preprocessing phase: In this phase, we prepare the images by extracting, resizing, normalizing, and optimizing their lighting to suit the models.

Object Localization and Extraction: This phase focuses on locating the hand only, as it is the primary element in sign language.

Recognition phase: In this phase, we use different deep learning techniques to recognize sign language using separate models, including Convolutional Neural Networks (CNN), YOLOv8.

Sentence or Word Formation: The recognized letters from the models, whether in English or Arabic, are used to form words and sentences.

Sign Translation: This phase is based on ensuring the correctness of words and sentences in both American and Arabic sign languages to preserve the intended meaning. In addition, a 3D avatar was created and animated based on the corrected, generated words.

From Text to Signs: This phase is considered the reverse operation, where the input is plain text and the output is sign language. It aims to ensure effective communication between speakers and signers. Figure 3-9 illustrates the proposed approach of our system.

3.4.1. Language Selection

Users can choose between the Arabic and English alphabets for sign language recognition with this function. The system recognizes and deciphers hand signals that correspond to letters in both languages with accuracy. The application's usability is increased by supporting both alphabets, making it suitable for Arabic sign language learners and English speakers. Because users can quickly switch between the two alphabets according to context or choice, the application is adaptable and usable by a wide variety of users.

3.4.2. Preprocessing

To guarantee precise outcomes in the following stages of the hand sign recognition system, our system starts by preprocessing the input photos. A generalized preprocessing pipeline for this stage is represented by the pseudocode below:

```
function preprocessImage(image):
    normalized_image = normalizeImage(image)
    adjusted_image = adjustImage(image)  # Resize or scale the image to fit the model requirements while preserving aspect ratio
    corrected_image = correctLighting(adjusted_image)
    return corrected_image
```

A. Image Normalization

At this point, each image's pixel values are adjusted to fall between 0 and 1. For 8-bit images, this entails translating the original pixel values, which range from 0 to 255, to a scale from 0 to 1. During training, this normalization enhances the model's performance and stability.

B. Image Resizing

The image is resized to fit the model's input specifications while, whenever feasible, maintaining the original aspect ratio rather than resizing it to fixed dimensions. This guarantees that the key hand sign features won't be distorted when the model handles changes in image sizes.

C. Lighting Correction

After resizing the photos, techniques like histogram equalization or other lighting correction approaches are applied to lessen lighting variances and improve the visibility of key characteristics, which helps to increase the overall recognition accuracy.

3.4.3. Full landmark extraction

Key hand landmarks are extracted from the photos in the following step when the preprocessing is finished. Since it only recognizes the most important aspects of the hand, this stage is necessary to achieve high accuracy in sign identification. In order to prepare the photos for tracking models and improve the identification system's overall performance, these extracted features are essential.

A. Object Localization and Extraction

At this point, precise object detection and position adjustment are accomplished by the use of specialized algorithms. These techniques also aid in fixing any improper rotations that can show up in the pictures.

a. Rotation and Mirroring

When the hand in an image is not oriented correctly, our system uses transformation techniques including rotation and mirroring. During training, this step increases data diversity and enhances model performance. Both CNN models for hand posture classification and YOLO models for hand detection depend on this procedure.

b. Concatenation

Following the hand landmark extraction process, the output is transformed into a table that includes each landmark's visibility as well as its x, y, and z coordinates. A zero array is produced if no hand is found. CNN models that depend on hand landmark points for accurate classification or identification are the main users of this data structure. In the meantime, the photos are downsized to 640 by 480 pixels and immediately transformed to RGB format for YOLO models in order to detect hands within the frame.

B. Holistic detection

Key landmarks like hands and body position may be detected simultaneously and incorporated thanks to Google's MediaPipe Holistic machine learning engine. By integrating several sub-models, this model offers a comprehensive solution for precise human motion analysis. This phase comes after the one that concentrated on extracting basic features of the detected items since the model combines position and hand tracking data into a coherent and complete set of human body landmarks. During this phase, we used this integrated model on a continuous stream of photos to generate up to 75 landmarks, 33 for body stance and 21 for each hand. The model was trained using a predetermined quantity of real photographs. There are two kinds of landmarks in the model.

a. Pose Landmark Recognition

A Convolutional Neural Network (CNN) model is used to determine the overall body pose after the pose of the provided image has been evaluated. The posture parameters of the body

and its components, which comprise roughly 33 landmarks, are detected by this model Figure 3-10 depicts the model we used for full-body tracking.

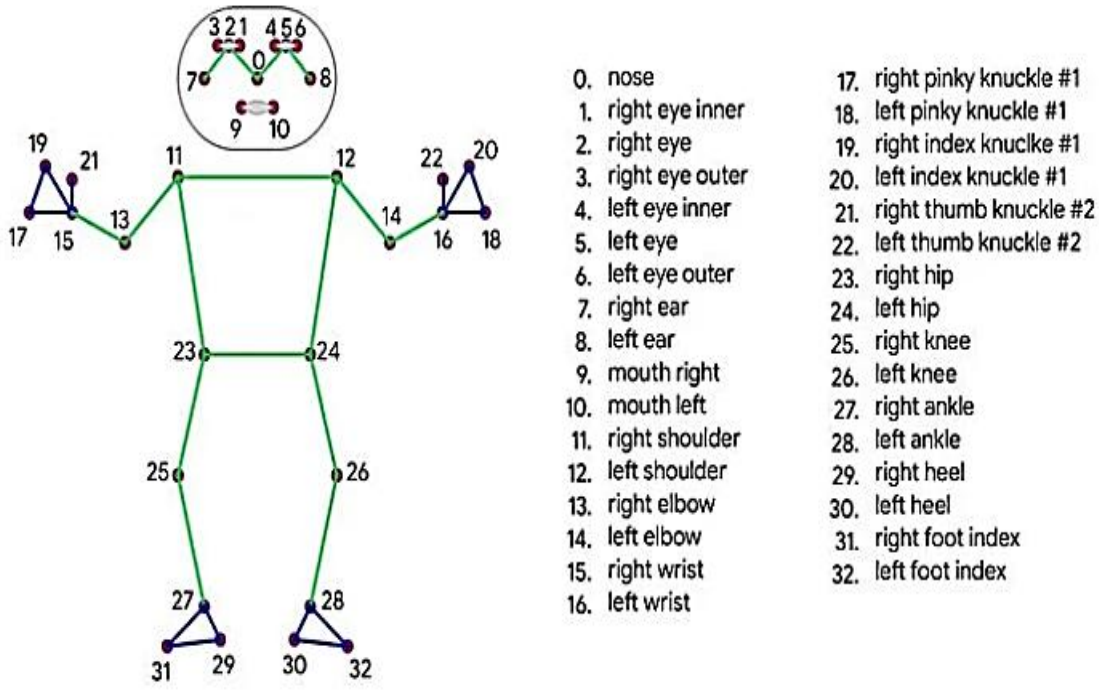

Figure 3-6 shows the possible landmarks present throughout the entire body. [TJB24]

b. Identifying Hand Landmarks

After determining the general body posture, we use its markers to pinpoint the palm areas. Figure 3-11 shows the 20 points that make up the hand landmark model.

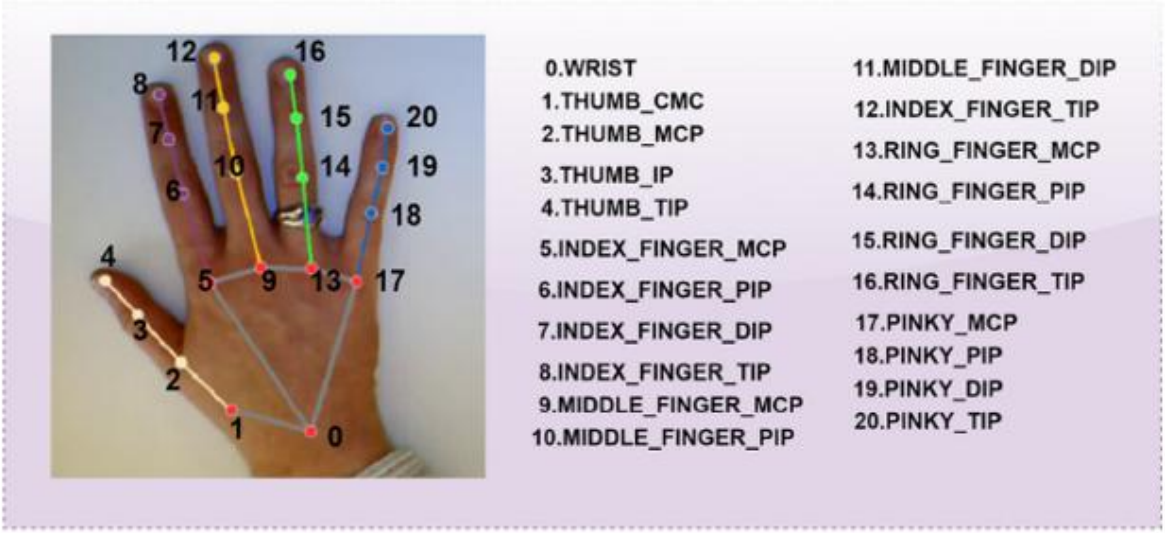

Figure 3-7 illustrates the possible landmarks found on a single hand. [RG23]

C. Key point Values Extraction

A vector of size 21 × 3 = 63 items is produced by representing each hand as a NumPy array with 21 landmarks, each of which has three values (x, y, and z). Because they accurately depict the location and orientation of every hand component, these values are essential for training and enhancing the hand gesture recognition model's performance. Figure 3-12 illustrates the final result of the object localization and extraction phase

Figure 3-8 An example of hand landmark extraction.

3.4.4. Sign Recognition

Deep learning techniques are used to understand the key points vector once it has been extracted. In order to determine the most effective method for sign recognition, we concentrated on different models in our work. Each component of the model is described in detail in the sections that follow.

A. YOLOv8 model

For comparison, we use the YOLOv8 model, whose design is shown in Figure 3-13. We used two separate datasets, each containing images only (no videos), and representing sign language alphabets (see section 3.3):

- The first dataset contains sign images of the Arabic alphabet, consisting of 28 classes, one for each Arabic letter.

- The second dataset contains sign images of the English alphabet, consisting of 26 classes, one for each English letter.

Each dataset was divided into three subsets:

- Training Set: used to train the model to recognize sign classes.
- Validation Set: used to tune and improve the model's performance during training.
- Test Set: used to evaluate the model's performance on unseen data.

Each image is accompanied by a text label file stored in a "label" folder, containing the annotation information necessary for YOLO training, handled independently for each dataset.

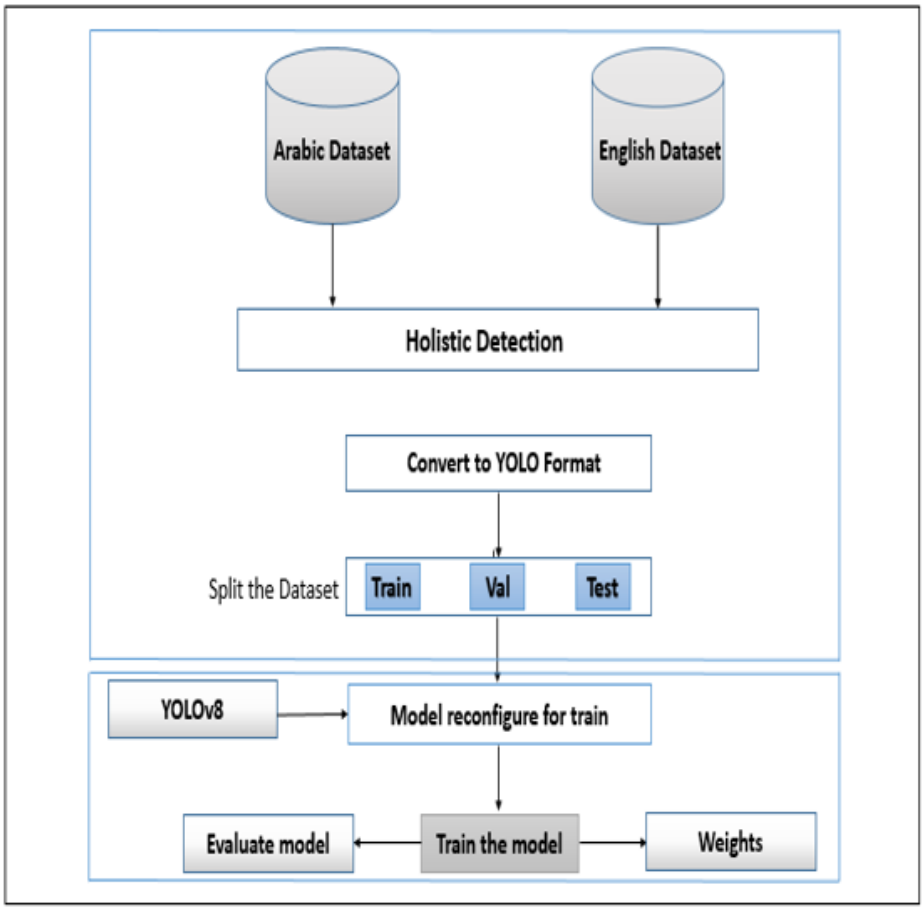

Figure 3-9 Phases of YOLOv8 Model.

Initially, the model showed poor performance. To improve this, we applied the Holistic Detection technique before dataset transformation, which enhanced the data quality and improved the model's accuracy.

a. Creating the Data File (data.yaml)

Distinct data.YOLOv8 was used to build a yaml file for every dataset. Each file includes:

- Paths to the training and validation image directories.
- Each dataset's unique class names (either 26 English or 28 Arabic classes).
- Important details for independently configuring the training environment for every dataset.

b. Training the Model

To guarantee input uniformity across the model, the photos were scaled to 416 × 416 pixels during training. For effective computing, a batch size of 100 was chosen, enabling the model to process 100 photos at once. To make sure each model learns efficiently from its particular dataset, the training process was conducted independently for 120 epochs on both the Arabic and English datasets. The configuration specified in the matching data.yaml file for every dataset, which contained paths, class names, and other crucial factors, served as the basis for all training processes.

c. Model Evaluation

The model.val () script was used to evaluate each model following the training process. To guarantee that the best version of the model was used for evaluation, the best weights that were acquired during training and stored in the best.pt file were imported in this stage. To ensure consistency with the training conditions, photos were scaled to 416 × 416 pixels throughout evaluation. We were able to assess the accuracy and generalization capacity of each model separately for the Arabic and English sign language alphabets thanks to this examination. This is an example of using the YOLOv8 model to translate hand gestures (sign language) into text.

Figure 3-10 Example of Hand Gesture Recognition in Arabic signs.

Figure 3-11 Example of Hand Gesture Recognition in English signs.

B. CNN Model

In this project, we implemented a custom CNN from scratch to classify hand sign images into their corresponding sign language letters. (see Figure 3-16)

- Dropout Layers: Dropout layers are applied after pooling and dense layers to randomly deactivate a percentage of neurons during training, improving generalization and reducing overfitting.
- Flatten Layer: This layer converts the 2D feature maps into a 1D feature vector to prepare for the fully connected classification layers.
- Dense (Fully Connected) Layers: The flattened output is passed through a dense layer with 512 units and ReLU activation. Finally, a softmax dense layer is used to produce the class probabilities.

a. Training the Model

To train the model, we used the following settings:

b. Evaluation and Results

After training, the model was evaluated on both the validation and test datasets. The evaluation results were visualized using :

- Training Accuracy and Loss Curves: These graphs illustrate how well the model learned over time and whether overfitting occurred.
- Confusion Matrix: A confusion matrix was generated to visualize the performance of the model in predicting each individual class.

C. CNN (MobileNet) Model

MobileNet is a lightweight and efficient convolutional neural network (CNN) architecture, originally designed for mobile and embedded applications. In this project, MobileNet was used for both static image classification and real-time gesture recognition using webcam video input, due to its fast inference speed and compact design. **Figure 3-17** illustrates the architecture of the MobileNet model, highlighting its use of depthwise separable convolutions and its suitability for both static image classification and real-time hand gesture recognition.

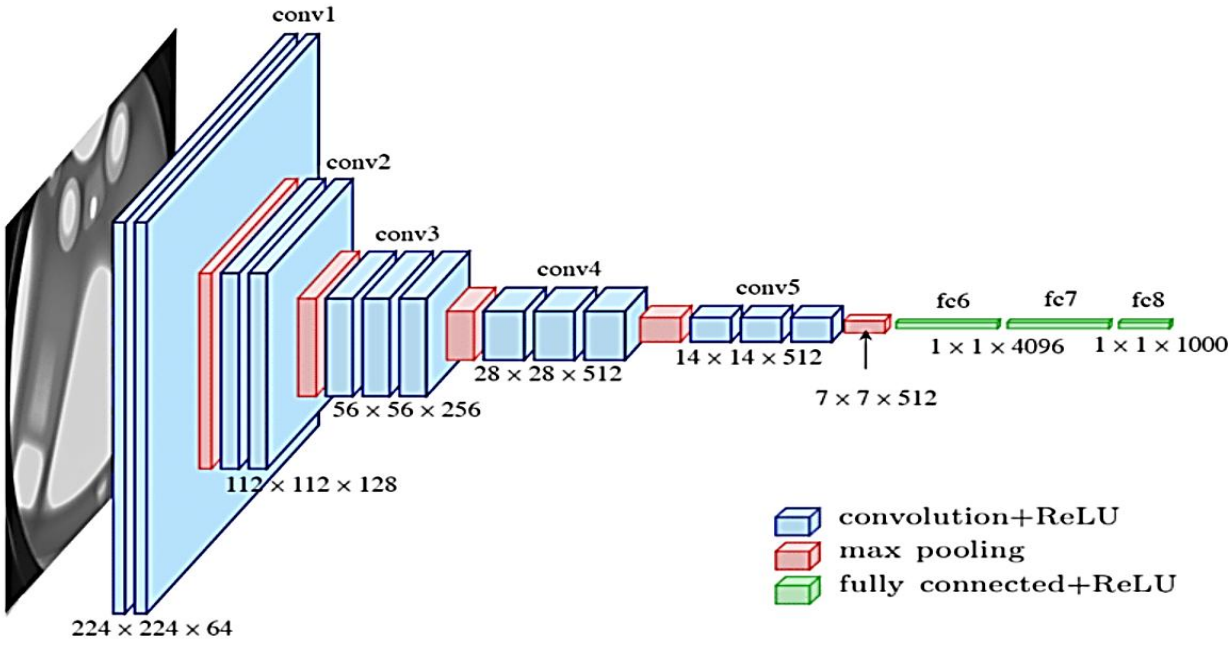

Figure 3-12 MobileNet Architecture.

a. Key Concept: Depthwise Separable Convolutions

MobileNet replaces the standard convolution operation with depthwise separable convolution, which significantly reduces the model's size and computation cost. This operation consists of :

- Depthwise Convolution: Applies a single filter per input channel to extract spatial features.
- Pointwise Convolution (1×1): Combines the spatial features across channels.

This separation drastically lowers the number of parameters and makes the model well-suited for real-time and resource-constrained environments.

b. Architecture Overview

The general architecture used in both applications includes the following components:

- Input: Whether from a static image or a real-time frame, all inputs are resized to 224×224 pixels and normalized.
- Backbone Network: A pre-trained MobileNetV2 model (or similar), optionally fine-tuned on hand gesture data to enhance performance on our specific task.

- Global Average Pooling: Reduces each feature map to a single representative value.
- Fully Connected Layer: Uses a softmax activation function to output probabilities for each gesture class.

c. Integration in Both Use Cases

For Static Images

- Pre-collected images are loaded, preprocessed, and passed through the MobileNet model.
- The predicted class is displayed or used for further processing such as translation or evaluation.

For Real-Time Webcam Input

- Frame Capture: Live video feed is captured using OpenCV.
- Preprocessing: Each frame is resized and normalized before prediction.
- Prediction: The frame is passed through the MobileNet model.
- Output Display: The recognized gesture is displayed on-screen with bounding boxes and class labels for immediate feedback. The image (Figure 3-16) represents an example of translation using a CNN model.

Figure 3-13 Example of Hand Gesture Recognition Using a CNN Model in ArSL.

Figure 3-14 Example of Hand Gesture Recognition Using a CNN Model in ASL.

At this stage, the outputs of classification models such as CNN or YOLOv8 are converted into understandable symbolic characters, like alphabet letters. These characters represent the first level of translation, where no semantic meaning is extracted yet, only the visual or gestural signals are mapped to linguistic units such as:

- Arabic letters: “ت“ ,”ب“ ,”أ”...
- Latin letters (for ASL): “A”, “B”, “C”...

This step serves as the foundation for any further linguistic processing, bridging the gap between raw visual data and symbolic representation.

3.4.5. Sign Translation

A. Grammar Correction Stage

The Sign Translation stage is responsible for transforming sequences of recognized symbols (such as individual letters) into meaningful linguistic structures, such as words or phrases. While the previous stage (**Characters**) produces isolated symbols, this stage works to **assemble and organize these symbols** into comprehensible words within the appropriate linguistic context. This process is not limited to simply concatenating letters; it often involves:

- **Correcting spelling errors** that result from recognition inaccuracies,
- **Applying language modeling** to predict the most likely word based on the letter sequence,
- And in some systems, using **grammatical rules** to ensure proper sentence structure.

Practical Example:
If the following sequence of letters is recognized:
"S", "A", "L", "A", "M"
It is transformed into the word: ***"SALAM".***

B. Text Modeling Stage

Once the signs are translated into words, the system proceeds to generate complete sentences or display individual words based on the real-time context of the gestures. This stage may involve tools for sequencing words according to the target language's grammar, as well as applying morpho-syntactic corrections, such as:

- **Sequence Modeling** for proper word ordering
- **Grammatical Correction** to ensure syntactic coherence

The output at this stage represents the fully interpreted content of the input signs and can be presented in the form of spoken audio, textual display, or animated avatars that visually articulate the message.

•**Textual Display :** The translated signs are initially presented as text, where the system generates coherent words and phrases based on the recognized gestures. Each sign is mapped

to a corresponding letter , resulting in meaningful word or sentences displayed on the screen to enhance clarity and ensure accessibility.

•**Spoken Audio :** In addition to text, the output can be converted into spoken language using text-to-speech (TTS) technology. This audio output plays a crucial role in inclusive communication, allowing hearing individuals to understand the message without relying on visual cues.

•**Animated 3D Avatar :** To strengthen visual communication, especially for the deaf and hard of hearing, the system includes a virtual character (3D avatar) that performs lip-synchronized speech. This is achieved through the Wav2Lip model, an advanced AI system capable of generating highly accurate lip movements synchronized with the spoken audio. This visual representation adds a realistic and human-like dimension to the translation, improving the user experience and supporting comprehension through synchronized facial expression with speech.

3.4.6. Reverse Translation: From Text to Sign

The reverse translation branch transforms written text into visual sign representations through three main steps. First, Text Parsing breaks the input into individual words or characters. Next, Sign Media Matching assigns each element to a corresponding image that represents the appropriate sign, using databases of static sign language images. Finally, Sign Display presents the matched signs in a sequential visual format using only images. This process enables effective communication from hearing individuals to deaf individuals in a clear and accessible visual manner.

3.5. Conclusion

This chapter concluded with outlining the general design structure that serves as the cornerstone of our undertaking. We gave a comprehensive knowledge of the entire process by providing a succinct summary of the important processes that were involved in our work. We will transition from the conceptual design to workable solutions in the upcoming chapter, "Implementation Details," which will demonstrate the practical use of the suggested architecture.

Chapter 4 Implementation

4.1. Introduction

In the previous chapter, we presented the detailed conceptual design of our system. In this chapter, we shift our focus to the development environment and the key libraries and tools used during implementation. Additionally, we provide an overview of the core components of our code base and highlight the main results obtained throughout the development process.

4.2. Development environment

To implement our application, we utilized two personal computers with the following

Table 04-1 Characteristics of the material used.

Model Part	**Laptop 1**	**Laptop 2**
Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
RAM	8,00 Go	4,00 Go
System type	64-bit operating system, x64 processor	64-bit operating system, x64 processor
Edition	Windows 10 Famille	Windows 10 Famille

4.2.1. Programming language

In this study, we used the Python language, which is detailed as follows:

A. Python

Python is a versatile high-level programming language known for its simplicity, readability, and clear syntax, which allows developers to express concepts in fewer lines of code compared to other languages. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, providing flexibility for various applications. Moreover, Python features an extensive standard library alongside a robust

ecosystem of third-party packages such as NumPy, Pandas, and TensorFlow, which extend its capabilities to fields like data analysis, machine learning, and artificial intelligence. This combination of ease of use, flexibility, and powerful libraries has established Python as one of the most popular programming languages in both academia and industry [RSM19] [IDH21].

B. Google Colaboratory

Google Colaboratory (Colab) is a cloud-based platform that provides a free, GPU-accelerated environment to facilitate machine learning research. It is built on Jupyter Notebooks and allows users to run Python code in a browser with integration to Google Drive for data storage. Colab comes pre-installed with key libraries such as TensorFlow, Keras, and PyTorch, eliminating the need for manual setup. It also offers a powerful NVIDIA Tesla K80 GPU to accelerate model training. Despite some resource limitations, Colab remains an effective tool for performing high-performance computing tasks without local infrastructure [CDN18].

C. Visual Studio Code

Visual Studio Code (VS Code) is a lightweight, open-source code editor developed by Microsoft that supports multiple programming languages, including Python, C++, and JavaScript. Designed for flexibility and performance, it offers a rich set of features such as intelligent code completion (IntelliSense), debugging tools, version control integration (Git), and a vast extension marketplace. VS Code is cross-platform, running on Windows, macOS, and Linux, and is highly customizable to fit individual development needs. While it does not provide built-in cloud computing resources like Colab, it is widely used for local development and seamlessly integrates with external tools and environments. [VSC25]

4.2.2. Libraries

In this project, several important libraries were used to support the development and implementation of the system. Each library served a specific purpose, as described below:

A. Tensorflow

TensorFlow is a widely used open-source library for machine learning and deep learning. It offers powerful tools for building, training, and deploying neural network models. Its flexibility makes it suitable for a range of applications, including image recognition, natural

language processing, and more. It supports both training models and performing inference on them efficiently [TFD24].

B. Keras

Keras is a high-level API built in Python for constructing and experimenting with neural networks. It is known for being user-friendly and modular, allowing researchers and developers to quickly design and test deep learning models. Keras works seamlessly with TensorFlow, making model building more accessible]KER25].

C. Mediapipe

MediaPipe is a cross-platform framework developed for building multimedia machine learning pipelines. It's especially useful for real-time applications involving video or audio data. MediaPipe is commonly used for tasks such as hand tracking, face detection, and object tracking, due to its efficiency and accuracy [MED25].

D. NumPy

NumPy is an essential library for scientific computing in Python. It provides support for multi-dimensional arrays and a wide range of mathematical functions to perform efficient computations. NumPy is widely used in data analysis, simulations, and machine learning, and serves as a core dependency for many other scientific [NMP24].

E. Tkinter

Tkinter is the standard GUI toolkit for Python, used to create graphical user interfaces. It allows developers to design interactive windows, buttons, labels, and other components easily. Its simplicity and cross-platform compatibility make it ideal for building basic GUI applications [PYD24].

F. PIL (Python Imaging Library)

The Python Imaging Library, or PIL, provides functionality for opening, manipulating, and saving many different image formats. It is particularly useful for basic image processing tasks and offers efficient tools for handling pixel-level image data [PIL24].

G. Moviepy

MoviePy is a Python library for video editing. It allows for tasks such as trimming, combining clips, adding text, creating transitions, and generating video effects. It supports most common video formats and can also be used to generate GIFs. MoviePy is suitable for both simple and complex video processing tasks [MOV24].

H. CAMeL Tools

CAMeL Tools is an open-source NLP toolkit designed specifically for Arabic. It offers features such as morphological analysis, and grammar correction. In this project, it was used to analyze and correct the grammatical structure of Arabic sentences after generation, significantly enhancing output fluency [HAB20].

I. Arabic Spellcheckers

Various Arabic spellchecking tools were integrated to detect and correct spelling errors in real time during user input. These tools offered word suggestions while typing and supported grammar correction tools in maintaining high linguistic quality at the word level.

J. LanguageTool

LanguageTool is an open-source grammar and spell checker that supports multiple languages, including English. In this work, it was used to automatically identify and fix grammatical and typographical errors in English translations, improving clarity and correctness [NAB03].

K. Natural Language Toolkit

NLTK is a widely used Python library for English NLP tasks. It was applied for basic word-level operations such as tokenization, spellchecking, and preprocessing, helping to validate and refine English sentence construction [BKL09].

4.3. System overview

This section is dedicated to presenting the interfaces of our system HandSay and the function of each. Our system includes three different interfaces, starting with the first interface that appears when the application is launched. This interface allows the user to select the sign language they want to work with, either Arabic or American. After selecting the sign

language, the user is directed to the main interface dedicated to translating the chosen sign language, which contains all the necessary features and functions.

Figure 4-1 Home page of our system.

After the sign language is selected, the basic interface shown in the image below appears.

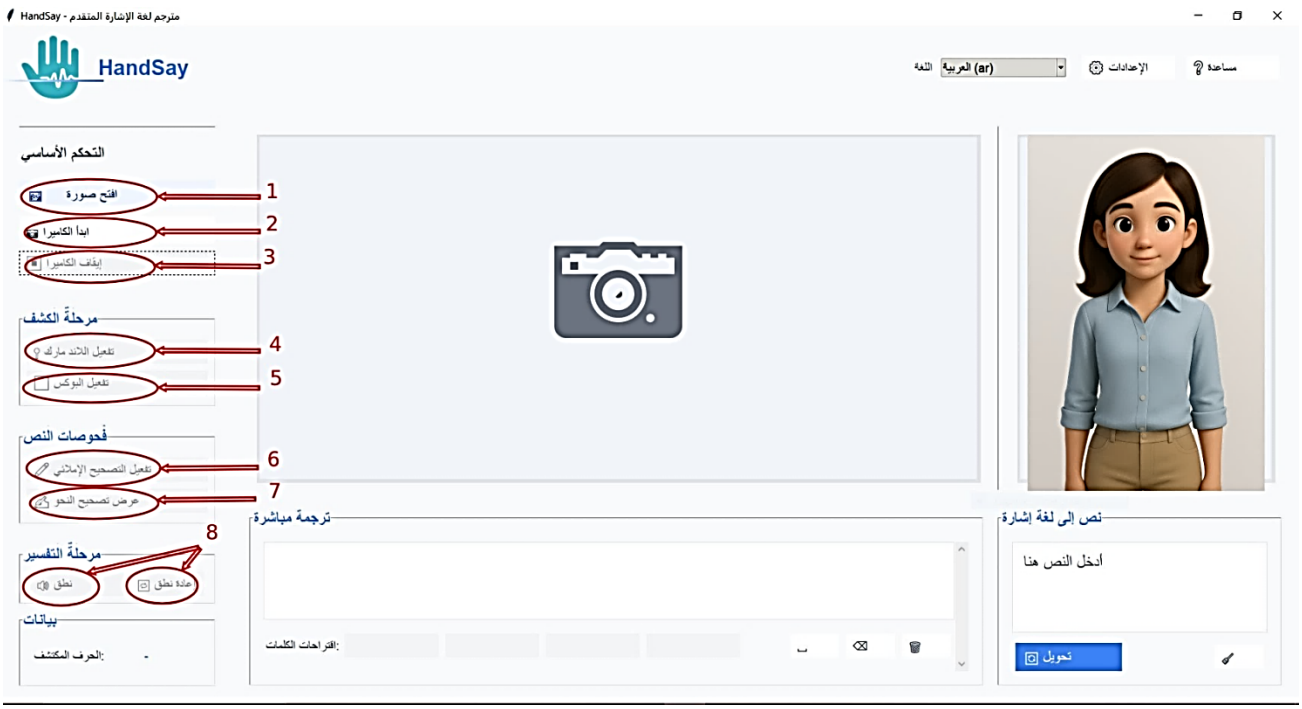

Figure 4-2 Basic interface of our system.

The main modules of our application's interface are structured into four functional phases, each serving a distinct purpose:

1. Open Image: Allows the user to load an image containing a sign for analysis.
2. Start Camera: Starts the live camera stream to detect hand signs in real time.
3. Stop Camera: Stops the live camera stream whenever the user wants.
4. Enable Landmark: Displays detected hand landmarks on thescreen for
5. Enable Box: Draws bounding boxes around detected hand regions
6. Enable Spell Checking: Automatically corrects spelling errors in the Recognized text.
7. Display Grammar Correction: Highlights grammatical errors and suggests fixes using built-in NLP tools.
8. Speaking Avatar: Uses a virtual avatar to speak the translated text, giving both Audio and visual feedback to the user.

4.4. Usage scenario

This section presents the user journey within the Handsay system, starting from launching the application and concluding with the final spoken translation of detected signs. The system is designed to offer a seamless and intuitive experience, guiding users through each phase with minimal effort while ensuring accurate and meaningful translations. Upon opening the application, the user first selects the appropriate sign language from the home interface, as previously described in Section 4.1. This selection leads to the main interface, where all essential translation tools are readily available. The journey begins with the user choosing the input method, either by uploading an image containing a sign or by activating the camera for live video capture. As illustrated in Figure 4-3 , when the live camera option is selected, the system enters real-time translation mode, automatically initiating sign detection and translation as the camera feed begins.

Figure 4-3 Input interface activating the live camera (detecting the letter "sheen" in real-time).

Once the input source is activated, the system transitions to the detection phase. The user is now able to activate the detection tools, which become available automatically. These tools use MediaPipe to detect hand landmarks and bounding boxes, enabling the system to precisely analyze the performed sign. As shown in Figure 4-4 and Figure 4-5, the interface highlights the critical hand regions to enhance detection accuracy.

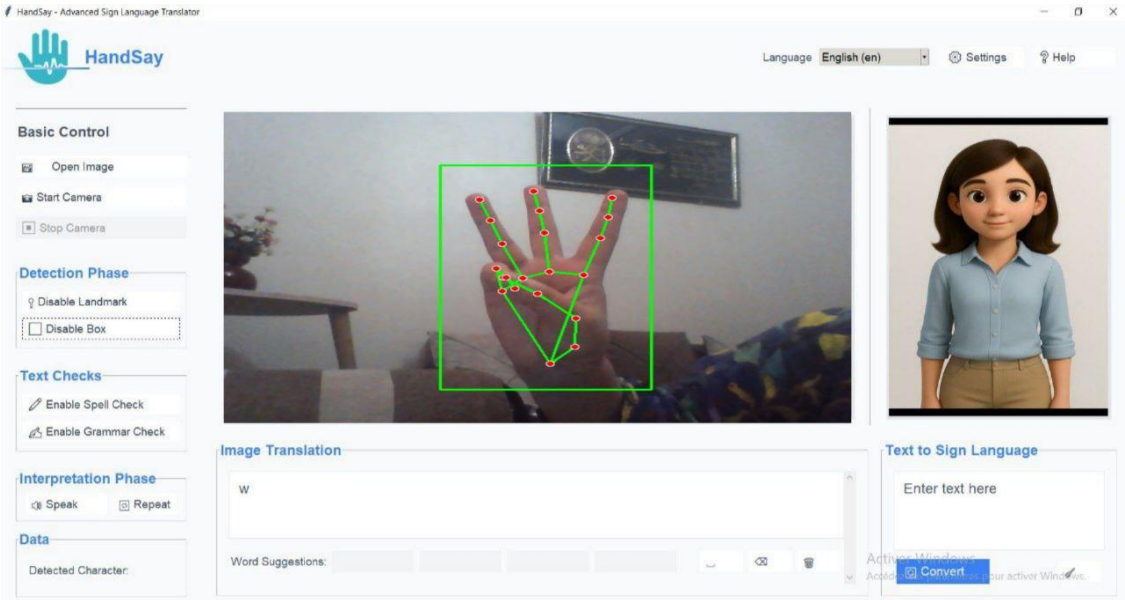

Figure 4-4 Detection of ASL hand landmarks and bounding box using MediaPipe.

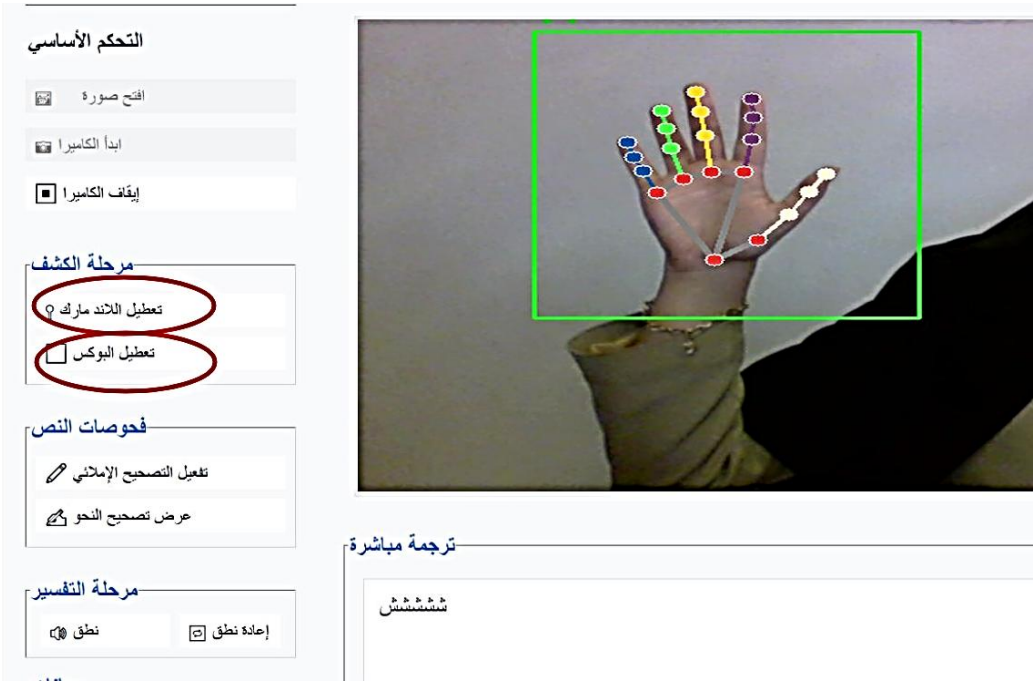

Figure 4-5 Detection of hand landmarks and bounding box using MediaPipe.

Following successful detection, the recognized sign is converted into text and displayed beneath the video area. At this point, the user enters the refinement phase, where integrated language tools help polish the output. The system suggests context-aware word alternatives and corrections as the user interacts with the text. This dynamic interaction ensures the translation is not only technically correct but also semantically appropriate. As seen in Figure 4-6 and Figure 4-7, users can engage with spelling suggestions in real-time via the “Enable Spell Checking” feature.

Figure 4-6 Real-time output display with word suggestions and spelling correction tools.

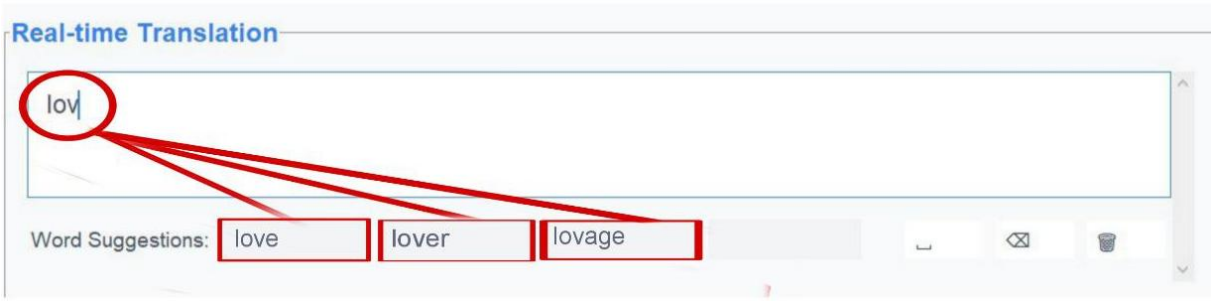

Figure 4-7 Real-time output display with American word suggestions.

In addition, the system includes grammar enhancement functionalities. As illustrated in Figure 4-8 and Figure 4-9, the system automatically detects and highlights grammar and syntax issues in the translated sentence. Users can review these suggestions and choose whether to apply them, ensuring the final output is grammatically sound and clearly expressed.

Figure 4-8 Exemples of Output text with grammar and spell checking tools.

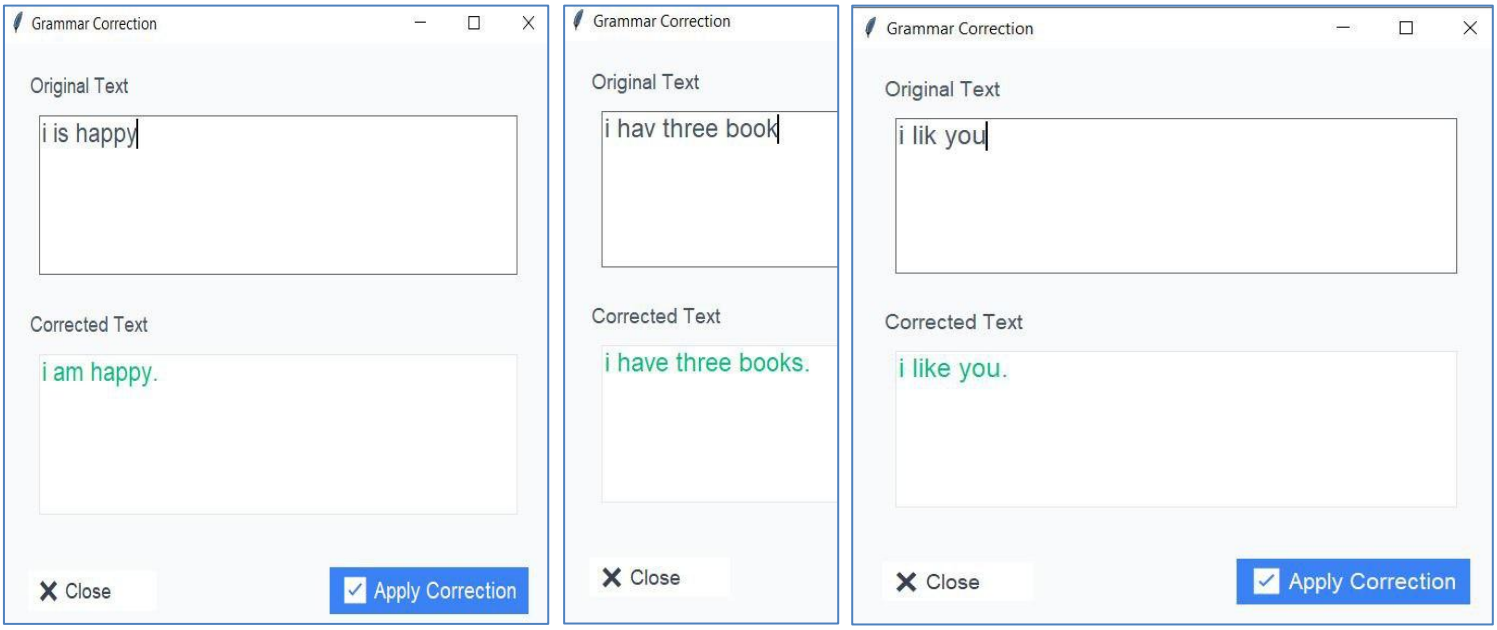

Figure 4-9 Examples of Output text in American English with grammar and spell checking tools.

Once the text has been refined, the final translation is presented in an interpretive, multimodal format. The system enters the interpretation phase, where a speaking avatar reads the translated sentence aloud. This avatar is synchronized to animate realistic lip movements in real-time, enhancing accessibility for users who benefit from both visual and auditory feedback. As demonstrated in Figure 4-14, this feature brings the translation to life through interactive voice and lip-sync animation.

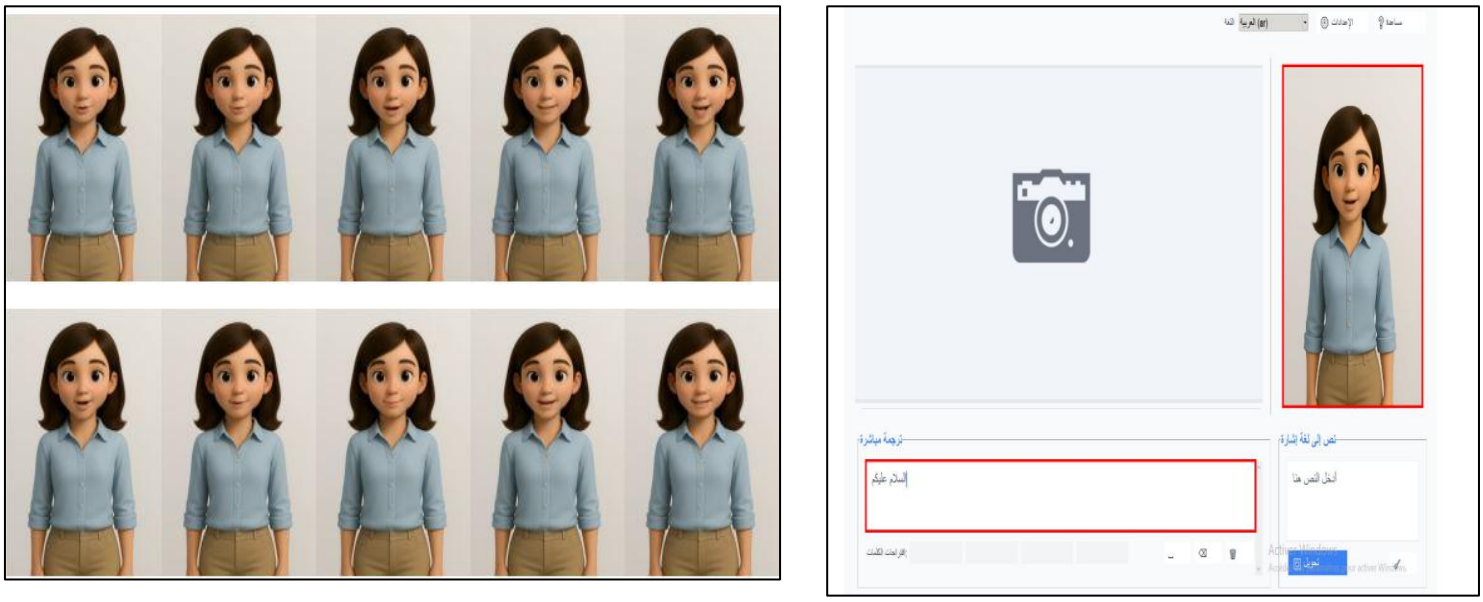

Figure 4-10 Speaking avatar reading the translated sentence with synchronized lip movement.

Furthermore, Handsay includes a text-to-sign visualization feature. When a user types a word, the system displays each of its letters sequentially in sign language to form the complete word. This is particularly useful for spelling practice and educational reinforcement. An example is shown in Figure 4-11, where the system spells out the word "Ahlan" (أهلاً) letter by letter in sign language. In another example, as shown in Figure 4-12, the system spells the word 'Hello' letter by letter using sign language.

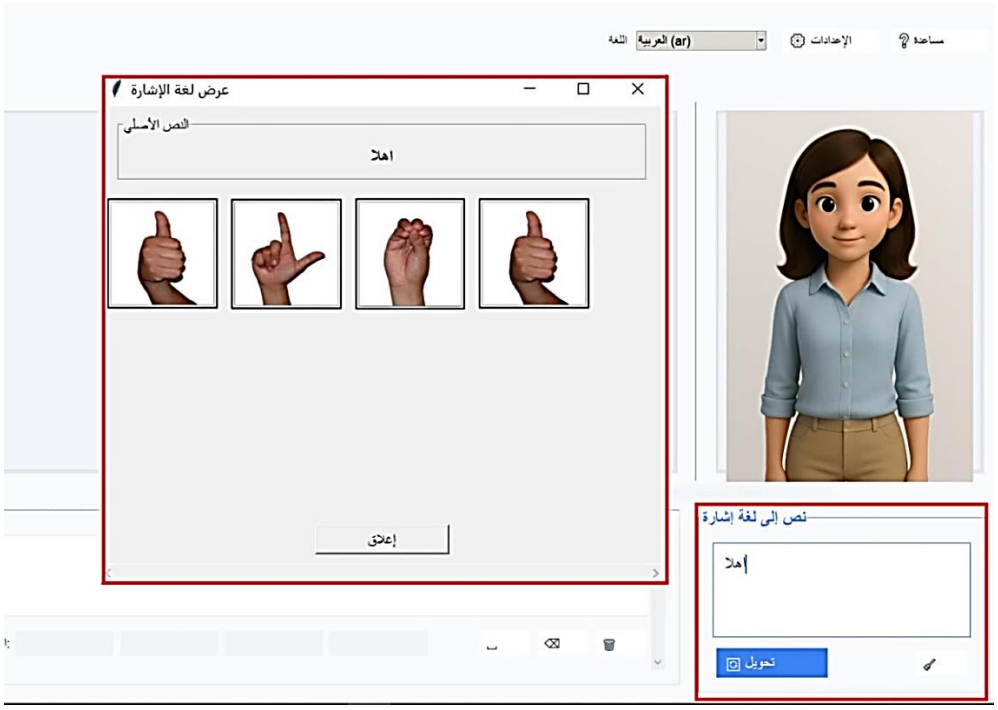

Figure 4-11 Sequential letter visualization to form the full word in sign language.

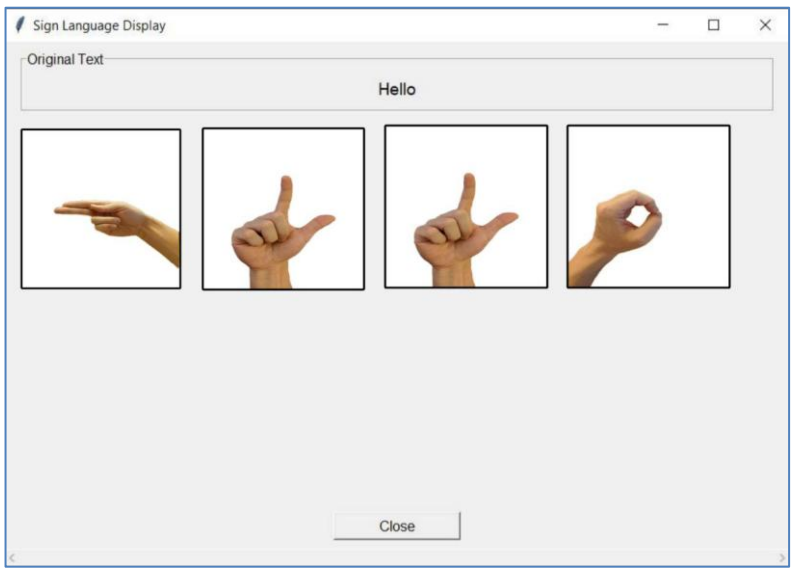

Figure 4-12 Sequential letter visualization to form the full word in American Sign Language.

At the top of the application interface, three key buttons provide quick access to essential customization and guidance features, as shown in Figure 4-13.

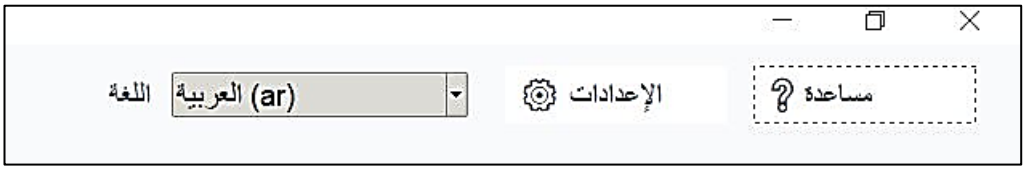

Figure 4-13 *T*op bar settings: language selection, help menu, and sign language preferences.

1. **Help Button:** This button opens a comprehensive help section containing detailed usage instructions, step-by-step guidance, and tips to assist users in navigating the application and utilizing its full functionality.
2. **Interface Language Button**: Allows users to switch the application interface language between Arabic and English, offering a bilingual experience to accommodate different user preferences.
3. **Settings Button:** Provides access to two critical customization options:
    A. Sign Language Selection: Users can choose the target sign language for recognition, selecting either Arabic Sign Language or American Sign Language (ASL) depending on their needs.
    B. Model Selection: Users can select the detection model to be used during translation. Available options include YOLOv8, CNN, and CNN MobileNetV2, allowing the user to adapt the system's performance and accuracy based on available resources or preferences.

As shown in Figure 4-14, the settings window offers a simple and clear interface for configuring both the sign language and the detection model.

Figure 4-14 Settings interface showing options for model selection and sign language configuration.

This scenario illustrates how the system delivers a cohesive and flexible user experience—combining ease of input, accurate analysis, and clear, interpretable output. The same steps and features demonstrated for Arabic Sign Language have also been applied to support English Sign Language, ensuring a consistent and inclusive translation experience across both languages.

4.5.Model Performance and Analysis

In this section, we present the results across three main areas: result presentation, analysis, and a comparison between the models we used and other related works with similar characteristics. Due to the limitations of the available dataset and the computational constraints, we focused exclusively on alphabet letters in both Arabic and English, rather than using predefined words. This approach offered greater flexibility, allowing users to construct any sentence by combining individual letters. It reflects both the nature of the dataset and the capabilities of our system during development.Model Results.

4.5.1. Model Results

Here, we present the results of each Arabic and American Sign Language recognition model. To evaluate the performance of these models, we use standard metrics derived from the confusion matrix, which is composed of four components:

- **True Positives (TP):** The model correctly predicts a label that is present in the ground truth.
- **True negatives (TN):** The model correctly ignores a label that is not present in the ground truth.
- **False positives (FP):** The model incorrectly predicts a label that is not present in the ground truth.
- **False negatives (FN):** The model fails to predict a label that is actually present in the ground truth.

These metrics can be presented as the following:

Accuracy: Accuracy measures the proportion of total correct predictions (both true positives and true negatives) to the overall number of predictions. A higher accuracy indicates better overall model performance:

$$Accuracy = \frac{TP+TN}{TP+TV+FP+FN} \qquad (1)$$

Precision: Precision evaluates the proportion of correctly predicted positive instances out of all instances that the model predicted as positive. It reflects the model's ability to avoid false positives :

$$Precition = \frac{TP}{TP+FP} \qquad (2)$$

Recall: Recall (also known as sensitivity or true positive rate) measures the proportion of correctly predicted positive instances out of all actual positive instances. It evaluates the model's ability to capture all relevant cases:

$$Recall = \frac{TN}{TN+FP} \qquad (3)$$

F1 score: The F1 score is the harmonic mean of precision and recall. It is particularly useful in cases of class imbalance, as it provides a single score that balances both false positives and false negatives:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \qquad (4)$$

Mean Average Precision (mAP): is a key metric used to evaluate the performance of computer vision models. It is calculated as the mean of the Average Precision (AP) values

across all classes within a model. mAP allows for effective comparison between different models on the same task or between various versions of the same model. The value of mAP ranges from 0 to 1, with higher values indicating better performance in object detection or classification tasks.

Loss: The loss function quantifies how well or poorly a model's predictions align with the actual outcomes. It measures the discrepancy between the predicted values and the true labels. In most machine learning algorithms, the objective is to minimize this loss during the training phase, thereby enhancing the model's predictive accuracy over time.

In this study, the model was trained on multiple distinct classes. To evaluate its performance, we employed the confusion matrix, which provides a detailed representation of the model's ability to distinguish between different classes. It reports the number of true positives, true negatives, false positives, and false negatives for each class, offering insight into the model's classification accuracy.

A. Arabic sign results

a) YOLOv8 model

Using the YOLOv8 model for comparison purposes, we achieved the following results, based on the **ArSL21L dataset [AS21]**, where the matrix of the 32 letters is presented in Figure 4-15.

Figure 4-15 Result of Confusion Matrix of our algorithm.

The training result of this model is shown in **Figure 4-16** and **Table 4-2**.

Table 4-2 the training result statistics of Arabic sign language Yolov8.

Figure 4-16 The training results of YOLOv8.

b) CNN1 (MobileNet) model

The model demonstrated strong performance,

(a) Accuracy curve (b) Training loss curve

Figure 4-17. Accuracy and training loss Curve of CNN1 model.

Below, we show the confusion matrix (see Figure 4-18) for our model that displays the true label compared to the predicted label shown in the figure below:

Figure 4-18 Confusion Matrix of CNN1 model.

c) CNN-LSTM

This experiment evaluated a CNN+LSTM model using sequences

.

(a) Accuracy curve (b) Training loss curve

Figure 4-19 *Accuracy and training loss Curve of CNN+LSTM model.*

d) CNN (Resnet)

The dataset (Arabic Sign Language Detection) [AS21],

Figure 4-20 Result of Confusion Matrix of our algorithm.

B. American results

a) YOLOv8 model

Using the YOLOv8 model for comparison purposes, we achieved the following results, based on the **ASL dataset [MSK24]**, where the matrix of the 26 letters is presented in Figure 4-21.

Figure 4-21 Result of Confusion Matrix of our algorithm.

The training result of this model is shown in Figure 4-22 and Table 4-3.

Table 4-3 the training result statistics of American Sign Language Yolov8.

Figure 4-22 The training results of YOLOv8.

b) CNN1 (MobileNet) model

This section aims to analyze the model's performance based on the loss and accuracy curves shown in Figure 4-23.

Figure 4-23 Accuracy and training loss Curve of MobileNet model.

In Figure 4-24, we show the confusion matrix for our model that displays the true label compared to the predicted label shown in the figure below:

Figure 4-24 Confusion Matrix of *MobileNet* model.

c) CNN2 model

This section aims to analyze the model's performance based on the loss and accuracy curves shown in Figure (4-25).

Figure 4-25 *Accuracy and training loss Curve of CNN2 model.*

Below, we show the confusion matrix for our model that displays the true label compared to the predicted label shown in the figure below:

Figure 4-26 *Confusion Matrix of CNN2 model.*

4.6. Result discussion

This section presents a comparison of the performance of our proposed models with related works, specifically focusing on Arabic and English sign language recognition using CNN, YOLO, and CNN+LSTM architectures. All models are trained and evaluated using static images only, without any video sequences.

4.6.1. Comparative Results

This section presents a comparative evaluation of the performance of the models we implemented for Arabic and American Sign Language recognition tasks.

4.6.2. Challenges We Faced

- **Scarcity of Arabic Sign Language Datasets** We encountered difficulty in obtaining high-quality Arabic sign language datasets that are diverse in terms of background, lighting, and camera angles. This lack of diversity limited the models' ability to generalize

4.6.3. Future Steps

We have already started building a **custom dataset** that includes Individual letters, Expanding the project to cover **sign languages across the Maghreb region**, in order to promote greater accessibility and inclusion for the hearing-impaired community in North Africa.

4.7. Conclusion

In this chapter, we presented the implementation details of our system for hand detection and sign language recognition in both Arabic and American Sign Language. The system supports two independent modes: real-time translation, where users perform gestures in front of the camera for immediate recognition using Mediapipe and deep learning models (CNN, LSTM, YOLOv8, and a hybrid CNN-LSTM); and static image translation, where users can upload an image of a hand sign to be processed separately. Experimental results showed high accuracy in both scenarios, confirming the effectiveness of the proposed system.

General Conclusion

The field of sign language recognition has seen significant advancements, particularly in English sign language, where rich and well-structured datasets have enabled the training of accurate and effective models. In contrast, Arabic sign language still suffers from a noticeable lack of advanced research and resources, especially in recognizing full words and sentences, which presents a real challenge to developing systems capable of supporting comprehensive communication in Arabic. In this project, a system was designed to enable real-time recognition of alphabet signs in both English and Arabic, allowing users to form words on the fly through a sequence of hand gestures. To achieve this, four deep learning-based models were developed: a YOLOv8 model for detecting static signs in real time, known for its fast and accurate performance; a Convolutional Neural Network (CNN) model; and a hybrid model combining both CNN and LSTM, which achieved the best performance due to its ability to capture both spatial and temporal features of the signs. These models were primarily trained using English sign language datasets to ensure high accuracy, and their effectiveness was also partially tested on Arabic letter data, which showed promising results during the initial stages of translation. Despite the limited resources available for Arabic sign language, the system demonstrated strong performance in detection, classification, and speed, making it a practical and efficient tool for supporting communication for individuals with hearing impairments. As future work, a custom Arabic Sign Language dataset is already being developed, including static images of isolated letters and video recordings of commonly used words, with multiple samples collected per class to represent realistic variations in sign performance. This dataset is expected to improve the system's ability to provide accurate and real-time translation of Arabic Sign Language at both word and sentence levels. There are also plans to expand the system to support Algerian Sign Language (LSA) by developing a specialized dataset that reflects regional dialects and local signs used across Algeria. Additionally, the system's English Sign Language capabilities are being extended to ensure usability in multilingual environments and enhance inclusivity in various educational and communicative contexts.

Bibliography

[AAM20] Ahmed, A. M., Abo Alez, R., Tharwat, G., Taha, M., Belgacem, B., & Al Moustafa, A. M. (2020). Arabic sign language intelligent translator. The Imaging Science Journal, 68(1), 11-23.

[AAM21] Ahmed, A. M., Tharwat, G., Elnakib, A., & Hassanien, A. E. (2021). Arabic sign language recognition system for alphabets using machine learning techniques. Journal of Electrical and Computer Engineering, 2021, 1-15.

[AA20] Aly, S., & Aly, W. (2020). DeepArSLR: A novel signer-independent deep learning framework for isolated Arabic sign language gestures recognition. IEEE Access, 8, 83199-83212.

[AC21] Alani, A. A., & Cosma, G. (2021). ArSL-CNN: a convolutional neural network for Arabic sign language gesture recognition. Indonesian Journal of Electrical Engineering and Computer Science, 22.

[AHL23] Ahli, M. E. M. (2023). Towards a Reliable Machine Learning-based Model Designed for Translating Sign Language Videos to Text. Rochester Institute of Technology. [Online].

[AMA24] Ahmadi, S. A., Muhammad, F. D., & Others. (2024). CNN-TCN: Deep Hybrid Model Based on Custom CNN with Temporal CNN to Recognize Sign Language. Journal of Disability, ScienceOpen.

[AMH20] Al-Hammadi, M., Muhammad, G., Abdul, W., Alsulaiman, M., Bencherif, M. A., & Mekhtiche, M. A. (2020). Hand gesture recognition for sign language using 3DCNN. IEEE Access, 8, 79491-79509.

[AJH01] Al-Jarrah, O., & Halawani, A. (2001). Recognition of gestures in Arabic sign language using neuro-fuzzy systems. Artificial Intelligence, 133(1-2), 117-1.

[AMJ24] Al Moustafa, A. M. J., & Others. (2024). Automatic Arabic Sign Language Recognition Using Image Processing and Machine Learning Techniques. Indian Journal of Computer Science and Engineering (IJCSE), 15.

[ATE22] Altememe, M. S., & El Abbadi, N. K. (2022). Gesture interpreting of alphabet Arabic sign language based on machine learning algorithms. 2022 Iraqi International Conference on Communication & Information Technologies (IICCIT), IEEE.

[AR17] Arora, S., & Roy, A. (2017). Recognition of sign language using image processing. International Journal of Engineering and Technology (IJET), 9(2), 164-176. https://doi.org/10.14419/ijet.v9i2.8042.

[BAA24] Baihan, A., Alutaibi, A. I., Alshehri, M., & Sharma, S. K. (2024). Sign language recognition using modified deep learning network and hybrid optimization: A hybrid optimizer (HO) based optimized CNNSa-LSTM approach. King Saud University & Majmaah University.

[BHI22] Bala, D., Hossain, M. A., Islam, M. A., Mynuddin, M., Hossain, M. S., & Abdullah, M. I. (2022). Effective recognition system of American Sign Language alphabets using machine learning classifiers, ANN, and CNN. IEEE Conference on Computational Intelligence, 1-10.

[CHK20] Camgoz, N. C., Hadfield, S., Koller, O., & Bowden, R. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. arXiv preprint, arXiv:2003.13830. Retrieved from https://arxiv.org/abs/2003.13830.

[GLM19] G. Latif, N. Mohammad, J. Alghazo, R. AlKhalaf, and R. AlKhalaf, "Arasl: Arabic alphabets sign language dataset," Data in brief, vol. 23, p. 103777, 2019.

[GSS19] Gaikwad, S., Shetty, A., Satam, A., Rathod, M., & Shah, P. (2019). Recognition of American Sign Language using image processing and machine learning. International Journal of Computer Science.

[GBG25] Goel, R., Bansal, S., & Gupta, K. (2025). Improved feature reduction framework for sign language recognition using autoencoders and adaptive Grey Wolf Optimization. Scientific Reports, 15, 2025.

[HT21] Halder, A., & Tayade, A. (2021). Real-time vernacular sign language recognition using Mediapipe and machine learning. International Journal of Research Publications, 7(1). Retrieved from https://www.ijrpr.com.

[HSA24] Hassan, M. A., Sabri, A. A., & Ali, A. H. (2024). Detection of Arabic sign language by machine learning techniques with PCA and LDA. Engineering and Technology Journal. Retrieved from iasj.net.

[HEB25] Herbaz, N., El Idrissi, H., & Badri, A. (2025). Advanced sign language recognition using deep learning: A study on Arabic sign language (ArSL) with VGGNet and ResNet50 models. DOI: 10.21203/rs.3.rs-5822261/v1.

[JK18] Joze, H. R. V., & Koller, O. (2018). MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. arXiv preprint arXiv:1812.01053.

[KG24] Kaur, K., & Garg, R. (2024). AESRM - Automatic English Sign Language Recognition with Machine Learning Techniques. IEEE TENCON 2024.

[KZM24] Khattab, M. M., Zeki, A. M., Matter, S. S., Abdella, M. A., & Others. (2024). Alphabet Recognition in Arabic Sign Language: A Machine Learning Perspective. Journal of the Faculty of Arts.

[KTD16] Kumar, A., Thankachan, K., & Dominic, M. M. (2016). Sign language recognition. In 3rd IEEE International Conference on Recent Advances in Information Technology (RAIT).

[LK99] Lee, H. K., & Kim, J. H. (1999). An HMM-based threshold model approach for gesture recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(10), 961-973.

[MAB23] M. Al-Barham, "RGB Arabic alphabets sign language dataset," 2023.

[MTA25] Miah, A. S. M., Tomioka, Y., Almehedihasan, M., & Shin, J. (2025). Hand gesture recognition for multi-culture sign language using graph and general deep learning network. [Journal/Conference Name]. Japan Society for the Promotion of Science (JSPS) KAKENHI, Grant JP23H03477.

[MRM22] Mitra, D., Rakshit, P., Mazumder, S., Dutta, A., Manna, S., Das, S., & Banik, S. (2022). Sign Language Recognition with Machine Learning. In Proceedings of Intelligent Computing, Devices, and Networking. Springer.

[MK21] Mohammed, R. M., & Kadhem, S. M. (2021). A review on Arabic sign language translator systems. Journal of Physics: Conference Series, 1897(1), 012002.

[MAM24] Moustafa, A. M. J., et al. (2024). Automatic Arabic Sign Language Recognition Using Image Processing and Machine Learning Techniques. Indian Journal of Computer Science and Engineering (IJCSE), vol. 15.

[NRE22] Natarajan, B., Rajalakshmi, E., Elakkiya, R., & Others. (2022). Development of an end-to-end deep learning framework for sign language recognition, translation, and video generation. IEEE.

[NWS24] Nurhadi, N., Winanto, E. A., Said, R. M., Jasmir, & Afuan, L. (2024). Pattern classification sign language using feature descriptors and machine learning. Jurnal Teknik Informatika (JUTIF), 5(2), 349-356. DOI: 10.52436/1.jutif.2024.5.2.1228.

[ONB24] Oulad-Naoui, S., Ben-Abderrahmane, H., & Others. (2024). An LSTM-based System for Dynamic Arabic Sign Language Recognition. International Conference on Advanced Technologies and Applications. Atlantis Press.

[PWP24] Paul, S. K., Walid, M. A. A., Paul, R. R., Uddin, M. J., & Others. (2024). An Adam-based CNN and LSTM approach for sign language recognition in real-time for deaf people. Bulletin of Electrical Engineering and Informatics. Retrieved from beei.org.

[RMA23] Rawf, K. M. H., Mohammed, A. A., Abdulrahman, A. O., Abdalla, P. A., & Ghafoor, K. J. (2023). A comparative technique using 2D CNN and transfer learning to detect and classify Arabic-script-based sign language. Acta Inform Malays, 7(1), 66.

[RSH14] Riad, A. M., Elminir, H. K., & Shohieb, S. M. (2014). Hand gesture recognition system based on a geometric model and rule-based classifier. British Journal of Applied Science & Technology, 4(9), 1432-1444.

[RW18] RWTH-PHOENIX-Weather-2014T. (2018). RWTH-PHOENIX-Weather-2014T dataset. Retrieved from https://www-i6.informatik.rwth-aachen.de/~koller/RWTH-PHOENIX-2014-T/

[SMT24] Said, Y., Boubaker, S., Altowaijri, S. M., Alsheikhy, A. A., & Atri, M. (2025). Adaptive Transformer-Based Deep Learning Framework for Continuous Sign Language Recognition and Translation. Mathematics, 13(6), 909. https://doi.org/10.3390/math13060909.

[SMH21] Shin, J., Matsuoka, A., Hasan, M. A. M., & Srizon, A. Y. (2021). American Sign Language Alphabet Recognition by Extracting Features from Hand Pose Estimation. Sensors, MDPI.

[ZJ19] Zakariya, A. M., & Jindal, R. (2019). Arabic sign language recognition system on smartphone. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE.

[SD25] ScienceDirect. *Unsupervised Learning – an overview*. Retrieved May 10, 2025

[SAN20] Sandra. *(2020).* **Arabic Alphabet in Sign Language** [Image]. Wikimedia Commons. Available: https://commons.wikimedia.org/wiki/File:الأبجدية_العربية_بلغة_الإشارة_(Arabic_Alphabet_in_Sign_Language).jpg

[WU24] Wu, M. *(2024)*. **Gesture Recognition Based on Deep Learning: A Review**. *EAI Endorsed Transactions on e-Learning*, **10**.

[PBK20] A. Prikhodko, M. Grif, and M. Bakaev, *The five components of signs in sign languages*, in *Sign Language Recognition Based on Notations and Neural Networks*, Jun. 2020. Available: https://www.researchgate.net/figure/The-five-components-of-signs-in-sign-languages_fig1_348346456

[AA23] H. Alawadh and A. Altamimi, *Sign Language Recognition Systems: A Decade Systematic Literature Review*, Archives of Computational Methods in Engineering, 2023.

[GAS09] Garg P, Aggarwal N, Sofat S (2009) Vision based hand gesture recognition. World Acad Sci Eng Technol 49(1):972–977

[BKA15] Badhe PC, Kulkarni V (2015) Indian Sign Language translator using gesture recognition algorithm. In: Proceedings of IEEE international conference on computer graphics on vision and information security (CGVIS), Bhubaneshwar, India, pp 195–200

[ASL] ASL University, *Fingerspelling: Introduction*. [Online]. Available: http://www.lifeprint.com/asl101/fingerspelling/fingerspelling.htm

[EKH06] K. El-Darymli, O.O. Khalifa, and H. Enemosah, *Speech to Sign Language Interpreter System (SSLIS)*, presented at the IEEE International Conference on Computer and Communication Engineering (ICCCE'06), Kuala Lumpur, Malaysia, Jan. 2006. Available: https://www.researchgate.net/publication/236634112_Speech_to_Sign_Language_Interpreter_System_SSLISResearchGate

[DK13] Dour S, Kundargi M (2013) Design of ANFIS system for recognition of single hand and two hand signs for Indian Sign Language. Int J Appl Inf Syst, pp 18–25

[AUD16] Amrutha CU, Davis N, Samrutha KS, Shilpa NS, Chunkath J (2016) Improving language acquisition in sensory defcit individuals with mobile application. Procedia Technol 24:1068–1073

[WK21] Wadhawan, A., & Kumar, P. (2021). *Sign language recognition systems: A decade systematic literature review*. Archives of Computational Methods in Engineering, Springer.

[MDL14] Mohandes, M., Deriche, M., & Liu, J. (2014). *Image-based and sensor-based approaches to Arabic sign language recognition*. IEEE Transactions on Human-Machine Systems, 44(4), 551–557. https://doi.org/10.1109/THMS.2014.2313625

[RAZ14] Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: An astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 806–813. [**Google Scholar**]

[KRI12] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 26th Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105. [**Google Scholar**] [**CrossRef**]

[LAW97] Lawrence, S.; Giles, C.L.; Tsoi, A.C.; Back, A.D. Face recognition: A convolutional neural-network approach. IEEE Trans. Neural Netw. 1997, 8, 98–113. [**Google Scholar**] [**CrossRef**]

[GRA13] Graves, A.; Mohamed, A.R.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649. [**Google Scholar**]

[GJ14] Graves, A.; Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1764–1772. [**Google Scholar**]

[TB07] Trivino, G., & Bailador, G. (2007, June). Linguistic description of human body posture using fuzzy logic and several levels of abstraction. In 2007 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (pp. 105-109). IEEE.

[ZCL20] Zhou, Z., Chen, K., Li, X. et al (2020). Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. Nat Electron, 3(9), 571–578.

[MOT18] Mori, Y., & Toyonaga, M. (2018, December). Data-glove for japanese sign language training system with gyro-Sensor. In 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS) (pp. 1354-1357). IEEE.

[AZS18] Ahmed MA, Zaidan BB, Zaidan AA, Salih MM, Lakulu MM (2018) A review on systems based sensory gloves for sign language recognition state of the art between 2007 and 2017. Sensors 18(7):2208

[LHL22] Li, J.; Li, C.; Han, J.; Shi, Y.; Bian, G.; Zhou, S. Robust Hand Gesture Recognition Using HOG-9ULBP Features and SVM Model. Electronics 2022, 11, 988. https://doi.org/10.3390/electronics11070988

[MS09] Murthy, G.R.S., & Jadon, R.S. A review of vision-based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, vol. 2(2), pp. 405-410, Dec. 2009. [GAS09] P. Garg, N. Aggarwal and S. Sofat. (2009). "Vision Based Hand Gesture Recognition," World Academy of Science, Engineering and Technology vol. 49, pp. 972-977.

[GW] Gesture Wikipedia website. http://en.wikipedia.org/wiki/Gesture

[MA07] S. Mitra, and T. Acharya. (2007, May). “Gesture Recognition: A Survey” IEEE Transactions on systems, Man and Cybernetics, Part C: Applications and reviews, vol. 37 (3), pp. 311 324, available: doi: 10.1109/TSMCC.2007.893280 .

[MG01] Moeslund, T.B., & Granum, E. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, vol. 81(3), pp. 231–268, Mar. 2001. Available: doi: 10.1006/cviu.2000.0897.

[LJ99] LaViola Jr., J.J. A survey of hand posture and gesture recognition techniques and technology. Master’s thesis, NSF Science and Technology Center for Computer Graphics and Scientific Visualization, USA, 1999.

[SM11] Meena, S. A study on hand gesture recognition technique. Master’s thesis, Department of Electronics and Communication Engineering, National Institute of Technology, India, 2011.

[HM10] Hasan, M.M.; Mishra, P.K. HSV brightness factor matching for gesture recognition system. *Int. J. Image Process.* 2010, *4*(5), 456–467.

[WEH19] Wehrmeyer E (2019) Rethinking handshape: a new notation system for sign language. Sign Lang Linguist 22(1):83–111

[Par06] R. Paradiso and D. De Rossi, “Advances in textile technologies for unobtrusive monitoring of vital parameters and movements,” in Proc. IEEE EMBS, 2006, pp. 392–395. [188] F.

[LST05] F. Lorussi, E. Scilingo, M. Tesconi, A. Tognetti, and D. De Rossi, “Strain sensing fabric for hand posture and gesture monitoring,” IEEE Trans. Inf. Technol. Biomed., vol. 9, no. 3, pp. 372–381, Sep. 2005.

[Ghu14] Ghunawat, M. R. (2014). *Multi-Point Gesture Recognition Using LED Gloves for Interactive HCI*. International Journal of Computer Science and Information Technology. Récupéré de Academia.edu.

[Her02] J. L. Hernandez-Rebollar, N. Kyriakopoulos, and R. W. Lindeman, “The AcceleGlove: A whole-hand input device for virtual reality,” in Proc. SIGGRAPH, 2002, p. 259

[CF03] Coiffet, P., & Burdea, G.C. *Virtual Reality Technology*. New York: Wiley, 2003.

[MH13] Mohandes, M.A. Recognition of two-handed Arabic signs using the CyberGlove. *Arabian Journal for Science and Engineering*, 38(3), 669–677, 2013. Available: https://doi.org/10.1007/s13369-012-0378-z.

[KT04] Kuroda, T., Tabata, Y., Goto, A., Ikuta, H., & Murakami, M. Consumer price data-glove for sign language recognition. In *Proc. Int. Conf. Disabil., Virtual Reality Associated Technol.*, 2004, pp. 253–258.

[LV99] LaViola, J.J. A survey of hand posture and gesture recognition techniques and technology. Brown Univ., Providence, RI, Tech. Rep. CS-99-11, Jun. 1999.

[DH16] Dobry, M.W., & Hermann, T. Differences in power distribution in the subsystems of the human–anti-vibration glove–tool system. *Vibrations in Physical Systems*, 27, 91–98, 2016. Available: https://bibliotekanauki.pl/articles/127968.

[CB07] Clues, L.M., Bromwich, D., & Jones, B. How gloves fail. *Journal of Occupational Health and Safety - Australia and New Zealand*, 23(3), 2007. Available: https://www.researchgate.net/publication/41213210_How_gloves_fail.

[CW02] Chen, W., Wang, S., & Li, Z. Research of using data glove in virtual assembly environment. In *Proceedings of the Second International Conference on Virtual Reality* (Vol. 4875). SPIE, 2002. Available: https://doi.org/10.1117/12.466727.

[KT04] Kuroda, T., Tabata, Y., Goto, A., Ikuta, H., & Murakami, M. Consumer price data-glove for sign language recognition. In *Proceedings of the 5th International Conference on Disability, Virtual Reality & Associated Technologies* (pp. 253–258), Oxford, UK, 2004. Available: https://www.researchgate.net/publication/228706126_Consumer_price_data-glove_for_sign_language_recognition.

[BW01] Bowman, D.A., Wingrave, C.A., Campbell, J.M., & Ly, V.Q. Using Pinch Gloves™ for both natural and abstract interaction techniques in virtual environments. Virginia Tech, 2001. Available: https://eprints.cs.vt.edu/archive/00000547/01/PinchGlovePaper.pdf.

[GE19] Görgü, E., & Erol, D. **(2019).** ***Sign language recognition with artificial neural networks using a novel dataset*****. Applied Sciences,** 9**(20), 4396. https://doi.org/10.3390/app9204396**

[DSD08] DiPietro, L., Sabatini, A. M., & Dario, P. (2008). *A survey of glove-based systems and their applications*. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(4), 461-482.

[FR21] Farooq, U., Rahim, M. S. M., Sabir, N., Hussain, A., & Abid, A. (2021). Advances in machine translation for sign language: approaches, limitations, and challenges. Neural Computing and Applications, 33(21), 14357 14399

[HL23] Luqman, H. (2023, January). ArabSign: A Multi-modality Dataset and Benchmark for Continuous Arabic Sign Language Recognition. In 2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG) (pp. 1-8). IEEE.

[CH11] H. Cooper, B. Holt, R. Bowden, "Sign Language Recognition", in *Visual Analysis of Humans: Looking at People*, Springer, 2011.

[DK10] Kelly, D. (2010). Computational Models for the Automatic Learning and Recognition of Irish Sign Language (Doctoral dissertation, National University of Ireland Maynooth).

[BC19] Borg, M., & Camilleri, K. P. (2019, May). Sign language detection "in the wild" with recurrent neural networks. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1637-1641. IEEE.

[DKYL20] N.-T. Do, S.-H. Kim, H.-J. Yang, and G.-S. Lee, "Robust hand shape features for dynamic hand gesture recognition using multi-level feature LSTM," *Appl. Sci.*, vol. 10, no. 18, p. 6293, 2020. https://doi.org/10.3390/app10186293

[LL19] Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences, 9*(20), 4396.

[ACT18] Alani, A.A., Cosma, G., Taherkhani, A., & McGinnity, T.M. (2018). Hand gesture recognition using an adapted convolutional neural network with data augmentation. *2018 4th International Conference on Information Management (ICIM)*, 5-12.

[IKK12] Ibraheem, N.A.; Khan, R.Z. Vision based gesture recognition using neural networks approaches: A review. *Int. J. Hum. Comput. Interact.* 2012, *3*(1), 1–.

[LCJ14] Liu, K., Chen, C., Jafari, R., & Kehtarnavaz, N. (2014, October). Multi-HMM classification for hand gesture recognition using two differing modality sensors. In *2014 IEEE Dallas Circuits and Systems Conference (DCAS)* (pp. 1-4). IEEE.

[GGL25] Google. Example Different Signs for the Same Word "Thank You" in Sign Language. Retrieved May 10, 2025.

[SS25] Signing Savvy. (2025). *Signs beginning with the letter U*. Retrieved May 10, 2025,

[NAJ25] Najib, F. M. (2025). Sign language interpretation using machine learning and artificial intelligence. *Neural Computing and Applications*, 37(2), 841–857.

[ABF05] Abdel-Fattah, M. A. (2005). Arabic Sign Language: A Perspective. *Journal of Deaf Studies and Deaf Education*, 10(2), 212–221.

[TJB24] T. Javed, *Pose Detection using Mediapipe Solutions: DabMove Detection*, Medium, Jan. 18, 2024.

[RG23] ResearchGate, *The 21 landmarks of the skeleton model of MediaPipe Hands*, 2023.

[VSC25] Microsoft. *Editing evolved – Visual Studio Code Documentation*. Available at: https://code.visualstudio.com/docs/editing/editingevolved [Accessed 13 June 2025].

[DHN25] Dang Hoang Nhan, *MobileNet Architecture*, [Online].

[EE20] Elatawy, S. M., Hawa, D. M., Ewees, A. A., & Saad, A. M. (2020). Recognition system for alphabet Arabic sign language using neutrosophic and fuzzy c-means. Education and Information Technologies, 25, 1035–1052. https://doi.org/10.1007/s10639-019-10053-5.

[GA20] Grassknoted, *"ASL Alphabet,"* Kaggle, 2020. Available at: [https://www.kaggle.com/datasets/grassknoted/asl-alphabet]

[MSK24] Moskwa, P., "American Sign Language (ASL) Dataset," GitHub, 2024.

[AS21] Batnasan, G., Gochoo, M., Otgonbold, M. E., Alnajjar, F., & Shih, T. K., *"ArSL21L: Arabic Sign Language Letter Dataset,"* Mendeley Data, v1, 11 Feb. 2022. DOI: 10.17632/f63xhm286w.1

[AS18] Latif, G., Mohammad, N., Alghazo, J., AlKhalaf, R., & AlKhalaf, R., *"ArASL: Arabic Alphabets Sign Language Dataset (ArSL2018),"* Mendeley Data, v1, 5 Nov 2018. DOI: 10.17632/y7pckrw6z2.1. متاح علىhttps://data.mendeley.com/datasets/y7pckrw6z2/1