TUNISIAN REPUBLIC
Ministry of Higher Education and Scientific Research
University of Carthage
National Institute of Applied Sciences and Technology

End Of the Year Project

4th Year

Computer Networks and Telecommunication

Arabic Sign Language Recognition

Prepared By

Issam JEBNOUNI

Hela KHADHAR

Imen BAJJAR

Nadia FRIKHA

Under the supervision of :

Professor. Hazar MLIKI

Academic Year : 2022 – 2023

Acknowledgments

We would like to give, by these few lines, our warmest thanks to all of those who helped us achieve this work.

We would like to thank Ms. Hazar Mliki for her enormous efforts to make this work done in a well planned and strategic path. Her encouragement, guidance, and advice helped us reach our goals and have the wanted results. We are really glad to work under her leadership.

Many thanks should also go to Ms. Rabaa Youssef who took the time and effort to read and examine this report.

Tunis, 27 May 2023.

Abstract

This project was done in the context of an end-of-year annual project.

In this report, we explain the steps taken to complete our work. We have implemented a model that recognized arabic sign language and translated the signs into words. This solution is designed to help deaf and mute people communicate with the outside world.

At first, we had to understand how sign language works and study the existing solutions and algorithms related to this project. Then, we worked on preprocessing the data, then implemented our model based on BiLSTM architecture. The final step was to evaluate the performance of our proposed model .

Keywords : CRISP-DM, Computer Vision, Hand Detection, Hand Tracking, Deep Learning, Classification, Mediapipe, BiLSTM

Table of contents

List of Figures

List of Tables

General Introduction

Artificial intelligence (AI) has become increasingly important in the last decade due to its ability to facilitate daily tasks, in normal life or in industries. It actually helps the process of industry transformation, solve complex problems and improve the quality of several tasks.

The positive impact of AI can be explained via the automation it provides, the personalization and the reliability of most of its results. It can even be relied-on in helpful aspects such as helping people with special needs communicate with the outsider world.

Computer vision can be defined as a multidisciplinary field of AI study that focuses on developing algorithms and techniques to enable computers to acquire, analyze, understand, and interpret visual information from images or videos. It involves the utilization of various image processing methods, pattern recognition, and machine learning algorithms to extract meaningful information and make intelligent decisions based on visual data.
AI technologies such as machine learning and computer vision can enable the creation of more accurate and efficient hand sign language recognition systems. These systems can be used to improve communication and accessibility for people who are deaf or hard of hearing by enabling them to communicate more effectively with the hearing world.

In this context, this project was done to implement a solution that detects sign language, specifically the arabic one, and translates it into written words.

Sign language recognition is the process of automatically identifying human gestures using their hands and other body parts features, then interpreting these gestures into the corresponding words.
The present manuscript is structured as follows :

In chapter 1, we introduce the chosen methodology CRISP-DM and its first phase, Business Understanding. In chapter 2, we present the Data Understanding and Preprocessing. In chapter 3, we present the built model and we evaluate it on the KArSL Dataset. Finally, we conclude with a general conclusion and some future works.

Chapter 1

Field Comprehension

Plan

1.1 Introduction

In this chapter we present the CRISP-DM methodology adopted in our work.
Next, we introduce the first step "Business Understanding" in which we explain the *Sign Language Recognition* subject and its related problematics. Finally we study the existing solutions to define our proposed method.

1.2 CRISPM-DM methodology

1.2.1 Definition

The Cross Industry Standard Process for Data Mining (CRISP-DM) includes descriptions of typical project phases and the tasks in each phase, and an explanation of the relationships between them. It is by far the most efficient method for all Data Mining and Data Science projects .
It breaks down into 6 steps [2] as shown in figure 1.1.

Figure 1.1: CRISP-DM steps [1]

1.2.2 Steps

1. **Business Understanding** : This phase focuses on understanding the business elements, objectives, requirements, and problems of the project that Data Mining aims to solve or improve.
2. **DATA Understanding** : This step drives the focus to identify, collect, and analyze the data sets and their quality that can help you accomplish the project goals.

3. **DATA Preparation** : It prepares the final Data Set for modeling by selecting, cleaning, classifying and especially recoding Data to make it compatible with the algorithms that will be used. The parametricity of the numerical data and their recoding into categorical data are extremely important All these data must indeed be centralized in a structured database called Data Hub.

4. **Modeling** : This step includes the choice, settings and testing of different algorithms as well as their chaining, which constitutes a model. This process is at first descriptive to generate knowledge, by explaining why things happened. It then becomes predictive by explaining what will happen in future situations.

5. **Evaluation** : It aims to assess the degree to which the model meets our objectives. It helps in deciding whether or not to deploy the model, or in improvement if it is necessary. At this step, the obtained models are tested in terms of robustness and accuracy.

6. **Deployment** : This step consists of a production launch for the final users of the end users of the models obtained. Its objective is to put the knowledge obtained by modeling, in an adapted form, and integrate it into the decision-making process. Depending on the objectives, the deployment can go from the simple generation of a report describing the knowledge obtained to the implementation of an application, allowing the use of the model obtained, for the prediction of unknown values of an element of interest.

1.2.3 Benefits

This methodology is cost-effective as it includes a number of processes to take out simple data mining tasks and the processes are well-established.It provides a uniform framework for planning and managing a project. Moreover, as a cross-industry standard, CRISP-DM can be implemented in any Data Science project irrespective of its domain.

1.3 Sign language

Sign languages (also known as signed languages) are languages that use the visual-manual modality to convey meaning, instead of spoken words. Sign languages are expressed through manual articulation in combination with non-manual markers and are full-fledged natural languages with their own grammar and lexicon. It is a visual means of communicating through **hand signals, gestures, facial expressions, and body language**. [3]

It represents the main form of communication for the Deaf and Hard-of-Hearing community, but sign language can be useful for other groups of people as well. People with disabilities including Autism, Apraxia of speech, Cerebral Palsy, and Down Syndrome may also find sign language beneficial for communicating.[4]

1.4 Problematic

Sign Languages are **not universal** and are usually not mutually intelligible. In fact, there is no single sign language used around the world. Like spoken language, sign languages developed naturally through different groups of people interacting with each other, so there are many varieties. There are somewhere between 138 and 300 different types of sign language used around the globe today. Each sign language has its own style and modifications and remains unique. So, there is a lack of understanding and knowledge of this language among the general population, which results in a communication barrier between the two groups.
This sign language recognition project aims to develop a system that can recognize Arabic sign language gestures and movements and translate them into spoken or written language, which would facilitate communication between deaf or hard-of-hearing individuals and hearing individuals who do not understand sign language.

1.5 Existing Solutions

In general, sign language recognition projects are composed of three main phases:(1) Hand Segmentation and Tracking,(2) Feature Extraction, and (3)Hand Gesture Classification Phase.
In the following subsections, we present the most commonly used solutions.

1.5.1 Existing Algorithms

The process is composed of three main phases[5]:

Phase 1: Hand Segmentation and Tracking

The hand segmentation and tracking phase begins with image pre-processing in order to be able to detect the human hand. Image pre-processing consists of several processes that may include image segmentation, image normalization, and image filtering. Image pre-processing is performed in order to identify the regions of interest of an object that is the human hand. This process is referred to as hand detection. The process of image pre-processing segmentation is crucial because it involves the

process of separating the hand from the background. Some features that can be used in segmentation include skin color, the shape of the hand, the motion of the hand and structural features of the hand. Algorithms involved in hand segmentation and tracking can be classified into several categories[6].

— **Threshold-based image segmentation algorithms:** Thresholding is used in different image segmentation techniques to separate the region of interest from the background. Thresholding is achieved by converting an image into a binary image by assigning each pixel in an image intensity values 1 to the hand, and 0 to the background. Among the thresholding techniques, the edge maximization technique provides the best performance[7].

— **Region-based image segmentation algorithms:** This process involves dividing the image into smaller segments that have a certain set of rules[8]. This technique applies an algorithm that divides the image into several components with common pixel features. The process looks out for chunks of segments within the image. Small segments can include similar pixels from neighboring pixels and subsequently grow in size.

— **Edge-based image segmentation algorithms** The edge detection method is mainly used to find the contour of an object in an image. This is achieved by varying the image brightness at the edges by thresholding techniques. Therefore, the edges are detected by the change in the grey tone or intensity value. Some notable types of edges includes step and line edges. Step edge is where the intensity varies abruptly from one value to the other confirming discontinuity. Line edges are where intensity values vary abruptly and resume the normal values after a small Euclidian distance[6].

— **Fuzzy theory-based image segmentation algorithms** Fuzzy set theory is a method of segmenting an image into multiple regions or objects using fuzzy logic. It is used to analyze and extract information from images that are inherently imprecise or uncertain. For example, in an image, it may be difficult to determine exactly which pixels belong to a particular object or feature, as there may be some degree of overlap or ambiguity. In this approach, the image is represented as a collection of fuzzy sets, which assigns each pixel in the image a degree of membership to a particular region or object.

— **Artificial neural network-based image segmentation algorithms** In ANN-based image segmentation, each pixel in the input image is represented by a neuron in the network. The ANN learns to recognize patterns and features that distinguish regions or objects in the image by adjusting the connections between neurons during training. Once trained, the model can

be used to segment new images by assigning each pixel a label indicating its region or object.

— **Partial differential equations-based image segmentation algorithms** These algorithms describe the behavior of functions as it involves over time. In fact, the image is treated as a function of space and time. They are used to model the underlying processes that determine the boundaries between different regions or objects. The PDEs typically include terms that control the smoothing and regularization of the image function. They are solved numerically using techniques such as finite differences or finite element methods, and the resulting solution is used to segment the image into different regions or objects.

Phase 2: Feature Extraction

The purpose of feature extraction is to find and isolate the features of vector points from the hand gesture patterns. The features extracted for hand gesture recognition include geometric features (fingertips, finger directions and hand contours), and non-geometric features(color, silhouette, and texture).

— **Color feature:** Color is one of the features that can be considered in gesture recognition. It consists of different components like hue, luminance, and saturation. There are also different mathematical models for representing the color, called color spaces which are used in different algorithms for gesture recognition. The color spaces can be classified into different categories. First, Red Green Blue (RGB) based color space, for instance, RGB and normalized RGB, second, Hue based color space, for instance, HSI, HSV, and HSL, finally Luminance based color space (YCbCr, YIQ, and YUV). The choice of color space depends on the skin detection algorithm and the gesture application.

— **Texture Feature :** These features are considered spatial texture feature extraction methods.The spatial texture features are extracted by computing the pixel statics or finding the local pixel structures in the original image domain.

— **Shape Feature :** We can easily recognize objects by their shapes. This feature's extraction can be performed by using the contour to extract the boundary and edges of an object or by using the regional information to extract.

Phase 3: Hand Gesture Classification Phase

Gesture classification is the last step in the gesture recognition process. This phase involves matching the current gesture features with the stored features. Classifiers can be categorized as conventional hand gesture classification algorithms and Neural Network-based :

— **Conventional hand gesture classification algorithms:**

- K-nearest neighbor
- Fuzzy clustering algorithm
- Mean shift clustering
- Linear Quadratic Estimation (LQE)
- Hidden Markov model.
- Histogram-based feature algorithms

— **Neural Network-based hand gesture Classification Algorithms**

- ANN: Artificial Neural Network interconnected processing components working together as one for a common objective. It can be used in the feature extraction phase of the gesture recognition system as well as the classification phase.
- RNN: Recurrent Neural Networks are a class of artificial neural networks that are designed to process sequential data by considering the order and dependencies among the elements in the sequence. RNNs are particularly effective for tasks such as natural language processing, speech recognition,...
- CNN: Convolutional Neural Network Similar to ANN. It can outperform other 2D shapes classification methods. performs both feature extraction and classification automatically through gradient-based learning methods.

1.5.2 Mediapipe vs Yolo

1.5.2.1 Mediapipe

Mediapipe is an open-source, cross-platform framework developed by Google that provides a variety of pre-built machine learning models and building blocks for developing computer vision and machine learning applications. It includes a wide range of tools and components for building pipelines to process real-time video and audio streams. Some of the key features of MediaPipe include:

— Support for a variety of platforms and programming languages, including C++, Python, and Java.

— A collection of pre-built models for tasks such as face detection, hand tracking, and object detection.

— Building blocks for processing and manipulating video and audio data, such as image and

audio encoders/decoders, filters, and feature extractors.

— Tools for designing, training, and deploying custom machine learning models.

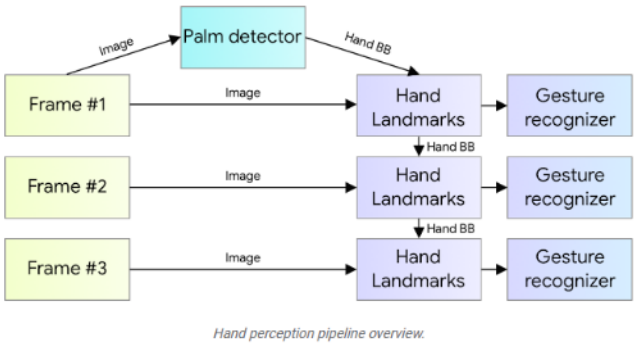

Figure 1.2: Hand perception pipeline overview

MediaPipe Hands implements the ML pipeline in Figure 1.2. It consists of three models for hand gesture recognition as follows: [9]

1. A palm detector model processes the captured image and turns the image with an oriented bounding box of the hand.

2. A hand landmark model processes a cropped bounding box image and returns 3D hand key points on hand.

3. A gesture recognizer that classifies 3D hand key points then configures them into a discrete set of gestures.

1.5.2.2 Yolo

YOLO is an object detection algorithm used in computer vision that was introduced in 2016. Instead of using region proposal networks like other object detection algorithms, YOLO predicts object classes and locations directly from a single pass through the neural network, which makes it significantly faster than other algorithms while still maintaining high accuracy. In fact, YOLO divides an image into a grid and applies a convolutional neural network (CNN) to each grid cell to predict the class probabilities and bounding boxes for any objects present in that cell.

1.5.2.3 Comparaison

Criteria	YOLO	Mediapipe
Algorithm	A real-time object detection algorithm for a wide range of objects, including hands.	A suite of computer vision tools developed by Google for gesture recognition, face recognition, motion detection, and more
Number of object detection	Multi-object	Single
Hand tracking	It can detect the hand but does not provide hand tracking.	It can detect and track the hand in real-time with high accuracy.
Detection accuracy	high detection accuracy but may lack accuracy in some cases It has difficulty detecting small or occluded objects, or objects that have similar features to the background.	high detection accuracy and can work in low light conditions or with difficult camera angles.

Table 1.1: Comparison table between YOLO and Mediapipe [10]

For conclusion, MediaPipe is more suitable than Yolo, as it provides advanced computer vision algorithms to track hand movements and positions in real-time, even in low light or complex background conditions.

1.6 Conclusion

In this chapter, we introduced the first step of CRISP-DM. In fact, we presented our context of study. Then we carried out a study of the existing solutions to finally select our proposed one.

Chapter 2

Data Comprehension and Preprocessing

Plan

2.1 Introduction

Data comprehension is a critical step of any data science project since the quality and relevance of the data can significantly impact the performance of the model. To achieve accurate and reliable results, it is essential to understand the data and its underlying characteristics to pre-process it correctly . Such a step helps making our data compatible with the used algorithms and extracting the best knowledge out of it.

In this chapter , we describe the used dataset. Next we present an analysis of our data and how we preprocessed it .

2.2 KArSL Dataset

Sign language is specific to each geographic region in the world. It actually differs from one country to another, as it shows a huge difference in body language and gestures used. In fact, it depends on the language spoken in each country.

KArSL is the largest video dataset for Word-Level Arabic Sign Language (ArSL) recognition, used in Arab countries. It consists of 502 isolated sign words performed by 3 professional signers[11]. Each word is repeated 50 times by each signer, which made it a dataset with a total of 75,300 samples.

2.3 Exploratory Data Analysis

In our project, we will be working on 100 classes, performed by 3 signers. Figure 2.1 presents an overview of 100 classes of KArSL.

	SignID	Sign-Arabic	Sign-English
0	71	هيكل عظمي	Skeleton
1	72	جمجمة	skull
2	73	عمود فقري	Backbone
3	74	قفص صدري	Chest
4	75	جهاز تنفسي	Respiratory device
...	...	...	...
95	166	يشم	inhale
96	167	يصعد	rise
97	168	ينزل	descend
98	169	يفتح	open
99	170	يقفل (يغلق)	close

Figure 2.1: Overview of 100 classes of KArSL

First we retrieve the *sequenceID* and *number of frames* of the videos in our dataset. Given the signer-id, the split wanted (train or test) as input, we get a dataframe containing features for each video.

As a final result, executing the *make-features-array()* function twice on the train and test data will return as a result test_features_df (c.f Figure 2.2) and train_features_df (c.f Figure 2.3) dataframes.

	SignID	signer_id	Sign-Arabic	Sign-English	SequenceID	num_frames
0	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_53_03)_c	47
1	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_54_36)_c	44
2	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_54_41)_c	48
3	71	1	هيكل عظمي	Skeleton	03_01_0071_(20_11_16_16_17_05)_c	46
4	71	1	هيكل عظمي	Skeleton	03_01_0071_(20_11_16_16_17_37)_c	46

Figure 2.2: Features extraction for the test subset

	SignID	signer_id	Sign-Arabic	Sign-English	SequenceID	num_frames
0	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_52_41)_c	29
1	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_52_44)_c	42
2	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_52_49)_c	44
3	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_52_53)_c	47
4	71	1	هيكل عظمي	Skeleton	03_01_0071_(01_12_16_15_52_58)_c	50

Figure 2.3: Features extraction for the train subset

Since we need to guarantee the conformity of dimensions for the tensors used in the training phase, we set a threshold for frame sampling by computing an average number of frames for the 100 classes.

```
total_test_frames = sum(test_features_df['num_frames'])
total_train_frames = sum(train_features_df['num_frames'])
total_test_videos = len(test_features_df)
total_train_videos = len(train_features_df)
f_avg=round((total_train_frames + total_test_frames) / (total_train_videos
+ total_test_videos))
f_avg
```

The computed value of f_avg is 48. So, for videos longer than 48 frames, we only keep the first 48 frames; and for videos shorter than 48, we pad with the last frame until we get 48 frames.

2.4 Data Preprocessing

2.4.1 Extracting the video features with mediapipe

After the comparaison between Mediapipe and Yolo solutions, we adopted the mediapipe because it is more suitable for a single person detection and it has pre-built models for hand and pose tracking

First, we looped through the frames using" OpenCV" and extracted pose and hands keypoints for each frame using mediapipe holistic model in order to obtain keypoints array for each video.

```
# Initialize the Mediapipe Holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking
_confidence=0.5) as holistic:
   # Loop through the video frames
   while True:
        # Read the next frame
        ret, frame = video.read()
        # Check if there are no more frames to read
        if not ret:
        break
        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        # Extract keypoints
        pose, lh, rh = extract_keypoints(results)
        # Add the keypoints to the list for this video
        pose_keypoints.append(pose)
        lh_keypoints.append(lh)
        rh_keypoints.append(rh)
        # Release the video file
        video.release()
```

After processing the frames and adjusting them to make the landmarks in the array relative to specific key points, we used the function *extract-keypoint()* to extract keypoints from the results object generated by the holistic model, such process make the model pass over the position of the signer in the frame and focus on the relative position of his hands and pose keypoints to the wrists and nose.

```
def adjust_landmarks(arr,center):
    # Reshape the array to have shape (n, 3)
    arr_reshaped = arr.reshape(-1, 3)
    # Repeat the center array to have shape (n, 3)
    center_repeated = np.tile(center, (len(arr_reshaped), 1))
    # Subtract the center array from the arr array
    arr_adjusted = arr_reshaped - center_repeated
    # Reshape arr_adjusted back to shape (n*3,)
    arr_adjusted = arr_adjusted.reshape(-1)
    return(arr_adjusted)
```

```
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z] for res in results.pose_landmarks.
    landmark]).flatten() if results.pose_landmarks else np.zeros(33*3)
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.
    landmark]).flatten() if results.left_hand_landmarks else np.zeros(21*3)
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.
    landmark]).flatten() if results.right_hand_landmarks else np.zeros(21*3)
    nose=pose[:3]
    lh_wrist=lh[:3]
    rh_wrist=rh[:3]
    pose_adjusted = adjust_landmarks(pose,nose)
    lh_adjusted = adjust_landmarks(lh,lh_wrist)
    rh_adjusted = adjust_landmarks(rh,rh_wrist)
    return pose_adjusted, lh_adjusted, rh_adjusted
```

As a result, we get in return 3 arrays: *pose-adjusted, lh-adjusted, rh-adjusted* which present respectively the adjusted keypoints for the pose, the left hand and the right hand in the input frame. Figure 2.4 displays the detected features by Mediapipe.

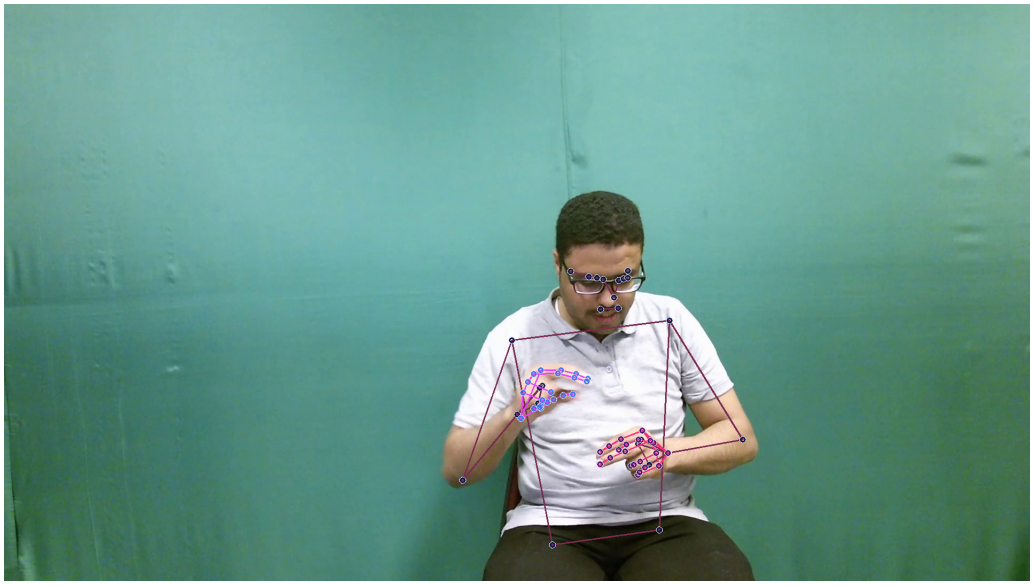

Figure 2.4: Features detection by mediapipe

2.4.2 Extracting the target features

To extract the target features,we create a list of the sign labels and then we generate a label map (c.f Figure 2.5).

{'هيكل عظمي': 0, 'جمجمة': 1, 'عمود فقري': 2, 'قفص صدري': 3, 'جهاز تنفسي': 4, 'قصبة هوائية': 5, 'رئتان': 6, 'شهيق - زفير': 7, 'جهاز هضمي': 8, 'و
جه': 9, 'بلعوم': 10, 'كبد': 11, 'البنكرياس': 12, 'الأمعاء الدقيقة': 13, 'الأمعاء الغليظة': 14, 'الزائدة الدودية': 15, 'جهاز عصبي': 16, 'قلب': 17, 'حواس خم
س': 18, 'عضلة': 19, 'أنسجة': 20, 'مستشفى': 21, 'إسعافات أولية': 22, 'جرح نازف': 23, 'حروق': 24, 'مخدر/ بنج': 25, 'عملية جراحية': 26, 'شاش / ضم
مادة': 27, 'شريط لاصق / بلاستر': 28, 'صيدلية': 29, 'تحليل دم': 30, 'فحص النظر': 31, 'فحص سريري': 32, 'ميزان حرارة': 33, 'سماعة أذن': 34, 'جهاز قيا
س الضغط': 35, 'نبض القلب': 36, 'تحليل طبي': 37, 'معمل التحاليل / مختبر': 38, 'صورة اشعة': 39, 'التهاب': 40, 'تورم': 41, 'زكام': 42, 'عدوى': 43,
'صداع': 44, 'ألم': 45, 'حمى': 46, 'إسهال': 47, 'إمساك': 48, 'مغص': 49, 'مرض السكر / سكري': 50, 'أزمة قلبية': 51, 'سرطان': 52, 'مرض فقدان المنا
عة / الإيدز': 53, 'تساقط الشعر': 54, 'سكتة قلبية': 55, 'شلل نصفي': 56, 'شلل دماغي': 57, 'ضغط الدم': 58, 'حساسية': 59, 'حكة / هرش': 60, 'دواء': 61,
'دورة شهرية': 62, 'مريض / مرض': 63, 'كبسولة': 64, 'دواء شراب': 65, 'مرهم': 66, 'قطارة': 67, 'أخذ إبرة': 68, 'تلقيح': 69, 'تطعيم': 70, 'أشعة ليز
ر': 71, 'مخدرات': 72, 'إدمان': 73, 'توحد / أوتيزم': 74, 'منغولي': 75, 'بكتريا': 76, 'جرثومة': 77, 'فيروس': 78, 'إنتشار': 79, 'إعاقة': 80, 'إعاقة ذهنية
ة': 81, 'اعاقة جسدية': 82, 'اعاقة بصرية': 83, 'إعاقة سمعية': 84, 'وباء': 85, 'مناعة': 86, 'عصب': 87, 'معافى': 88, 'يأكل': 89, 'يشرب': 90, 'ينام':
91, 'يستيقظ': 92, 'يسمع': 93, 'يسكت': 94, 'يشم': 95, 'يصعد': 96, 'ينزل': 97, 'يفتح': 98, 'يغلق (يغلق)': 99}

Figure 2.5: Labels of the classes(Target features)

2.4.3 Preprocessing the dataset

First, we concatenate the already extracted three arrays together to form the global features array. Following that, we obtained the corresponding labels for each video as our target features. Finally, we applied one-hot encoding to these labels.

2.5 Conclusion

In this chapter, we introduced the chosen dataset for our project. We explored the videos we have and the respective number of frames in order to set the threshold for frame sampling. Afterwards, we extracted our features and labels in order to prepare our data for training.

Chapter 3

Model Building and Evaluation

Plan

3.1 Introduction

In this chapter, we present the technological choices adopted in our context of study. Next, we present the metrics and optimization technique used and the experimental study performed on the KArSL dataset.

3.2 Implementation

To build our Arabic sign language model, we used the BiLSTM which is based on LSTM.
In fact, LSTM is a type of recurrent neural network (RNN) architecture to model and analyze sequential data. By using three gates(Input, forget and output), LSTM networks can effectively handle long sequences by selectively remembering or forgetting information at each time step. This enables them to capture dependencies over longer time lags and make accurate predictions.[12]
BiLSTM is a bidirectional LSTM, a model of sequence processing that consists of two LSTMs: one takes input in the forward direction and the other in the reverse direction. Bidirectional LSTMs effectively increase the amount of information available to the network, by improving the context available to the algorithm .
It is crucial to consider the context before and after a particular frame to understand the complete meaning of a sign. By processing the input sequence in both directions:

— The BiLSTM model can capture long-range dependencies more effectively.

— It can extract rich and context-aware features from the input videos.

— It can capture temporal patterns, sequential dependencies, and variations in hand movements over time. These features are essential for accurately distinguishing between different signs in sign language videos.

— The enhanced context modeling and robust feature extraction capabilities of the BiLSTM model generally lead to better classification performance compared to a simple LSTM. The bidirectional nature of the BiLSTM allows it to leverage more information from the input sequence, resulting in improved accuracy in sign language classification tasks.

Figure 3.1 presents a schema of BiLSTM model.

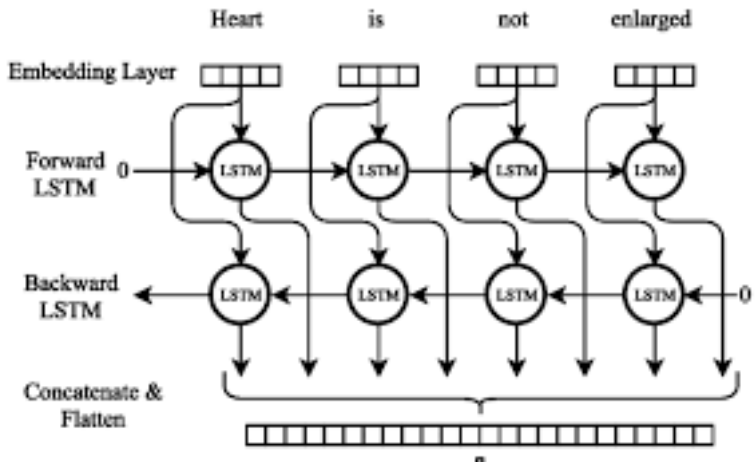

Figure 3.1: BiLSTM model

In our context of study, we notice that many signs in our dataset could have the same start or could end the same way, hence, we choose the BiLSTM model. Such model incorporates the information from past and future states in order to make predictions and improve context. For instance, we can see in **Figure 3.2** that the two signs of colic and heart display a slight difference in the hand's position.

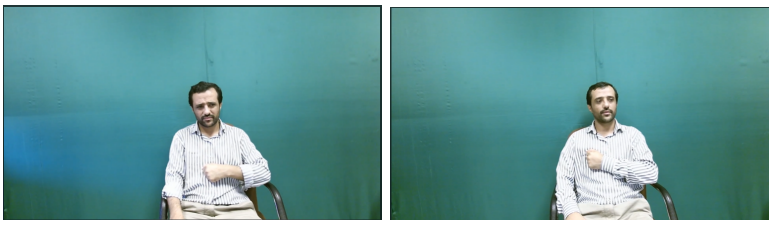

Figure 3.2: colic sign vs heart sign

Figure 3.3 presents the BiLSTM model architecture.

```
# Define the Bidirectional LSTM model with Attention model

 model= tf.keras.Sequential([
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(len(words), activation='softmax')
 ])
# Compile the model
 model.compile(optimizer='adam', loss='categorical_crossentropy',
 metrics=['categorical_accuracy'])
```

```
# Train the model on the holistic keypoints features and labels:
model_training_history = model.fit(X_train, y_train, batch_size=32,
validation_data=(X_val,y_val), validation_batch_size=32, epochs=50,
callbacks=[early_stopping])
```

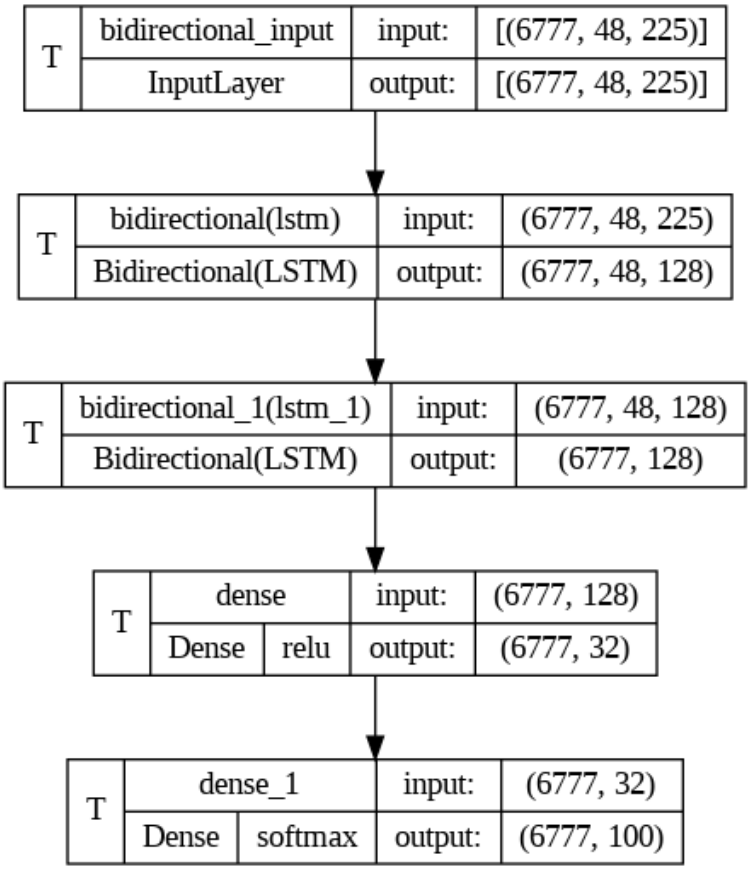

Figure 3.3: BiLSTM model architecture

3.3 Evaluation Metrics

To measure the performance of our solution, we used three evaluation metrics : confusion matrix, categorical accuracy and categorical-CrossEntropy Loss.

3.3.1 Confusion matrix

The confusion matrix, also known as the error matrix, is depicted by a matrix describing the performance of a classification model on a set of test data.[15] (cf. figure 4.1)

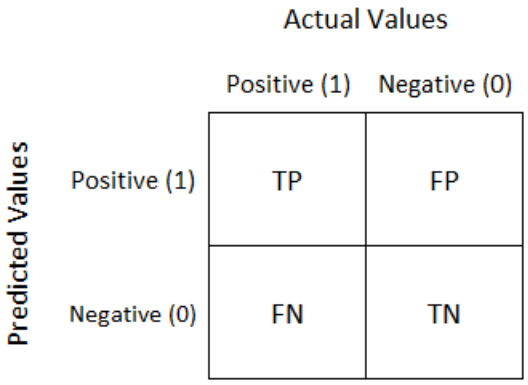

Figure 3.4: Confusion Matrix

Confusion matrices represent counts from predicted and actual values. The output **"TN"** stands for **True Negative** which shows the number of negative examples classified accurately. Similarly, **"TP"** stands for **True Positive** which indicates the number of positive examples classified accurately. The term **"FP"** shows **False Positive** value, i.e., the number of actual negative examples classified as positive; and **"FN"** means a False Negative value which is the number of actual positive examples classified as negative.[16]

3.3.2 Categorical Accuracy

Categorical accuracy is a variation of accuracy specifically designed for multi-class classification problems. It calculates the proportion of correctly classified samples in which the predicted class matches the true class. Categorical accuracy is computed as (3.1):

$$CategoricalAccuracy = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}} \tag{3.1}$$

The main difference between accuracy and categorical accuracy arises when dealing with

multi-class classification. In the case of multi-class problems, accuracy may still calculate the overall correctness of predictions, but it doesn't consider whether the predicted class matches the true class for each individual sample. Categorical accuracy, on the other hand, takes into account the alignment between predicted and true classes for all samples.

3.3.3 Categorical-CrossEntropy Loss

Loss is defined as the difference between the predicted value by your model and the true value. [18] We used the **Categorical-CrossEntropy** loss to minimize the loss. (the smaller the loss, the better the model.)

3.4 Optimization technique

To optimize our model, we used the early stopping technique and the Adam optimization algorithm.

3.4.1 Early Stopping Technique

The early stopping technique is used to reduce overfitting without compromising on the model's accuracy. It consists of stopping training the model when the validation loss begins to increase while the training loss continues to decrease as in figure 3.5 . [13]

```
 # Set up early stopping
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',  # Metric to monitor for early stopping
    mode='min',  # Set mode to 'min' for minimizing the metric
    patience=5,  # Number of epochs with no improvement before stopping
    restore_best_weights=True,  # Restore the best model weights
    verbose=1)
```

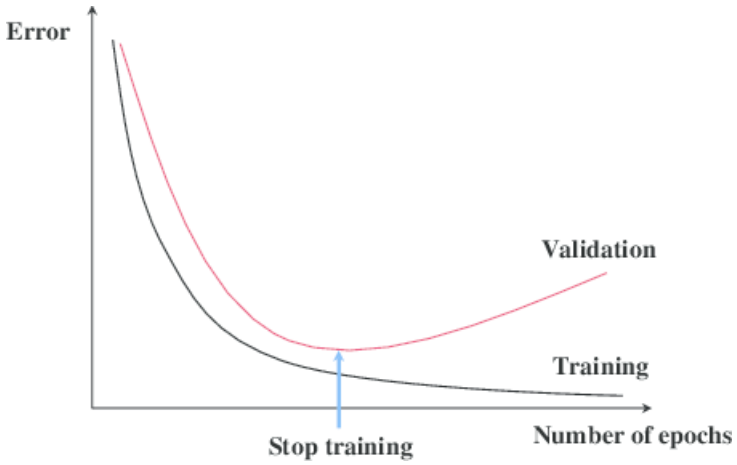

Figure 3.5: Early Stopping method

3.4.2 Adam Optimization Algorithm

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.[14]
It offers adaptive learning rates, which dynamically adjust the learning rate for each parameter based on past gradients. This adaptivity provides faster convergence and improves performance across various data and model architectures.

3.5 Experimental results

To evaluate our solution for Arabic sign language recognition, we performed two series of experiments.

3.5.1 First series of experiments

We decided to train our model on videos of signers 1 and 2 and test afterward on videos of the 3rd signer. Our data will be split into train and validation subsets, giving 20% of the whole dataset to the validation part.
In these experiments, we check the effect of data variability while training on the recognition performance. Therefore, we study two scenarii :

— **Scenario 1 :** The number of signers in the train are fixed to 1 and in the test to 2.

— **Scenario 2 :** The number of signers in the train are fixed to 2 and in the test to 1.

Table 3.1 reported the obtained results.

Scenario	Accuracy
Scenario 1	35%
Scenario 2	64%

Table 3.1: Experimental evaluation of the two scenarios

The obtained results are due to the fact that different persons perform the same sign with slightly different velocities and hand movements which affects the model greatly.

3.5.2 Second series of experiments

In this set of experiments, we trained our model on the three signers.

The achieved accuracy for the test set is **99.62%**. These high accuracy values demonstrate that the model has successfully learned the patterns in the data and succeeded to identify the different Arabic sign language.

Next, we implemented the early stopping technique using the validation set approach, employed the Adam optimizer, and used the categorical cross entropy loss function to optimize the model's learning process and enhance its generalization ability. (cf. Figures 3.6, 3.7)

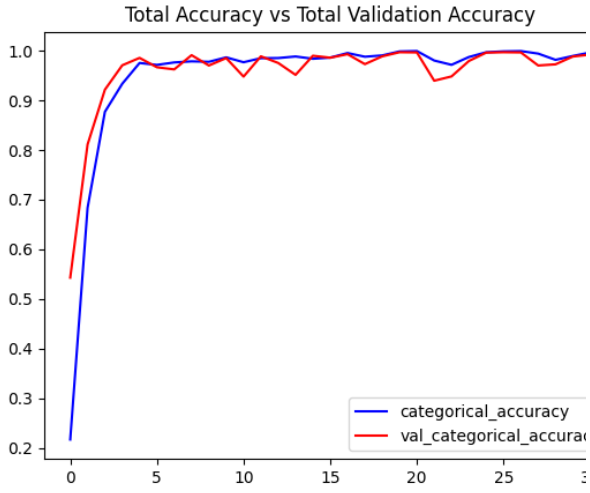

Figure 3.6: Total Accuracy vs Total Validation accuracy

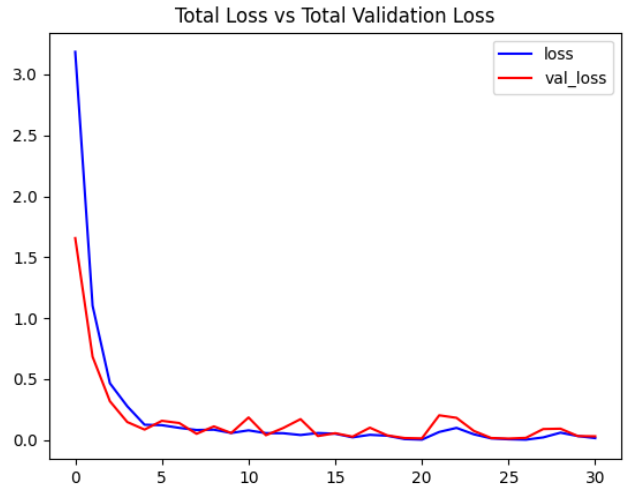

Figure 3.7: Total Loss vs Total Validation Loss

Figure 3.6 shows that our model consistently performs well in classifying examples in both the test and training sets. This indicates strong learning and performance on both seen and unseen data. The close proximity of the accuracy values between the sets suggests that the model generalizes effectively without overfitting.

Similarly, figure 3.7 demonstrate the model's optimization progress during training. The decreasing loss values indicate that the model's predictions closely match the true labels. This reflects improved predictive capabilities and parameter adjustments throughout the training process.

Additionally, we explored the confusion matrix for each class to gain more insights about the classes of misclassified instances. (cf. Figure 3.8)

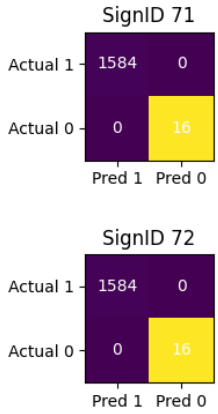

Figure 3.8: Confusion Matrix (test subset)

Furthermore, to test the model's generalization ability, we recorded two videos for the hospital and skeleton labels with two new signers. As shown in Figure 3.11, we notice that our Arabic sign language model provides promising results. Nevertheless, it still missclassified other videos which presents high inter-class similarity.

Figure 3.9: Sign of "Skeleton"

Figure 3.10: Sign of "Heart"

Figure 3.11 provides some qualitative results of our solution on our own recorded videos.As shown, our model is generic and provides promising results.

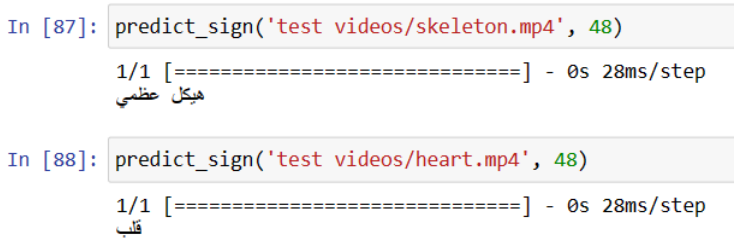

Figure 3.11: Correct Prediction of the words

3.6 Conclusion

In this chapter, we explained the techniques and methodologies used in building the model. Next, we discussed the different metrics and optimization techniques used to ensure that our model achieves high accuracy and low loss. This indicates the model's effective performance in recognizing and classifying sign language gestures.

General Conclusion and Perspectives

This report is the result of the work done by our team in the end of the year project for the acadamic year 2022/2023. Our project aimed to develop a solution for *Arabic sign language recognition* for people with special needs to communicate with others in daily life situations.

To achieve our goal, we proceeded with a CRISP-DM methodology.
As a first step, we gained insights into our dataset then we preprocessed the videos in order to extract the features needed for training. Afterwards,we fed the data to the BiLSTM model. As a last phase, we performed a deep experimental study to evaluate the performance of our model.

Nevertheless, our final model generalized better than the previous ones but there is ground for improvement. We should add more modalities like optical flow and image features that could enhance our model's performance on new unseen signers. We should also consider exploiting the entire dataset with all the 502 classes to broaden the model's vocabulary.

Bibliography

[1] Y. Tounsi. (), [Online]. Available: https://www.researchgate.net/figure/CRISP-DM-data-mining-framework_fig1_341627969.

[2] G. L. Team. « Why using crispm-dm ». (), [Online]. Available: https://www.mygreatlearning.com/blog/why-using-crisp-dm-will-make-you-a-better-data-scientist/.

[3] « Sign language ». (), [Online]. Available: https://en.wikipedia.org/wiki/Sign_language.

[4] « Sign language alphabets from around the world ». (), [Online]. Available: https://www.duplichecker.com/.

[5] C. Nyaga and R. Wario. « A review of sign language hand gesture recognition algorithms ». ().

[6] T. Ahram. « Advances in artificial intelligence, software and systems engineering ». ().

[7] (), [Online]. Available: https://www.geeksforgeeks.org/thresholding-based-image-segmentation/?fbclid=IwAR2kPNtWz0-mQpZJjK7FwanC18i-R1UspankDn-iEi9PTW4U2BXsCrSFv2A.

[8] (), [Online]. Available: https://www.tutorialspoint.com/region-and-edge-based-segmentation?fbclid=IwAR0_gHSvp_6pmRPDnGknvurr_4xKWvTzPFyizNByOYmKUARqu95DREXNmrM.

[9] I. M. A. S. Agoes. « Applying hand gesture recognition for user guide application using mediapipe ». (), [Online]. Available: https://www.researchgate.net/publication/357216549_Applying_Hand_Gesture_Recognition_for_User_Guide_Application_Using_MediaPipe#pf3.

[10] « Yolov7 pose vs mediapipe in human pose estimation ». (), [Online]. Available: https://learnopencv.com/yolov7-pose-vs-mediapipe-in-human-pose-estimation/?fbclid=IwAR04BVs2VRYMKqnGYdVLtxj7KXet_vtPllx8OSqvDQrz6lDkSmjMv1UKJR8#:~:text=MediaPipe\%20tracks\%20the\%20person\%20once,performs\%20detection\%20on\%20each\%20frame.

[11] S. M. M. M. Ala Addin I.Sidig Hamzah Luqman. « Karsl-data ». (seen the 20/03/2023), [Online]. Available: https://dl.acm.org/doi/10.1145/3423420.

[12] (), [Online]. Available: https://paperswithcode.com/method/bilstm#:~:text=A\%20Bidirectional\%20LSTM\%2C\%20or\%20biLSTM,other\%20in\%20a\%20backwards\%20direction..

[13] « What is early stopping? » (), [Online]. Available: https://www.educative.io/answers/what-is-early-stopping.

[14] J. Brownlee. « Gentle introduction to the adam optimization algorithm for deep learning ». (July 3, 2017), [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[15] G. K. S. V. Deepak Kumar Sharma Mayukh Chatterjee. « Deep learning applications for disease diagnosis ». (), [Online]. Available: https://www.sciencedirect.com/topics/engineering/confusion-matrix.

[16] F. A. B. Ajay Kulkarni Deri Chong. « Foundations of data imbalance and solutions for a data democracy ». (), [Online]. Available: https://www.sciencedirect.com/topics/engineering/confusion-matrix.

[17] « Accuracy and loss ». (), [Online]. Available: https://machine-learning.paperspace.com/wiki/accuracy-and-loss.

[18] « Loss vs accuracy ». (Friday, December 7, 2018), [Online]. Available: https://kharshit.github.io/blog/2018/12/07/loss-vs-accuracy.