



GIVE A SIGN

Omnia nassef abdel samad
rawan ragab abdel hady
Hisham Rifaay Ali

Mohamed Amin Ahmed
Waeir Abdullah Waeir
Zahraa Mohamed Ali

Supervised by:
Dr. Esraa Al-Hariri
Dr. Asmaa Sweidan

Acknowledgement

We would like to express my deepest gratitude to [supervisors/Dr. Esraa Al-Hariri and Dr. Asmaa Sweidan] for their invaluable guidance, unwavering support, and insightful feedback throughout the duration of this project. Their expertise and encouragement have been instrumental in shaping the direction and quality of this work.

Lastly, we are indebted to [faculty of computers and artificial intelligence] for their financial support, without which this project would not have been possible.

Abstract

This project presents the development and implementation of a Sign Language Recognition System designed to bridge the communication gap between the deaf and hearing communities. Utilizing advanced machine learning techniques, the system accurately interprets sign language gestures into spoken language and text. The core of the system is built on YOLO V8 that has been trained on a comprehensive dataset of sign language gestures, encompassing a wide range of signs from American Sign Language (ASL) and Arabic language.

Through the integration of high-definition cameras processing software, the system captures the nuances of sign language gestures, including hand shapes, orientations, and movements. The recognition process involves pre-processing of video or photo input to isolate hand gestures, feature extraction to identify distinguishing characteristics of each sign, and classification using the trained YOLO model. The system demonstrates high accuracy in recognizing both static and dynamic signs, offering an intuitive interface for users to communicate effectively.

The project also explores the challenges of sign language recognition, such as variations in signing style, background noise in video input, and the complexity of continuous sign language gestures. Solutions to these challenges, including data augmentation, background subtraction techniques, and sequence modeling for understanding continuous signing, are discussed.

Contents

| | |
|--|----|
| Acknowledgement..... | 1 |
| Abstract..... | 2 |
| Chapter 1. introduction..... | 7 |
| 1.1 Overview | 7 |
| 1.2 Problem Definition | 7 |
| 1.3 PROBLEM MOTIVATION..... | 8 |
| 1.4 Project Objectives..... | 8 |
| Chapter 2. project planning | 10 |
| 2.1 introduction | 10 |
| 2.2 project plan | 10 |
| 2.2.1 gathering requirements | 10 |
| 2.2.2 system analysis | 10 |
| 2.2.3 system design..... | 10 |
| 2.2.4 implementation | 10 |
| 2.2.5 project schedule Gantt chart | 11 |
| 2.3 feasibility study | 12 |
| 2.3.1 focus | 12 |
| 2.3.2 strategies | 12 |
| 2.3.3 action steps..... | 13 |
| 2.3.4 KPIs..... | 13 |
| 2.3.5 MOS | 13 |
| Chapter 3. Literature review..... | 14 |
| 3.1 Overview..... | 14 |

| | |
|--|----|
| 3.2 American Sign Language Detection using YOLOv5 and YOLOv8 | 14 |
| 3.2.1 related work | 15 |
| 3.2.2 Methodology | 18 |
| 3.2.3 Results | 21 |
| 3.2.4 Conclusion | 22 |
| 3.2.5 References | 22 |
| 3.3 Video Captioning Based on Sign Language Using YOLOV8 | |
| Model | 22 |
| 3.3.1 Proposed Methodology | 22 |
| 3.3.2 Results | 25 |
| 3.3.3 Conclusion | 26 |
| 3.3.4 References | 27 |
| Chapter 4. preliminaries | 28 |
| 4.1 Introduction | 28 |
| 4.2 sign language background | 28 |
| 4.2.1 Historical of sign-language | 29 |
| 4.2.2 There is no single sign language | 29 |
| 4.3 An intelligent diagnosis system | 32 |
| 4.4 Data preprocessing | 32 |
| 4.4.1 image scaling | 32 |
| 4.4.2 image noise reduction | 33 |
| 4.4.3 image segmentation | 33 |
| 4.4.4 skin color detection | 33 |
| 4.4.5 bounding boxes | 33 |

| | |
|---|----|
| 2.5 feature engineering | 34 |
| 4.5.1 feature extraction | 35 |
| 4.5.2 feature selection | 35 |
| 4.5.3 Feature fusion | 37 |
| 4.6 deep learning..... | 38 |
| Chapter 5. Proposed System..... | 39 |
| 5.1 Overview | 39 |
| 5.2 Proposed framework | 39 |
| 5.3 System analysis | 40 |
| 5.3.1 Process modeling | 40 |
| 5.3.2 Data flow diagram | 41 |
| 5.3.3 Requirements | 41 |
| 5.3.4 Use case diagrams..... | 43 |
| 5.3.5 Use case scenario | 44 |
| 5.3.6 Activity diagram | 46 |
| 5.3.7 Non functional requirements | 47 |
| 5.4 system design | 47 |
| 5.4.1 Sequence diagram..... | 47 |
| Chapter 6. Experimental Results and Comparative Analysis for YOLO v8 | 49 |
| 6.1 Results of The System | 49 |
| 6.1.1 Dataset description..... | 49 |
| 6.2 Comparative analysis..... | 51 |
| 6.2.1 YOLO versions | 53 |
| 6.3 Experiment Specifications and Used Materials | 65 |

| | |
|---|----|
| Chapter 7 System Development | 68 |
| 7.1 Overview | 68 |
| 7.2 App's architecture & design patterns | 69 |
| 7.3 Methodological assumptions | 69 |
| 7.3.1 User Requirements | 69 |
| 7.3.2 System Requirements | 69 |
| 7.4 Used Technologies in mobile application | 69 |
| 7.4.1 Dart | 69 |
| 7.4.2 Flutter | 70 |
| 7.5 Technologies used in Back-end | 70 |
| 7.5.1 Flask | 70 |
| 7.5.2 ASP.NET | 70 |
| 7.5 Mobile development | 71 |
| | 75 |
| Chapter 8 Conclusions & Future Work | 76 |
| 8.1 Conclusions | 76 |
| 8.2 Future Work | 76 |
| Chapter 8 References | 78 |
| 8.1 References for American Sign Language Detection using YOLOv5 and YOLOv8 | 78 |
| 8.2 References for Video Captioning Based on Sign Language Using YOLOV8 Model | 79 |

Chapter 1. introduction

Introduction This chapter reviews the basic definitions of the project, presenting the problem we face and the solution we developed for it, the reasons and motives we believed in to complete this idea, and the obstacles and obstacles we faced in completing the project.

1.1 Overview

Our project is a mobile app for deaf and mute patients, where patients download the application, log in to it, and they could communicate with others easily. Where the app converts sign language into English in writing. Hence the main purpose of the app, which is facilitate the process of communication between a large segment of society, namely the deaf, with the rest of society.

1.2 Problem Definition

Lately, the number of patients with Deaf has increased and speeded among all different age generations. This kind of people need special treatment for dis ability to speak or listen. Genetic cells or genes are cells that are passed from parents to children. Genes play an important role in hearing loss in children as they account for about half of all cases of hearing loss in children. Based on some analytics, we have a vision of: About 8 million Egyptians they suffering from Deaf.

PROPOSED SOLUTION What this app provides now stands as an instant assistant for the user. Where the app acts as a speaker on behalf of the deaf person.

1.3 PROBLEM MOTIVATION

The vision we believe in is to impact other people's lives positively. We see our ability, along with the opportunity provided by this application, to preserve someone's safety and prevent their suffering or health deterioration. With the increasing number of internet and application users in their 60s, we aim to reduce danger and enhance their safety. Additionally, COVID-19 has led to a positive shift in both technological and medical awareness levels.

1.4 Project Objectives

The objective of a sign language application can be presented as follows:

***Accessibility*:** To make communication accessible for the Deaf and Hard of Hearing community, enabling them to engage more easily with the hearing world and vice versa.

2. ***Education*:** To provide a resource for learning and practicing sign language, supporting both new learners and those seeking to improve their proficiency.

***Inclusion*:** To promote social inclusion by bridging the communication gap between Deaf and hearing individuals, fostering understanding and empathy.

***Communication Efficiency*:** To offer real-time translation and interpretation services, making everyday interactions more seamless for Deaf individuals.

5. ***Awareness and Advocacy*:** To raise awareness about the Deaf culture, sign languages, and the importance of accessibility, advocating for equal opportunities.

6. ***Technology Integration*:** To leverage technology (e.g., AI, AR, machine learning) in innovative ways to enhance sign language

recognition and translation, improving the application's effectiveness and user experience.

7. ***Community Building***: To create a platform for Deaf individuals and sign language learners to connect, share experiences, and support each other, fostering a sense of community.

Resource Availability: To provide a comprehensive and easily accessible library of sign language resources, including vocabulary, grammar, and cultural nuances.

Chapter 2. project planning

2.1 introduction

Project management is a process of planning and controlling the development of a system within a specified time frame with the right functionality, and de-fine Project Focuses, Strategies, and Action Steps, and measuring progress and performance.

2.2 project plan

The planning Phase is the fundamental process of understanding why a system should be built and also determining how the project team will go about building the system.

2.2.1 gathering requirements

The analysis of this information leads to the development of a concept for a new system.

2.2.2 system analysis

- functional and non-functional requirements
- (Data flow diagram) DFD
- Use Case

2.2.3 system design

- Sequence diagram
- Class diagram
- Activity diagram
- context diagram

2.2.4 implementation

- YOLOv8 model
- Back-end
- Mobile Application

2.2.5 project schedule Gantt chart

A Gantt chart is a horizontal bar chart used in project management to visually represent a project plan over time. Gantt charts typically show the timeline and status.

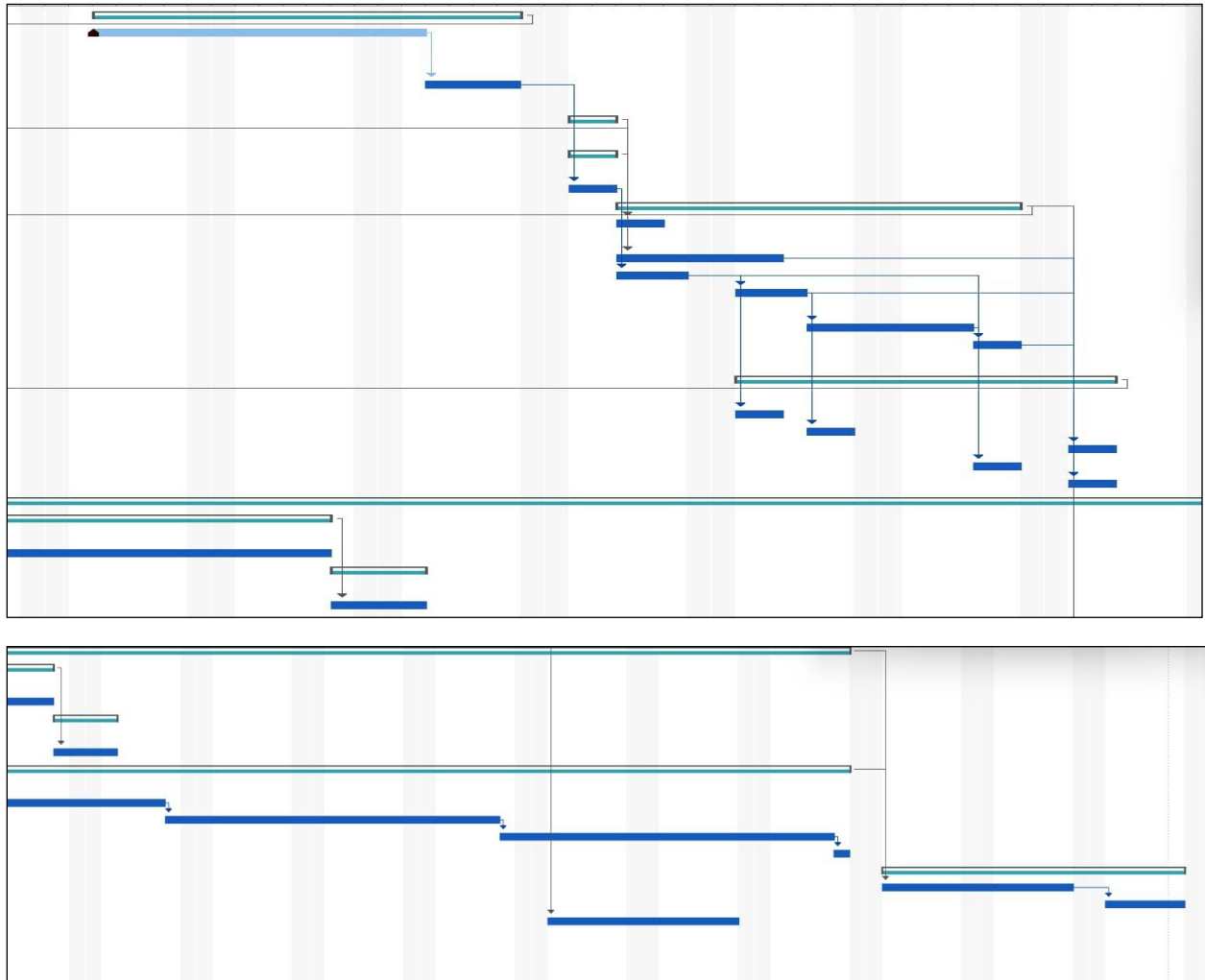


Figure 1 Gantt chart

| | Task Name | Resource Names | Duration | Start | Finish | Work | Predecessors |
|----|--|----------------|-----------------|----------------------------|----------------------------|--------------|--------------|
| 1 | Data collection | | 15 days | Tue 2/14/23 8:00 AM | Fri 3/3/23 5:00 PM | 0 hrs | |
| 2 | Collect appropriate photos and videos | | 10 days | Tue 2/14/23 8:00 AM | Mon 2/27/23 5:00 PM | 0 hrs | |
| 3 | Test the validity of the information | | 4 days | Tue 2/28/23 8:00 AM | Fri 3/3/23 5:00 PM | 0 hrs | 2 |
| 4 | Download the programs used | | 7 days | Mon 3/6/23 8:00 AM | Tue 3/7/23 5:00 PM | 0 hrs | |
| 5 | Finding the best tools | | 5 days | Mon 3/6/23 8:00 AM | Tue 3/7/23 5:00 PM | 0 hrs | |
| 6 | installation | | 2 days | Mon 3/6/23 8:00 AM | Tue 3/7/23 5:00 PM | 0 hrs | 3 |
| 7 | system analysis | | 20 days | Wed 3/8/23 8:00 AM | Fri 3/24/23 5:00 PM | 0 hrs | |
| 8 | work flow diagram | | 2 days | Wed 3/8/23 8:00 AM | Thu 3/9/23 5:00 PM | 0 hrs | 4 |
| 9 | DFD diagram | | 5 days | Wed 3/8/23 8:00 AM | Tue 3/14/23 5:00 PM | 0 hrs | 5 |
| 10 | use case diagram | | 3 days | Wed 3/8/23 8:00 AM | Fri 3/10/23 5:00 PM | 0 hrs | 6 |
| 11 | user sequanses diagram | | 3 days | Mon 3/13/23 8:00 AM | Wed 3/15/23 5:00 PM | 0 hrs | 10 |
| 12 | activety diagram | | 5 days | Thu 3/16/23 8:00 AM | Wed 3/22/23 5:00 PM | 0 hrs | 11 |
| 13 | admin sequanses diagram | | 2 days | Thu 3/23/23 8:00 AM | Fri 3/24/23 5:00 PM | 0 hrs | 12 |
| 14 | design UI Conceptually | | 10 days | Mon 3/13/23 8:00 AM | Tue 3/28/23 5:00 PM | 0 hrs | |
| 15 | home page | | 2 days | Mon 3/13/23 8:00 AM | Tue 3/14/23 5:00 PM | 0 hrs | 10 |
| 16 | user page | | 2 days | Thu 3/16/23 8:00 AM | Fri 3/17/23 5:00 PM | 0 hrs | 11 |
| 17 | admin page | | 2 days | Mon 3/27/23 8:00 AM | Tue 3/28/23 5:00 PM | 0 hrs | 13 |
| 18 | not found page | | 2 days | Thu 3/23/23 8:00 AM | Fri 3/24/23 5:00 PM | 0 hrs | 10,12 |
| 19 | setting page | | 2 days | Mon 3/27/23 8:00 AM | Tue 3/28/23 5:00 PM | 0 hrs | 9,11,13 |
| 20 | implementation | | 40 days | Fri 2/10/23 8:00 AM | Fri 4/14/23 5:00 PM | 0 hrs | |
| 21 | UI Flutter screens implementation | | 20 days? | Fri 2/10/23 8:00 AM | Thu 2/23/23 5:00 PM | 0 hrs | 14 |
| 22 | UI test | | 10 days | Fri 2/10/23 8:00 AM | Thu 2/23/23 5:00 PM | 0 hrs | |
| 23 | Database implementation | | 5 days | Fri 2/24/23 8:00 AM | Mon 2/27/23 5:00 PM | 0 hrs | |
| 24 | Database test | | 2 days | Fri 2/24/23 8:00 AM | Mon 2/27/23 5:00 PM | 0 hrs | 21 |

Figure 2project schedule

2.3 feasibility study

The feasibility study outlines and analyzes several alternatives or methods of achieving business success.

2.3.1 focus

Helping a deaf and dumb patient.

2.3.2 strategies

- Build system that help a deaf and dumb patient through Mobile Application.
- Create a Business Model

- lunching Application

2.3.3 action steps

- team members are familiar with technology to create an app easy to use.
- team members will get educated more about a deaf and dumb disease

To achieve them and set a time frame for achieving them.

2.3.4 KPIs

- API works properly with Mobile APP.
- Model of yolov8 detect things Correctly.

2.3.5 MOS

Application has been launched and a lot of a deaf and dumb patient use it

Chapter 3. Literature review

3.1 Overview

This chapter includes a brief history of some international Applications and their establishment, the technology involved in improving and facilitating the advising and enrollment processes, what problems happen and what they reach and their limitations, and references. As discussed in the previous chapter, this project presents:

1. American Sign Language Detection using YOLOv5 and YOLOv8
2. Video Captioning Based on Sign Language Using YOLOv8 Model

3.2 American Sign Language Detection using YOLOv5 and YOLOv8

over 1.5 billion people suffer from hearing loss around the globe. Additionally, about one billion teenagers are at risk of developing hearing loss due to the improper use of earbuds and headphones. Older adults often experience hearing loss, social isolation, loneliness, and frustration. Children with hearing loss may face delayed language development and communication difficulties. Sign language detection poses several challenges due to its unique characteristics and the complexity of capturing and interpreting sign language. each country has its own unique way of using sign language. Some of the key challenges in sign language detection include:

Variations in sign languages: SL vary across different regions and countries. Each sign language has its own vocabulary, grammar, and syntax. Detecting and understanding different sign languages require language-specific models and datasets, making developing a universal sign language detection system challenging.

Gesture recognition: Sign languages involve a combination of simultaneous hand movements, facial expressions, body postures, and other non-manual markers. Capturing and recognizing these subtle and dynamic gestures accurately is a complex task.

Data scarcity: SL data is relatively scarce compared to spoken languages. Building accurate sign language detection models requires large amounts of annotated data, which is often limited, especially for certain sign languages or specific sign variations.

Background noise and occlusion: Sign language is often performed in real-world environments, which can introduce background noise, occlusions, and cluttered backgrounds. These factors can interfere with the visibility of the signer's hands and facial expressions, making it difficult to detect and interpret signs accurately.

Ambiguity and context dependency: Sign languages rely on facial expressions, context, and body movements to express meaning. Isolated signs may have multiple interpretations depending on the surrounding signs or the speaker's intentions.

Another major issue faced by the deaf community is that some languages are officially recognized while others are not recognized by the government or the institutions.

3.2.1 related work

In the first method, an external device is used for sign recognition; while another method for detecting and recognizing hand gestures is to use deep learning. Several researchers have developed novel techniques for SL recognition with solutions to different aspects such as cost, latency, performance and portability. A. Bhattacharya talks about training a classifier on a dataset of 24 gestures that can be easily spelt on fingers using the "bag of visual words approach". The original system had a desk-mounted camera to watch the user and had a 92% accuracy rate.

The second device, which reaches 98% accuracy, mounts the camera in the user's hat. L. Aziz surveyed the most recent developments in visual object detection with deep learning, covering around 300 methods, including region-based object detection methods such as SPPnet, Faster R-CNN, Classification and regression-based object detection methods such as YOLO, etc. The authors also researched and analyzed publicly available benchmark datasets based on their origin, usage, advantages and limitations along with their evaluation metrics. D. Naglot recognized several signs using a Multi-layer Perceptron neural network on 520 samples contained in a dataset. Table 1. shows the recent state-of-the-art methods for sign language detection.

Table 1

| Related work | | | | |
|----------------------------|---|---|--|---|
| AUTHOR | METHOD | ALGORITHM USED | DATASET | ACCURACY |
| O. Vedak et al. [2] | SL interpreter utilizing machine learning and image processing | HOG & SVM | 6000 images of 26 English language alphabets. | 88% |
| A. Bhattacharya et al. [5] | Training a SL classifier using the bag of visual words approach | SURF and BRISK features are used for automatic feature identification. Algorithms such as KNN, SVM, etc, are used for evaluation. | 4972 images of 24 static single-handed fingerspelling gestures on a plain background | SURF Features SVM – 91.35 KNN – 86.97 BRISK Features SVM – 91.15 KNN – 87.38 |
| D. Naglot et al. [8] | In Real-time SL detection | MLP and Back propagation | 26 different alphabets of | 96.15% |

| | | | | |
|--------------------------|--|--|---|-----------------------------|
| | using Leap Motion Controller | (An idea for a categorization model that accepts a set of features as input.) | ASL of 520 samples (consisting of 20 samples of each alphabet) | |
| C. Chuan et al. [9] | SL recognition using Leap Motion Sensor | K-NN & SVM (Leap motion sensors and a webcam have a lot of potential to advance SL learning techniques.) | Four data sets were collected from the two signers with two sets from each individual. | KNN – 7.78% SVM – 79.83% |
| Z. wang et al. [11] | End-to-end SL detection in real time. | DeepSLR technology" is used to record the coarse and finger movements along with several sensors. | 5586 sign words and 26 alphabet signs | - |
| M. Fernando et al. [12] | Low-cost approach for Real-time SL recognition | Active shape models | 50 signs, 5 signs from 10 users (A, B, C, D, V Signs) | 76% |
| K. Dutta et al. [13] | Machine learning methods for ISL detection | ISL | 220 images of double handed ISL alphabets and 800 images of single handed Indian SL alphabets | 92% |
| T. Mangamuri et al. [14] | Two-handed ISL dataset for | HOG | THISL dataset with 26 | 87.67% |

| | | | | |
|--|--|--|--|--|
| | comparing machine learning classification models | | motions, each of which represents a letter of the English alphabet | |
|--|--|--|--|--|

3.2.2 Methodology

The most commonly used datasets for object detection and segmentation are Pascal VOC 2007 and Microsoft COCO. This research focuses on YOLO version 5 and 8, as these models have better performance in recognizing sign language with high accuracy.

YOLOv8

Also authored by Glenn Jocher and launched on January 23rd 2023, YOLOv8 is the latest in the family of algorithms and is still in development, along with adding many new features such as Anchor free detection and mosaic augmentation. A CLI that is included with YOLOv8 makes training a model easier to understand. Moreover, there is a Python package that offers a smoother development experience than the previous model. The Github repository for the YOLOv8 is available : [GitHub - ultralytics/ultralytics](https://github.com/ultralytics/ultralytics): NEW - YOLOv8 ☐ in PyTorch > ONNX > CoreML > TFLite. Predictions for both the bounding boxes as well as the classes are produced when the input image has been evaluated once. The algorithm is as since both the predictions – bounding box and classification, are performed simultaneously. The provided image is initially converted into a grid of equal lengths (S x S). Next, confidence scores are defined for each grid cell's "b" bounding boxes as shown in equation (i). Confidence is the probability that an object exists in every bounding box.

$$Confidence (C) = P(object) * IOUpredtarget \text{ ---(i)}$$

Where, $IOU = \text{Intersection over union}$

IOU stands for a fractional value in the range of 0 and 1. Union is the total area between the predicted and the target areas, whereas intersection is the overlap between the predicted bounding box and the target area. The ideal value is close to 1, which denotes that the estimated bounding box is near the target region. Along with this, every grid cell also predicts the Confidence conditional class probability as shown in equation (ii) and (iii).

$$C = P(\text{Class}_i / \text{object}) * P(\text{object}) * I.O.U_{predtarget} \text{---(ii)}$$

$$C = P(\text{Class}_i) * I.O.U_{predtarget} \text{---(iii)}$$

Now coming to the loss function, it is calculated by summing all the bounding box parameter's loss function result as shown in equation (iv),

$$\begin{aligned} & \lambda c o o r d \sum_{i=0}^S \sum_{j=0}^b 1[(x_i - x^i)^2 + (y_i - y^i)^2] \\ & + \lambda c o o r d \sum_{i=0}^S \sum_{j=0}^b 1[(w_i - w^i)^2 + (h_i - h^i)^2] \\ & + \sum_{i=0}^S \sum_{j=0}^b 1(C_i - C^i)^2 + \lambda n o o b j \sum_{i=0}^S \sum_{j=0}^b 1 i j o b j (C_i - C^i)^2 \\ & + \sum_{i=0}^S 21 i o b j \sum_c (p_i(c) - p^{i(c)})^2 \text{---(iv)} \end{aligned}$$

The given equation carries five important terminologies defined as follows:

- (x_i, y_i) – coordinates of target center (bounding box)
- (x^i, y^i) – coordinates of predicted center (bounding box)
- (w_i, h_i) – dimensions of target
- (w^i, h^i) – dimensions of predicted

Equation's first step computes the loss associated with the bounding box using coordinates (x_i, y_i) . If an object is present inside the j^{th} forecasted bounding box inside the i^{th} cell, $i j o b j$ is defined as 1, and if it is not

true then 0 as shown in equation (v). The prediction with the highest current IOU with the target region will be considered "responsible" for predicting an object by the predicted bounding box.

$$\lambda coord \sum_{i=0}^S \sum_{j=0}^b 1[(x_i - x^i)^2 + (y_i - y^i)^2] \text{---(v)}$$

The next portion is responsible for calculating the error in the prediction of the dimensions of the bounding box. The scale of the inaccuracy in the large boxes, however, is less impactful on the equation itself than it is in the small boxes. The square roots of width and height, which are both normalized in the range of 0 and 1, make the discrepancies between smaller values bigger than those between larger ones. As a result, rather than using the dimension values directly, the bounding box's square root is employed as shown in equation (vi).

$$\lambda c o o r d \sum_{i=0}^S \sum_{j=0}^b 1[(w_i - w^i)^2 + (h_i - h^i)^2] \text{---(vi)}$$

The loss value of confidence is calculated in the next section for both circumstances, regardless of whether the object is present inside the bounding box or not. However, if that predictor is in charge of the target box, only then the loss function shall penalize the object confidence mistake as shown in equation (vii). If there is an object in the cell, $1 i j o b j$ equals 1, else results 0.

$$\sum_{i=0}^S \sum_{j=0}^b 1(C_i - C^i)^2 + \lambda n o o b j \sum_{i=0}^S \sum_{j=0}^b 1 i j o b j (C_i - C^i)^2 \text{---(vii)}$$

With the exception of $1 i j o b j$, which is needed because the algorithm does not penalize classification errors if there are no objects present in the cell, the last portion computes the loss of class probability [19] as shown in equation (viii) .

$$\sum_{i=0}^S \sum_{j=0}^b 1 i o b j \sum_c (p_i(c) - p^{i(c)})^2 \text{---(viii)}$$

The first step is to install and initialize both the algorithms –YOLOv8, the algorithms will be running and also less time consuming. The data set that is chosen for this work is “American Sign Language letters” from the publicly available datasets on Roboflow. Along with that, we will use PyTorch which is based on the well-known Torch library. Moreover, a Python-based library that is more frequently used for computer vision and natural language processing. When downloading a dataset from Roboflow, many methods are provided as to how that dataset should be implemented in the model. One of those methods is to apply the pytorch code that installs a roboflow package and also downloads the dataset directly into the directory of the program. Another major package that is downloaded is Ultralytics, this package provides all the versions of the YOLO algorithm hence making it optimal for this particular program.

3.2.3 Results

The dataset used contains images for testing, training and validation in the ratio of 1: 21: 2 respectively (72 for testing, 1512 for training, 144 for validation) and the new model has been trained for 80 epochs. While YOLOv8’s case, those occurrences are 13. In hindsight, this comparison may seem redundant but after observing the two matrices, a conclusion is reached that background prediction for both cases are highly differentiable. One difference between v5 and v8 is the number of iterations needed to reach the minimum, v8 requires less number of iterations as compared to the v5. This may have happened because YOLOv8 is still in its first few stages of development as compared to its predecessor.

| BOUNDING BOX LOSS | | CLASSIFICATION LOSS | | mAP | |
|-------------------|------------|---------------------|------------|--------|-----|
| 1st epoch | 80th epoch | 1st epoch | 80th epoch | | |
| YOLOv8 | 1.444 | 0.312 | 4.352 | 0.1735 | 96% |

3.2.4 Conclusion

While YOLOv8 performs noticeably better during training and validation to some extent, even in testing. However, the model is not ready for any practical applications that involve constant frame movement, which makes YOLOv8 a more conceptual model at its current stage. On the other hand, YOLOv5 is easily able to predict the right gesture and even fills the prediction, which was left blank by the other model.

3.2.5 References

3.3 Video Captioning Based on Sign Language Using YOLOV8 Model

3.3.1 Proposed Methodology

1-Designing a SLR Real-Time Video-Based System Whose Purpose is to Detect the ASL Designing a SLR (sign language recognition) real-time video-based system involves creating an artificial intelligence model that can accurately detect and recognize American Sign Language (ASL) signs from video input in real-time. The system should be designed to work with a standard camera or webcam and process video input from any angle or orientation. To develop such a system, we would need to start by collecting a large dataset of sign language videos, preferably from multiple signers, with different skin colors and backgrounds. The videos should capture different signs and hand movements from different angles and lighting conditions. Next, the dataset would need to be preprocessed, which includes tasks like removing background noise, segmenting individual signs, and labeling the signs correctly. This would require using computer vision techniques and human annotation. The American Sign Language Letters dataset is an object detection dataset that includes a bounding box for each ASL letter. This project has a collection of 720 photographs take with

different hand postures held at different places. As this is a tiny dataset, manual labeling with bounding box coordinates was performed using the labeling software in Roboflow, and a probability of transformations function was used to generate many instances of the same picture, each with different bounding boxes. Figure 1 Represents the sample from the dataset with identified sign.



Fig. 1. Sample image from the dataset

2-Recognize the Hand Motions from the Acquired Image Frame Yolo-v8 is one of the currently leading models in this industry. YOLO can identify hand gestures in image frames because it uses deep neural networks to locate objects in real-time. Image acquisition is followed by pre-processing to assure image quality, object recognition with YOLO, hand gesture classification using further machine learning approaches, post-processing to fine-tune results, and output of the recognised hand gestures. The quality of the training data, the model architecture, and the hyperparameters all have an impact on how accurate the hand motion detection system is. The reliability and sturdiness of the hand motion recognition system must be ensured through careful examination and testing. Also, to extract the exact sign from the frame or the input video, it removes the background and precisely captured the hand from the camera input during real time sign language detection. Even though Yolo-v8 was not produced by the original YOLO writers, YOLO v8 is

believed to be quicker and lighter, with accuracy comparable to YOLO v5, which is widely regarded as the quickest and most accurate real-time object recognition model. 85% of the enhanced photos were used for training, with the remaining 15% set aside for testing and validation. The model was trained for 64 epochs with the YOLOv8 pre-trained weights using transfer learning.

3-Translate the Picture to Textual Content Images of sign language can be converted into text using the YOLO object identification technique and the spellchecker library. The spellchecker library is used to process the labels produced by YOLO, and it can detect and rectify any potential spelling mistakes in the text. As a result, sign language gestures can be converted into written text and the message expressed by the sign language image can be represented textually. The algorithms and dictionaries used in the spellchecker library assist in recommending adjustments depending on context, ensuring that the final literary material is grammatically and linguistically accurate. It is easier for a larger audience to access and comprehend because to this combined use of YOLO and the spellchecker library.

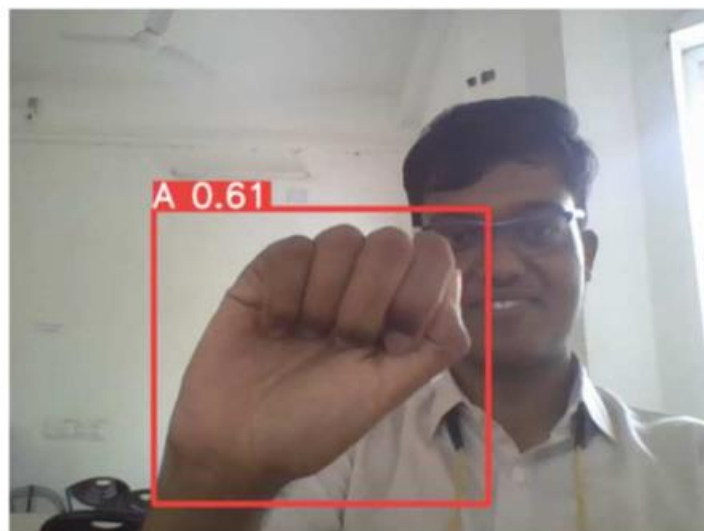


Fig. 2. Sample test

3.3.2 Results

Mean Average Precision Mean Average Precision (mAP) at 50 is a commonly used evaluation metric for object detection models, including those based on YOLO (You Only Look Once) algorithm. It measures the accuracy of the model in detecting objects across different categories (e.g., person, car, bicycle) at a certain intersection over union (IoU) threshold of 0.5. The mAP@.5 score of the created model is 95.7%. Figure 2 represents the predicted sample test results.

Confusion Matrix In object detection with YOLO (You Only Look Once), the confusion matrix can be used to evaluate the performance of the algorithm on a test set of images. The confusion matrix for object detection with YOLO is a table that summarizes the predicted and actual labels for each object in the test set. Figure 3 depicts the Confusion Matrix plotted for each character.

Precision-Recall Curve A precision-recall (PR) curve is a graphical representation of the performance of a binary classification algorithm that measures the trade-off between precision and recall at different threshold settings. Figure 4 illustrates the Precision-Recall (PR) curve for our model.

Video Captioning Based on Sign Language Using YOLOV8 Model

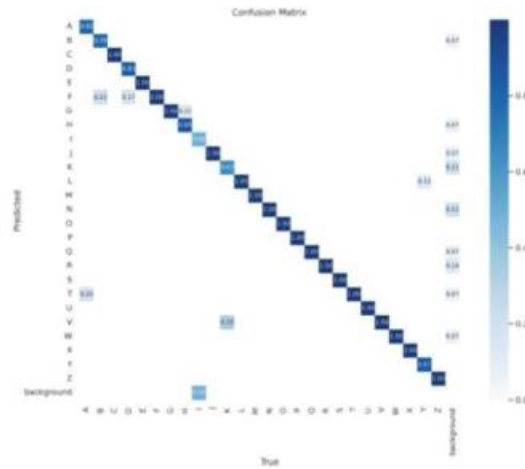


Fig. 3. Confusion matrix plotted for each character.

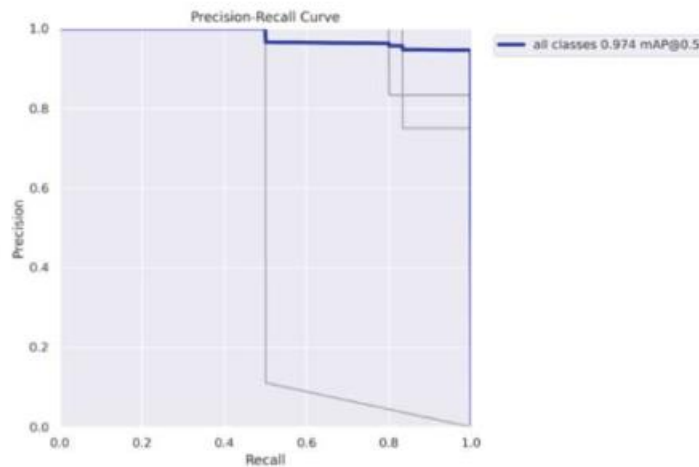


Fig. 4. PR Curve for our model.

3.3.3 Conclusion

Video captioning based on sign language is a crucial solution for improving accessibility for individuals who are deaf or hard of hearing, providing them with equal access to information, education, and multimedia content. With accurate and synchronized captions that convey sign language expressions, video content becomes more inclusive and promotes effective communication, bridging gaps between

sign language users and nonsign language users in various settings. Continued advancement and promotion of sign language video captioning can create a more inclusive world, fostering social inclusion and enhancing communication for individuals with hearing loss.

3.3.4 References

Chapter 4. preliminaries

4.1 Introduction

This chapter presents a brief idea concerning the core concepts of a smart SL caring mobile app, which enables deaf people to be able to communicate to request their needs. Then it presents a brief idea about learning.

4.2 sign language background

Sign language (SL) is the primary means of communication between people with hearing loss and other communities and is expressed through manual (body and hand movements) and non-manual (facial expressions) features. These features are combined to form phrases that convey the meaning of words or sentences. The ability to capture and understand the relationship between phrases and words is crucial for the deaf community to guide us into an era in which translation between phrases and words can be achieved automatically. The research community has long identified the need to develop sign language technologies to facilitate communication and social inclusion of people with hearing loss. Although developing such technologies can be difficult due to the existence of many sign languages and the lack of large, annotated datasets, recent developments in artificial intelligence and machine learning have played a key role in automating and enhancing these technologies.

4.2.1 Historical of sign-language

For thousands of years, people with marginal hearing loss believed they could learn language only through the spoken word. For example, the ancient Greek philosopher Aristotle asserted that “deaf men are, after all, also stupid.” “Under Roman law, people who were born deaf were denied the right to sign a will because they were not supposed to understand anything; Because he will not be able to intend to write or read. " Specific language for centuries. No one knows exactly when it was born, but it is believed that it is used because people need to communicate. Many cultures have used a particular language throughout history, even before it became a language with its own system.

4.2.2 There is no single sign language

Used all over the world. Like any universal language, languages were created naturally by different group of people interacting with many players, so there are many options. There are between 138 and 300 different types of languages used around the world throughout the day, Interestingly, most countries that share the same common language do not necessarily have the same language as part of a large portion of their words. English, for example, has three variants: American Sign Language (ASL), British Sign Language and Australian Sign Language (Auslan).

Sign Language Alphabets from Around the World

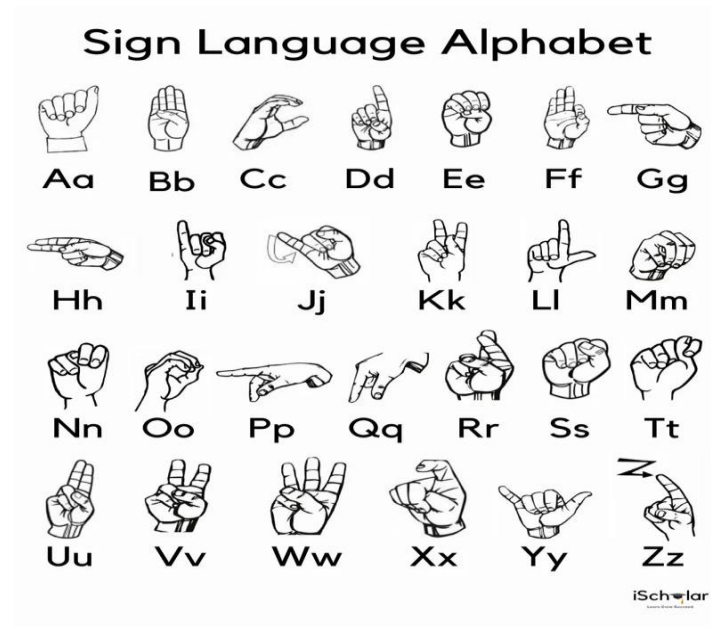


Figure 3 american sign language alphabet

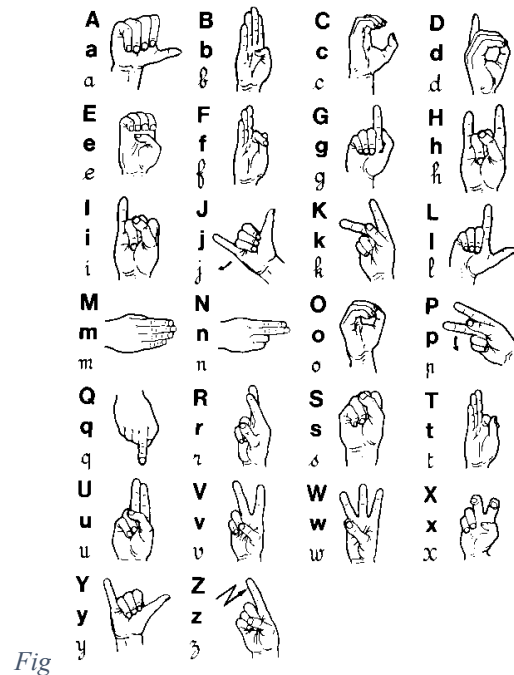


Figure 4 french sign language alphabet



Figure 5 Arabic sign language alphabet

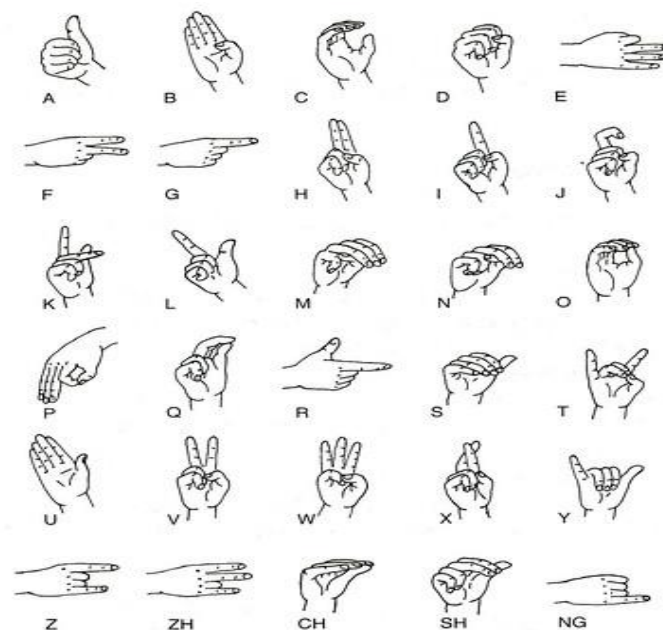


图8 汉语手指字母

Figure 6 Chinese sign language

4.3 An intelligent diagnosis system

An intelligent diagnosis system is a new concept that makes a patient's life easier using advanced information technology.

Deaf's disease makes you alive but not living and There is no medication for deaf disease, But healthcare programmer have been successful in helping people maintain their communicating function and understand their behavior.

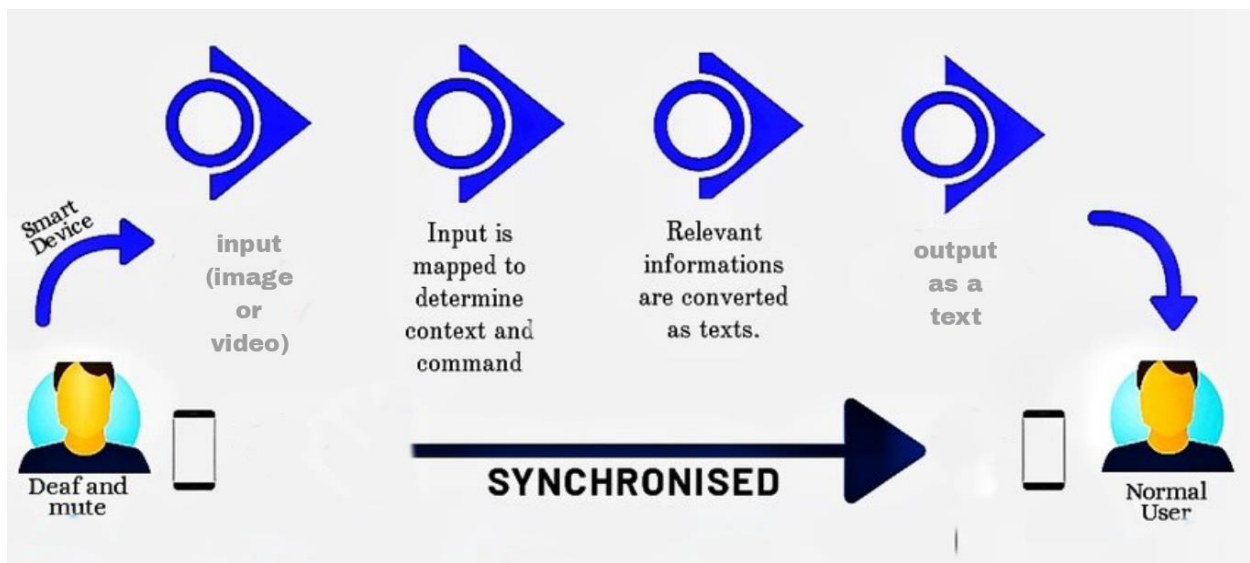


Figure 7 model road map

4.4 Data preprocessing

As sign language detection software naturally has to use cameras as input devices, this makes computer vision algorithms an important part of the implementation. For detection solutions with machine learning they typically make up a large part of the preprocessing pipeline by

4.4.1 image scaling

Image scaling is the task of resizing a digital image to a desired number of pixels in width and length, without removing parts from the image. This can either be done to a higher resolution, also known as up-scaling

or up sampling, or to a lower resolution, also called downscaling or down sampling.

4.4.2 image noise reduction

When filming with any digital camera, the image can never be perfectly captured. This is a problem of every physical measuring process: The measured value y is never the exact value x , it always diverges by a random error E . Mathematically this can be modeled as $y = x + E$, where E is called the random gaussian measurement noise.

4.4.3 image segmentation

Image segmentation is the task of recognizing related pixels in an image. For example separating an object from the background, or separating multiple objects from another.

4.4.4 skin color detection

Skin color detection is a subcategory of image segmentation specifically aimed at detecting humans, and therefore very suitable for sign language detection. It aims to find accumulations of pixels within a certain color range in the image.

4.4.5 bounding boxes

The term "bounding boxes" refers to a technique which combines other preprocessing steps with human knowledge about the task we want to solve.

4.5 feature engineering

Feature engineering is defined as the process of extracting features that better represent the underlying problem from raw data and transforming them into a suitable format for machine learning models, resulting in improving the performance of a trained model on unseen data. Feature engineering can be categorized into three categories, namely, feature extraction, feature selection and feature fusion as shown in figure the main aim of feature extraction and selection is to:

- Reduce high-dimensional feature space to low-dimensional representation
- Focus on the most relevant data
- Avoid overfitting the data
- Improve the quality of feature space and hence the performance of machine learning algorithms such as learning time and accuracy.

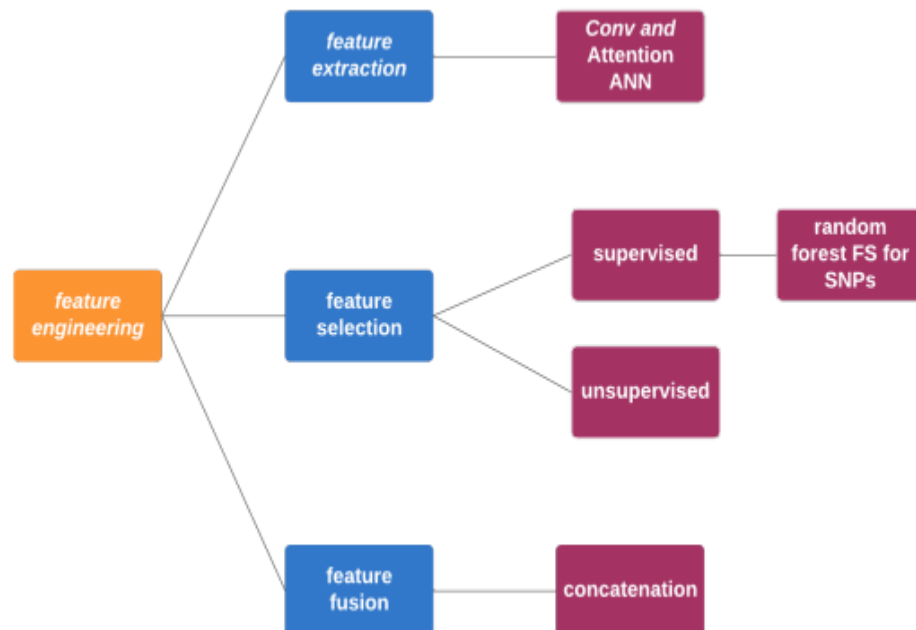


Figure 8feature sengineering

4.5.1 feature extraction

Feature extraction is the process of transforming raw data into a set of meaningful and relevant features that can be used as input to a machine learning model. Neural Network is used to extract feature extraction.

4.5.2 feature selection

Feature Selection Selecting the optimal subset of distinct features plays a key role in improving the performance of a classification model with lower computational effort, shorter learning time, data visualization, refined understanding of computational models, low risk of data overfitting, and lower dimensions of the problem. Feature selection picks out the subset of the features with maximum relevance to the target class and minimal redundancy from the original set to increase the classification accuracy. FS methods can be classified into:

1. supervised learning, class labels are specified beforehand, and the algorithms maximize some functions to select the relevant features, which are highly correlated with the class.
2. unsupervised learning, class labels are not given resulting in difficulty in finding relevant features simultaneously

Embedded methods

Embedded methods combine both filter and wrapper methods, where it is a built-in feature selection method embedding the feature selection in the learning algorithm and utilizing its properties to lead feature assessment. This makes the embedded method more efficient and tractable than the wrapper one, with similar performance and has a lower risk of overfitting. Similar to the wrapper method, they take into account the dependencies among features but is only specific to a given learning algorithm. Figure shows the embedded feature selection approach.

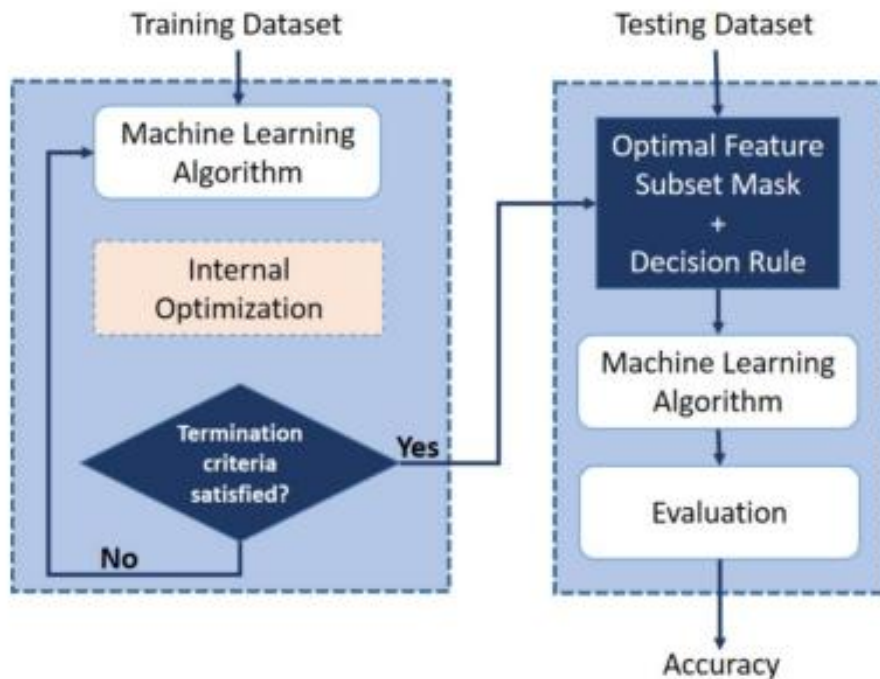


Figure 9feature selection

Random forest

Feature selection using Random forest comes under the category of Embedded methods. Embedded methods combine the qualities of filter and wrapper methods. They are implemented by algorithms that have their own built-in feature selection methods. Some of the benefits of embedded methods are :

1. They are highly accurate.
2. They generalize better.
3. They are interpretable

How does Random forest select features?

Random forests consist of 4 –12 hundred decision trees, each of them built over a random extraction of the observations from the dataset and a random extraction of the features. Not every tree sees all the features or

all the observations, and this guarantees that the trees are de-correlated and therefore less prone to over-fitting. Each tree is also a sequence of yes-no questions based on a single or combination of features. At each node (this is at each question), the tree divides the dataset into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket. Therefore, the importance of each feature is derived from how “pure” each of the buckets is.

- For classification, the measure of impurity is either the Gini impurity or the information gain/entropy.
- For regression the measure of impurity is variance.

Therefore, when training a tree, it is possible to compute how much each feature decreases the impurity. The more a feature decreases the impurity, the more important the feature is. In random forests, the impurity decrease from each feature can be averaged across trees to determine the final importance of the variable.

To give a better intuition, features that are selected at the top of the trees are in general more important than features that are selected at the end nodes of the trees, as generally the top splits lead to bigger information gains.

4.5.3 Feature fusion

Feature fusion, the combination of features from different layers or branches, is an omnipresent part of modern network architectures. It is often implemented via simple operations, such as summation or concatenation. Feature fusion attempts to extract the most discriminative information from several input features and eliminate redundant information. Feature Fusion may play an important role in improving the performance of classification tasks, in case of features Independence. To perform feature fusion, two or more different feature vectors are concatenated into one vector. Two main problems face the task of

feature fusion; namely; the compatibility of different features and the high dimensionality. To solve the first problem, a normalization step is performed to transform the features within a range $[0, 1]$.

4.6 deep learning

In the last few years, the deep learning (DL) computing paradigm has been deemed the Gold Standard in the machine learning (ML) has gradually become the most widely used computational approach in the field of ML, thus achieving outstanding results on several complex cognitive tasks, matching or even beating those provided by human performance. One of the benefits of DL is the ability to learn massive amounts of data. The DL field has grown fast in the last few years and it has been extensively used to successfully address a wide range of traditional applications. More importantly, DL has outperformed well-known ML techniques in many domains, e.g., cyber security, natural language processing, bioinformatics, robotics and control, and medical information processing, among many others.

Chapter 5. Proposed System

5.1 Overview

This chapter reviews the Development methodology also reviews the functional and non-functional requirements. It also discusses System analysis through the life cycle and its requirements showing the USE-CASE Diagram which is a list of actions or events steps typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system to achieve a goal, Activity diagram which is a graphical representation of workflows of stepwise activities and actions and process modeling.

5.2 Proposed framework

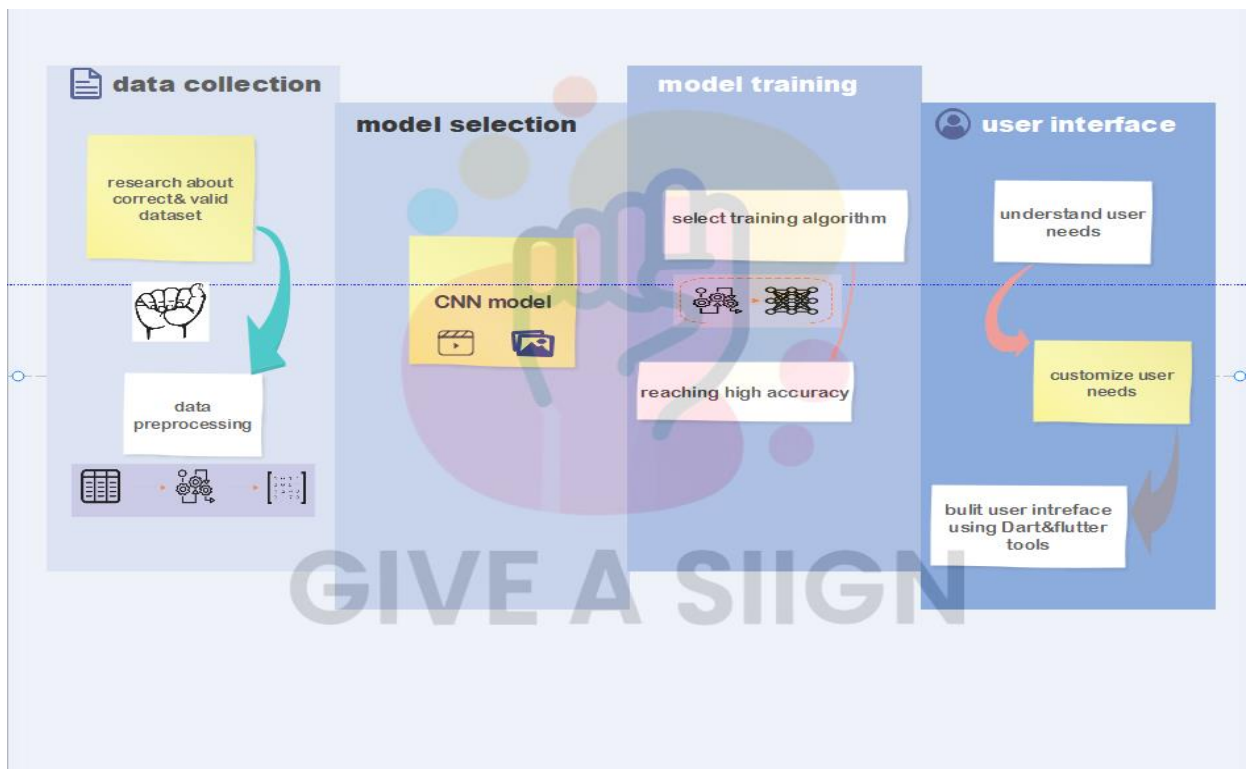


Figure 10framework diagram

5.3 System analysis

Through this chapter we are going through the project requirements that the system must satisfy are of two types, which are functional and non-functional requirements. Functional requirements are the requirements that define a function of the software that runs on the system. Non-functional requirements are the requirements that specify criteria that can be used to judge the operation of the system, rather than specific behaviors. In the following subsections, both functional and non-functional requirements of the proposed system are listed

5.3.1 Process modeling

Context Diagram

The Context Diagram shows the system under consideration as a single high level process and then shows the relationship that the system has with other external entities

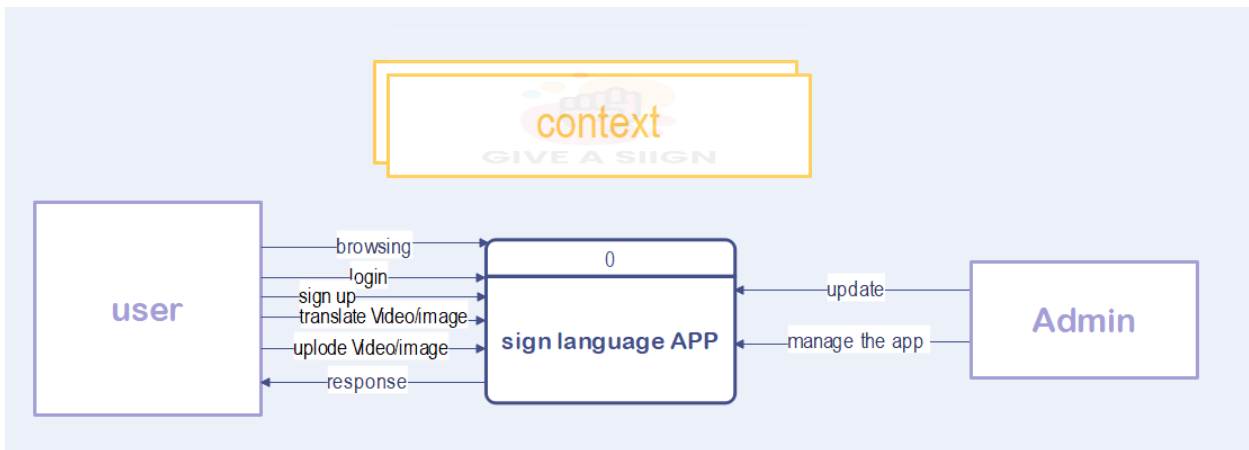


Figure 11context diagram

5.3.2 Data flow diagram

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes, and how it gets stored. It provides an overview of :

- What data is the system process?
- What transformations are performed?
- What data are stored?
- What results are produced, etc.

5.3.3 Requirements

Functional Requirements

Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements. Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements.

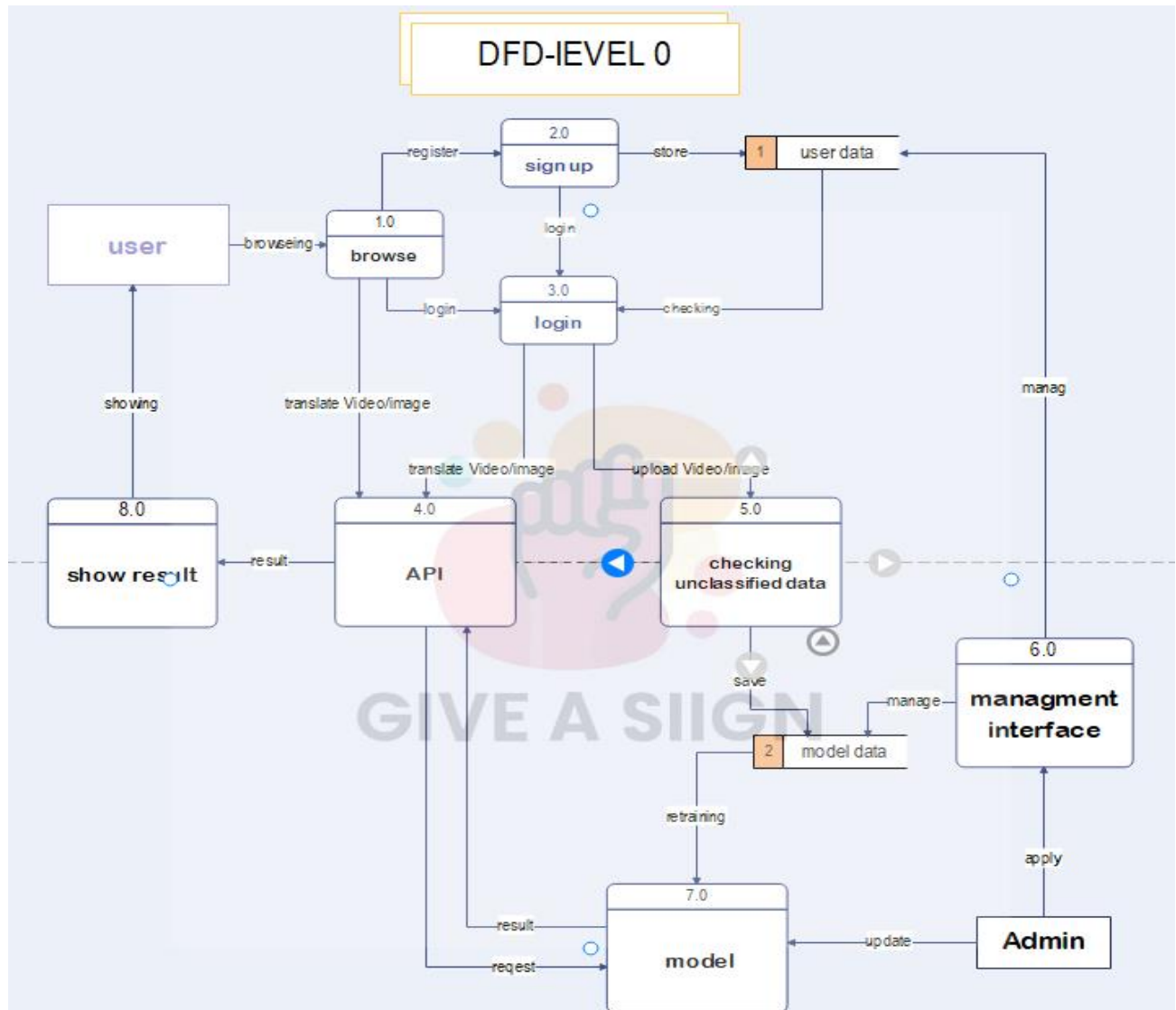


Figure 12 data flow diagram

5.3.4 Use case diagrams

- it represents system functionality from the user's perspective.
- describes who will use the system and in what ways the user expects to interact with the system.
- represents the interactions between use cases and actors.

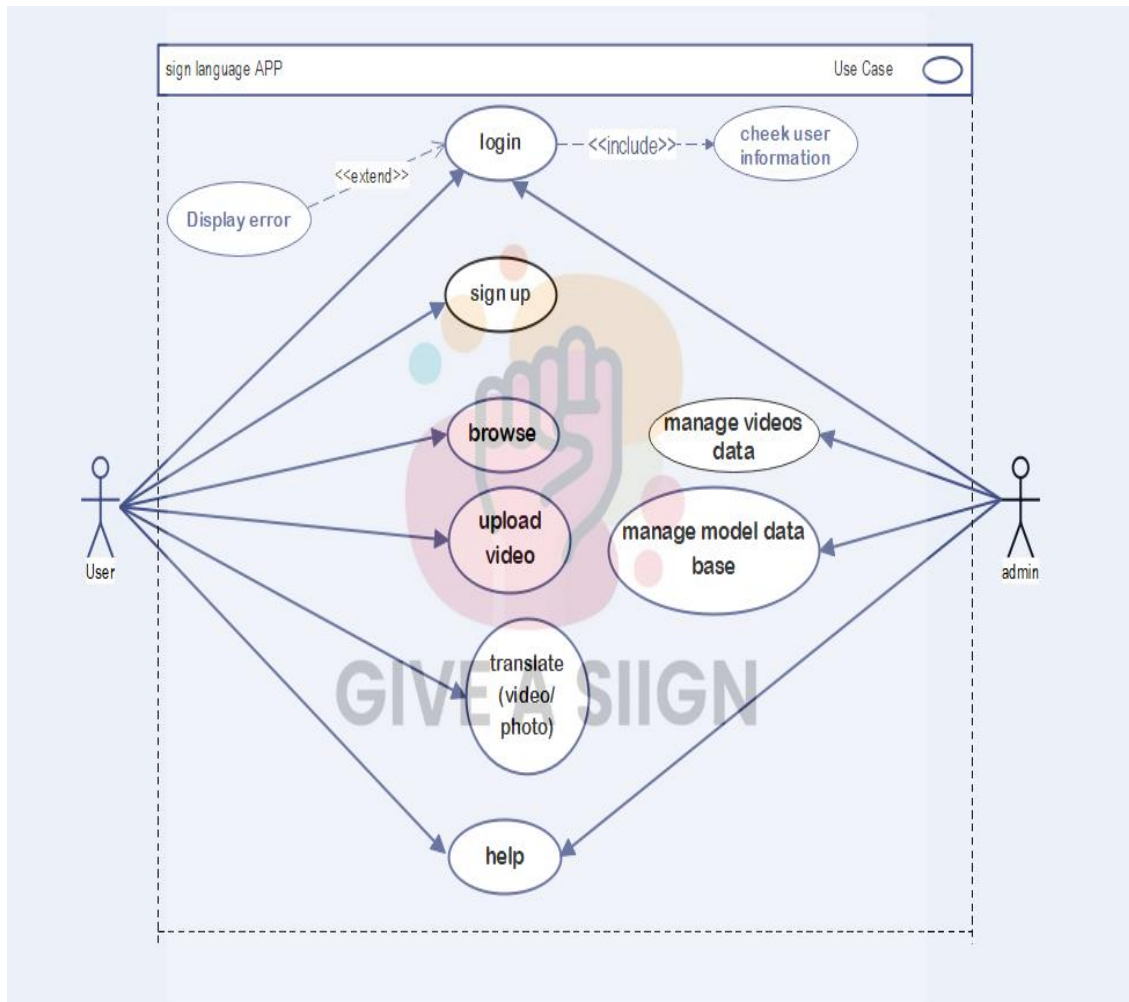


Figure 13 use case diagram

5.3.5 Use case scenario

A use case Scenario represents the sequence of events along with other information that relates to this use case. A typical use case specification template includes the following information:

- Description
- Pre- and Post- interaction condition
- Basic interaction path
- Alternative path

| | | |
|-----------------------------------|---|---|
| Use case Name: | browse | |
| Actor(s): | user | |
| Description | This use case describes the process of a user enter the app; and look for different features. | |
| Typical Course of Events: | Actor Action | System Response |
| | <u>Step 1:</u> This use case will be done when the user download the application and accept the rights | <u>Step2:</u> System will show all pages except the pages that require login |
| Alternate Courses: | <u>Step 6:</u> if user choose features that require login send notification to login or sign up | |
| Precondition: | App can only be entered by user agreed the app rights | |
| Post condition/ Assumption | non | |

| | | |
|----------------------------------|---|---|
| Use case Name: | login | |
| Actor(s): | user | |
| Description: | This use case describes the process of a user login to the app of Give A Sign to can make different action. | |
| Typical Course of Events: | Actor Action | System Response |
| | <u>Step 1:</u> This use case will be done when the user have account in site <u>Step 2:</u> User will enter personal information | <u>Step3:</u> System will cheek about information that the user enter it |
| Alternate Courses: | <u>Step 4:</u> If the personal information is not valid, send a notification to the user requesting the user to submit a valid information | |
| Precondition: | Login must be done by valid user. | |
| Post condition: | Action has been recorded | |
| Assumption: | Login from labtop or phone | |

5.3.6 Activity diagram

| | | |
|----------------------------------|--|---|
| Use case Name: | Translate video/photo | |
| Actor(s): | user | |
| Description: | This use case describes the process of translate video/photo from sign language to text in app | |
| Typical Course of Events: | Actor Action | System Response |
| | <u>Step 1:</u> This use case will be done when the user upload video/photo to be translated | <u>Step2:</u> System will show result as a text in the app |
| Alternate Courses: | <u>Step 3:</u>if video/photo can be translated <u>Step 4:</u> If the video/photo not valid, send a notification to the user requesting the user to submit a valid video/photo . <u>Step 4:</u> If the video/photo can not be recognize send a notification to the user that he/she can upload video/photo to be save in model DB. | |
| Precondition: | translate can only be submitted by valid user that login to app | |
| Post condition: | Action has been recorded in user BD/model DB | |
| Assumption: | User can not read | |

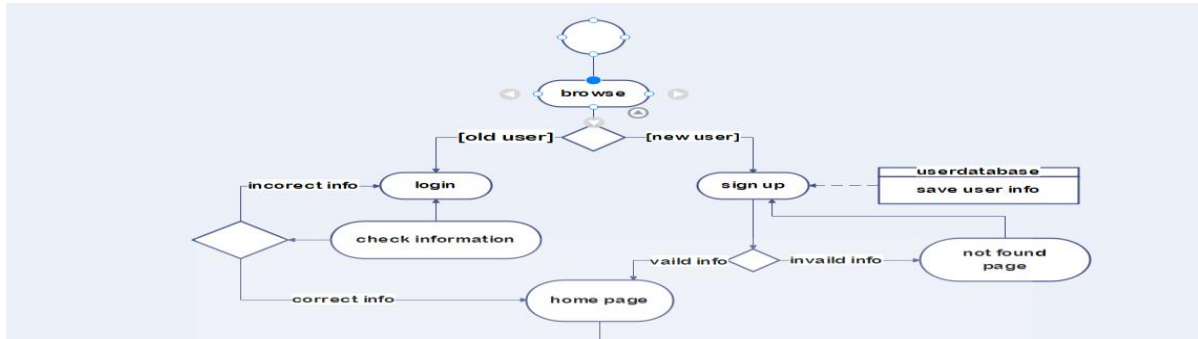
Figure 14use case analysis

An activity diagram gives a graphical representation of how data move aro

Figure 15activity diagram

5.3.7 Non functional requirements

Non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behavior. They are contrasted with functional requirements that



define specific behavior or functions.

- Usability: The Mobile Application cart should be easily usable by the Patient.
- Accessibility: the caregiver will receive the Call
- Performance: The main function must have specific real-time to be executed for each operation to not delay the whole system
- Security: The patient must be sure that all his details will be secured in the App options.
- User-friendly: Easy to use and manipulate every proposed feature

5.4 system design

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces, and data for a system based on the specified requirements .

5.4.1 Sequence diagram

sequence diagram given is used to show the interactive behavior of the system.

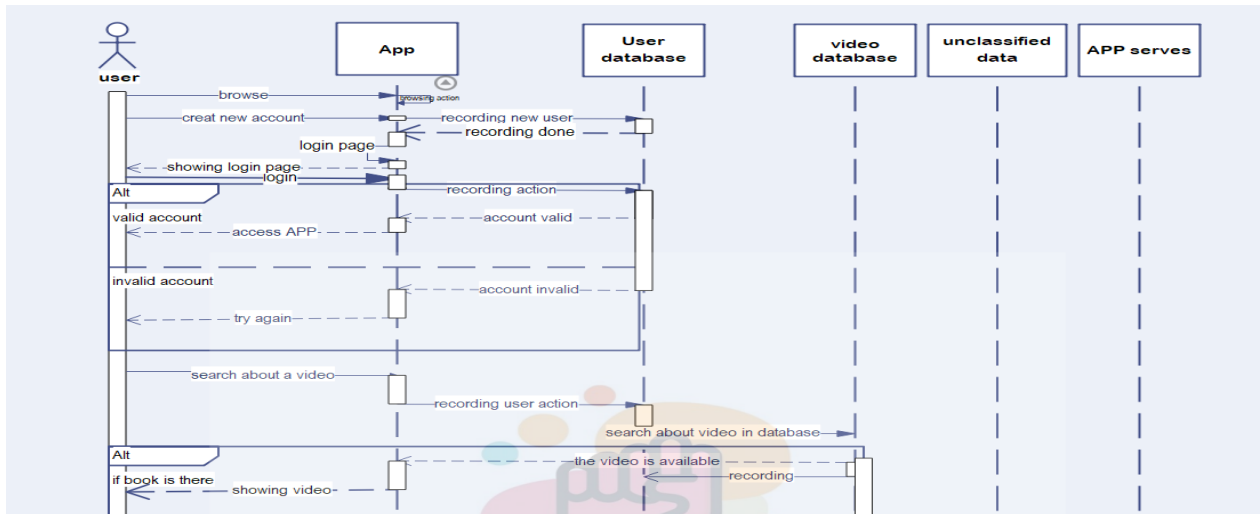
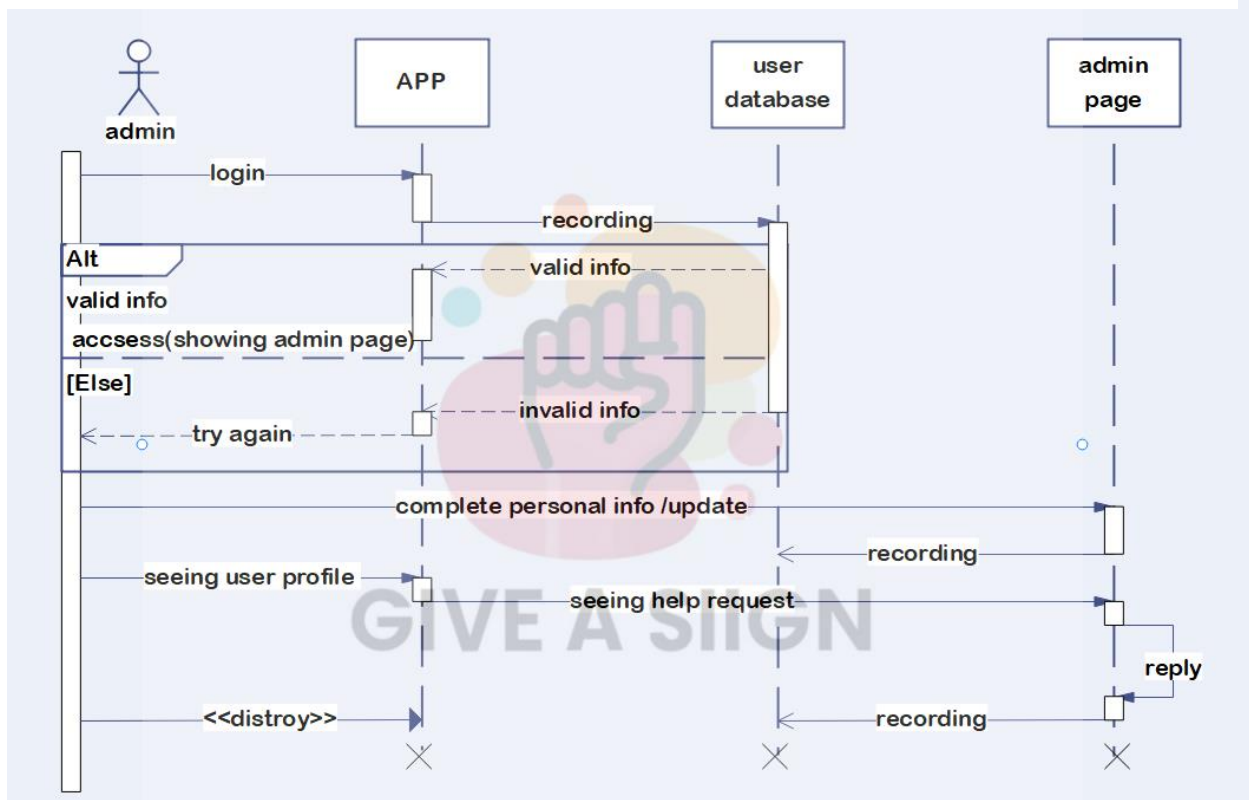


Figure 17 user sequence diagram



48
Figure 16 admin sequence diagram

Chapter 6. Experimental Results and Comparative Analysis for YOLO v8

6.1 Results of The System

6.1.1 Dataset description

Most of the data obtained by ourselves. As such, we and our friends who contributed to the design or provided data, then we make annotation on Roboflow. complete listing of data can be found at: <https://universe.roboflow.com/sign-language-6ccbk/final-isolt> for Arabic dataset and <https://universe.roboflow.com/sign-language-6ccbk/en-give-a-sign> for English dataset . The primary goal of creating a dataset of photos for a sign language project includes several key objectives:

Recognition and Translation: The primary goal is to develop machine learning models that can accurately recognize and translate sign language gestures into text or speech. This helps bridge the communication gap between sign language users and non-users.

Training and Validation: A large and diverse dataset of sign language photos is essential for training deep learning models. It allows the models to learn the variations in gestures made by different individuals and in different lighting or environmental conditions. The dataset also serves as a benchmark for validating the model's performance.

Assistive Technology Development: The dataset can be used to develop assistive technologies such as real-time sign language interpretation apps, wearable devices, or smart cameras that help sign language users communicate more effectively in various settings, including education, healthcare, and customer service.

Educational Tools: Creating a comprehensive dataset can support the development of educational tools and resources for teaching sign language. These tools can be used by both deaf and hearing individuals to learn and practice sign language more interactively and effectively.

Inclusivity and Accessibility: By facilitating the development of technologies that understand and translate sign language, the dataset helps promote inclusivity and accessibility for deaf and hard-of-hearing individuals. It enables them to participate more fully in society, breaking down communication barriers.

Research and Innovation: The dataset serves as a valuable resource for researchers exploring new methods and algorithms in computer vision, natural language processing, and human-computer interaction. It encourages innovation in the field of sign language recognition.

This dataset consists of 2 models including many classes of different stages of different category sign languagetotal number of images are 1612 of 42 class for Arabic detection and . It is also openly available on the internet. Each class contains approximately 30 image and its label.

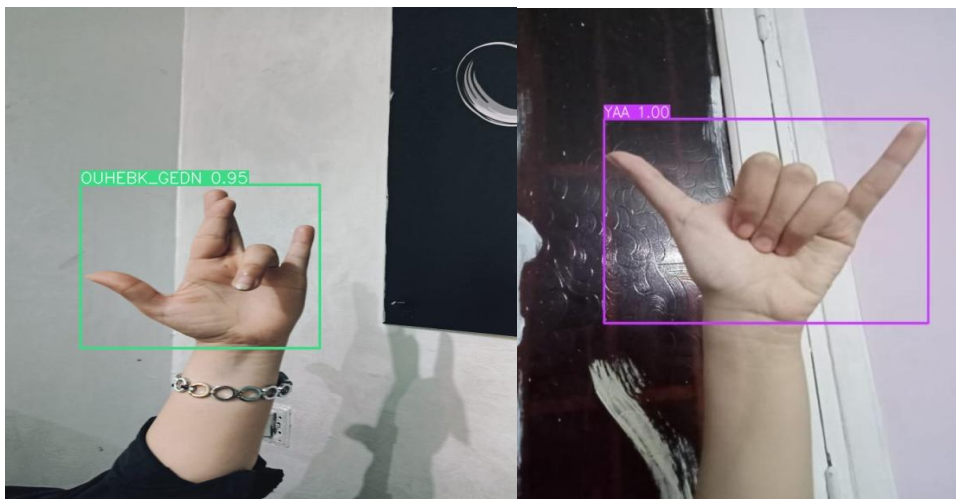


Figure 18 detection of sign language

6.2 Comparative analysis

This section details the experiments conducted to evaluate the performance of the YOLO v8 model. But firstly we need to know what is YOLO ?

What is YOLO?

Who develop YOLO?

YOLO (You Only Look Once) is a real-time object detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015. It is a single-stage object detector that uses a convolutional neural network (CNN) to predict the bounding boxes and class probabilities of objects in input images. YOLO was first implemented using the Darknet framework.

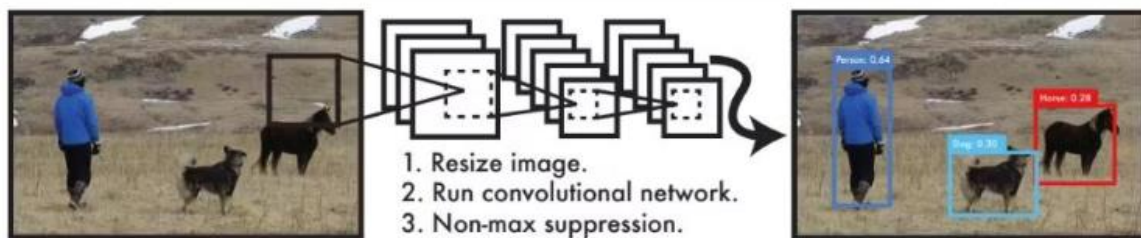


Figure 19 YOLO idea

The YOLO algorithm divides the input image into a grid of cells, and for each cell, it predicts the probability of the presence of an object and the bounding box coordinates of the object. It also predicts the class of the object. Unlike two-stage object detectors such as R-CNN and its variants, YOLO processes the entire image in one pass, making it faster and more efficient.

What is YOLO's versions?

YOLO has been developed in several versions, such as YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7 and YOLOv8. Each version has been built on top of the previous version with enhanced features such as improved accuracy, faster processing, and better handling of small objects.

What is importance of YOLO ?

YOLO is widely used in various applications such as self-driving cars and surveillance systems. It is also widely used for real-time object detection tasks like in real-time video analytics and real-time video surveillance.

YOLO Algorithm: How Does it Works?

The basic idea behind YOLO is to divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object. The process of YOLO can be broken down into several steps:

1. Input image is passed through a CNN to extract features from the image.
2. The features are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.
3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and class probabilities.
4. The output of the network is a set of bounding boxes and class probabilities for each cell.

5. The bounding boxes are then filtered using a post-processing algorithm called non-max suppression to remove overlapping boxes and choose the box with the highest probability.
 6. The final output is a set of predicted bounding boxes and class labels for each object in the image.
- One of the key advantages of YOLO is that it processes the entire image in one pass, making it faster and more efficient than two-stage object detectors such as R-CNN and its variants.

Structure of YOLO

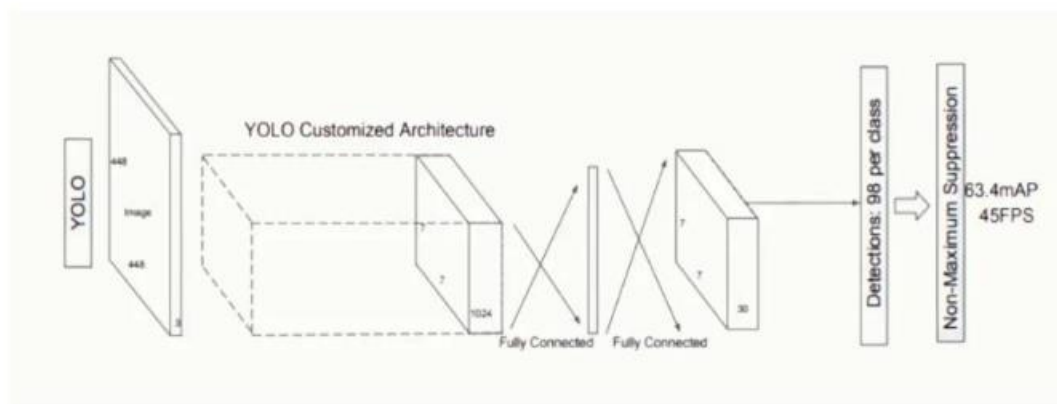


Figure 20 structure of YOLO

6.2.1 YOLO versions

YOLOv1: First Real-time Object Detection Algorithm

The original YOLO model treated object detection as a regression problem, which was a significant shift from the traditional classification approach. It used a single convolutional neural network (CNN) to detect objects in images by dividing the image into a grid, making multiple

predictions per grid cell, filtering out low-confidence predictions, and then removing overlapping boxes to produce the final output.

YOLO v2

- YOLO v2, also known as YOLO 9000, is an improved version of the original YOLO object detection algorithm. It builds upon the concepts and architecture of YOLO, but addresses some of the limitations of the original version.

One of the main differences between YOLO v2 and the original YOLO is the use of anchor boxes. In YOLO v2, CNN predicts not only the bounding box coordinates but also the anchor boxes. Anchor boxes are pre-defined boxes of different aspect ratios and scales, which are used to match the predicted bounding boxes with the actual objects in the image. This allows YOLO v2 to handle objects of different shapes and sizes better.

Another key difference is the use of a multi-scale approach. In YOLO v2, the input image is fed through CNN at multiple scales, which allows the model to detect objects at different sizes. This is achieved by using a feature pyramid network (FPN), which allows the model to extract features at different scales from the same image.

Additionally, YOLO v2 uses a different loss function than the original YOLO, called the sum-squared error (SSE) loss function. The SSE loss function is more robust and helps the model to converge faster.

In terms of architecture, YOLO v2 uses a slightly deeper CNN than YOLO, which allows it to extract more powerful features from the image. The CNN is followed by several fully connected layers, which predict class probabilities and bounding box coordinates.

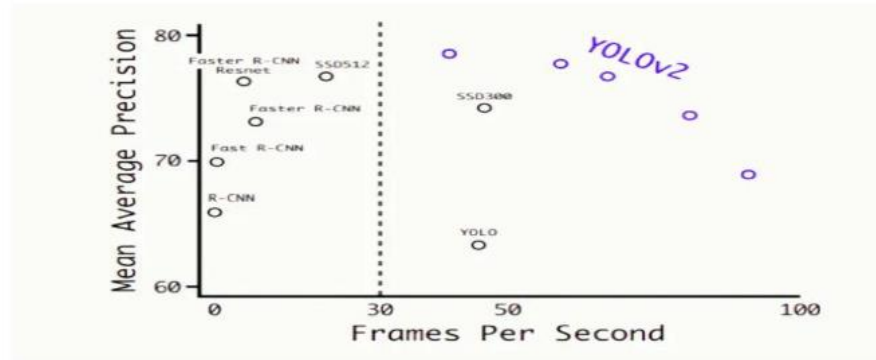


Figure 21MAP & FPS of YOLOv2

YOLO v3

YOLO v3 is the third version of the YOLO object detection algorithm. The first difference between YOLO v3 and previous versions is the use of multiple scales in the input image. YOLO v3 uses a technique called "feature pyramid network" (FPN) to extract features from the image at different scales. This allows the model to detect objects of different sizes in the image.

Another important difference is the use of anchor boxes. In YOLO v3, anchor boxes are used to match the predicted bounding boxes with the actual objects in the image. Anchor boxes are pre-defined boxes of different aspect ratios and scales, and the model predicts the offset of the anchor boxes relative to the bounding boxes. This helps the model to handle objects of different shapes and sizes better.

In terms of architecture, YOLO v3 is built on a deep convolutional neural network (CNN) that is composed of many layers of filters. The CNN is followed by several fully connected layers, which predict class probabilities and bounding box coordinates.

YOLO v3 also uses a different loss function than previous versions. It uses a combination of classification loss and localization loss, which

allows the model to learn both the class probabilities and the bounding box coordinates.

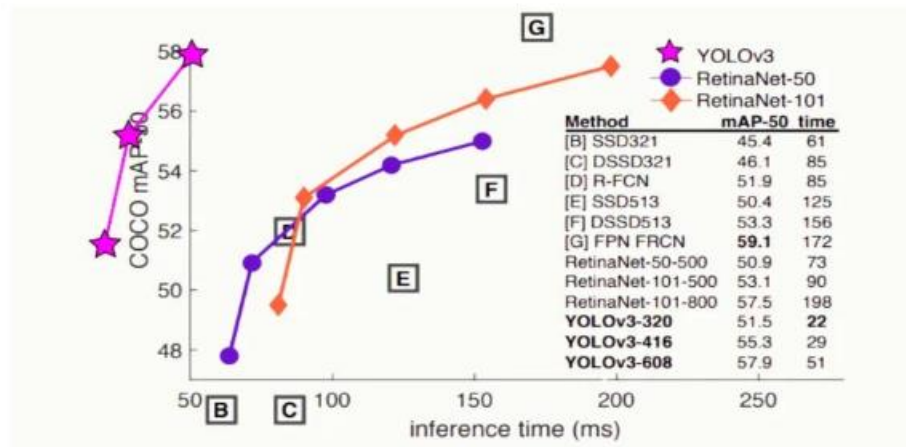


Figure 22 Speed & MAP compared with other detectors

YOLO v4

Now, let's take a look at one of the most important versions of YOLO. A key distinction between YOLO v4 and previous versions is using a more advanced neural network architecture. YOLO v4 uses a technique called "Spatial Pyramid Processing" (SPP) to extract features from the image at different scales and resolutions. This allows the model to detect objects of different sizes in the image.

Additionally, YOLO v4 also uses a technique called "Cross-stage partial connection" (CSP) to improve the model's accuracy. It uses a combination of multiple models with different architectures and scales and combines their predictions to achieve better accuracy.

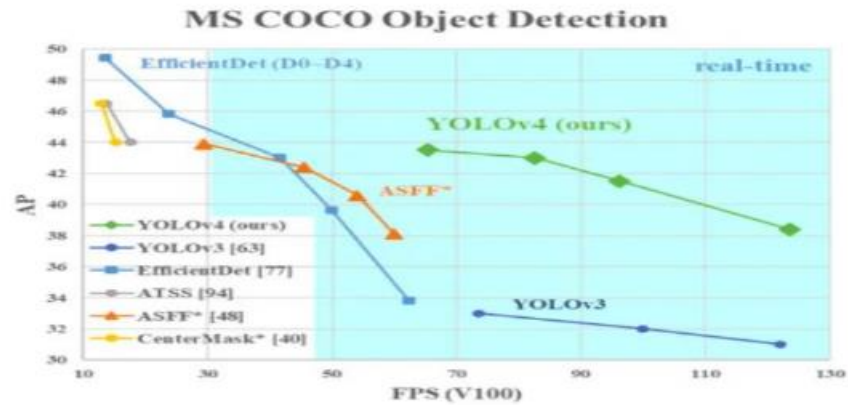


Figure 23 Accuracy & FPS compared with other detectors

YOLO v5

YOLO v5 was introduced in 2020 with a key difference from the previous versions, which is the use of a more efficient neural network architecture called EfficientDet, which is based on the EfficientNet architecture. EfficientDet is a family of image classification models that have achieved state-of-the-art performance on a number of benchmark datasets. The EfficientDet architecture is designed to be efficient in terms of computation and memory usage while also achieving high accuracy.

Another important difference is the use of anchor-free detection, which eliminates the need for anchor boxes used in previous versions of YOLO. Instead of anchor boxes, YOLO v5 uses a single convolutional layer to predict the bounding box coordinates directly, which allows the model to be more flexible and adaptable to different object shapes and sizes.

YOLO v5 also uses a technique called "Cross mini-batch normalization" (CmBN) to improve the model's accuracy. CmBN is a variant of the

standard batch normalization technique that is used to normalize the activations of the neural network.

Regarding training, YOLO v5 uses transfer learning, which allows it to be pre-trained on a large dataset and then fine-tuned on a smaller dataset. This allows the model to learn from a wide range of data and generalize better to new data.

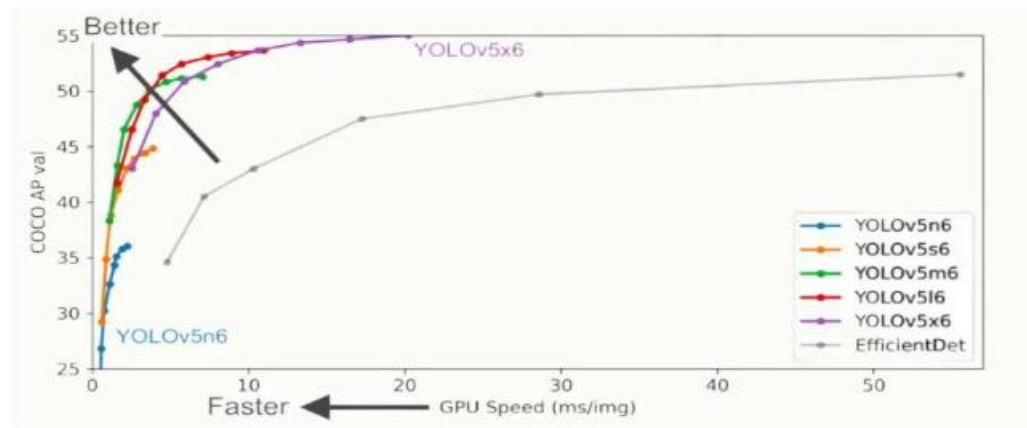


Figure 24 Speed of YOLO v5 compared with EfficientDet

YOLO v6

A notable contrast between YOLO v6 and previous versions is the use of a more efficient and lightweight neural network architecture; this allows YOLO v6 to run faster and with fewer computational resources. The architecture of YOLO v6 is based on the "EfficientNet-Lite" family, which is a set of lightweight models that can be run on various devices with limited computational resources.

YOLO v6 also incorporates data augmentation techniques to improve the robustness and generalization of the model. This is done by applying random transformations to the input images during training, such as rotation, scaling, and flipping.

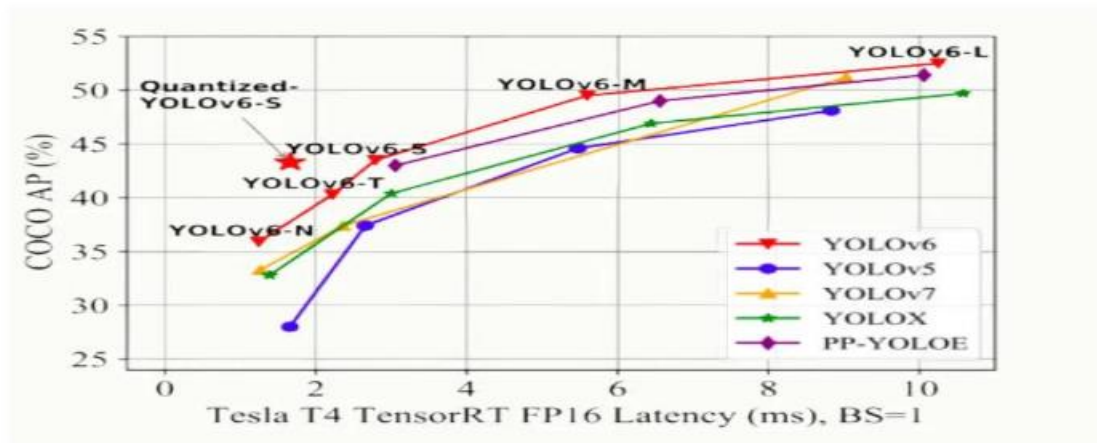


Figure 25 Accuracy & Speed

YOLO v7

YOLO v7, on the most recent stable iterations of the YOLO algorithm. It boasts a number of enhancements compared to previous versions. A key enhancement is the implementation of anchor boxes. These anchor boxes, which come in various aspect ratios, are utilized to identify objects of various shapes. The use of nine anchor boxes in YOLO v7 enables it to detect a wider range of object shapes and sizes, leading to a decrease in false positives.

In YOLO v7, a new loss function called "focal loss" is implemented to enhance performance. Unlike the standard cross-entropy loss function used in previous versions of YOLO, focal loss addresses the difficulty in detecting small objects by adjusting the weight of the loss on well-classified examples and placing more emphasis on challenging examples to detect.

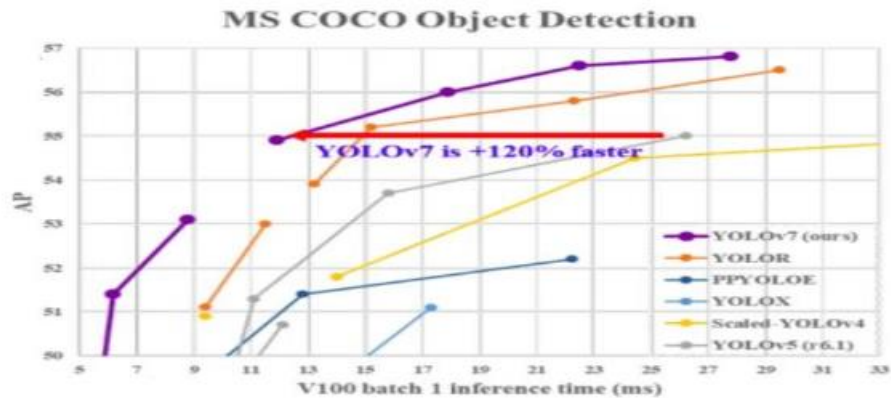


Figure 26 Accuracy & Speed

YOLOv8

YOLOv8 was developed by Ultralytics, who also created the influential and industry-defining YOLOv5 model. YOLOv8 includes numerous architectural and developer experience changes and improvements over YOLOv5. YOLOv8 comes with a lot of developer-convenience features, from an easy-to-use CLI to a well-structured Python package. YOLOv8 is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box.

One key technique introduced in YOLOv8 is multi-scale object detection. This technique allows the model to detect objects of various sizes in an image. Another significant enhancement in YOLOv8 is the use of the ELU activation function. ELU, or Exponential Linear Unit, helps to speed up learning in deep neural networks by mitigating the vanishing gradient problem, leading to faster convergence.

YOLOv8 adopted the GIoU loss. GIoU, or Generalized Intersection over Union, is a more advanced version of the IoU (Intersection over Union) metric that takes into account the shape and size of the bounding boxes, improving the precision of object localization.

The YOLOv8 algorithm shows a 1.2% increase in average precision (AP) compared to the YOLOv7, which is a significant improvement. It has achieved this while reducing the model's weight file size by 80.6 M, making the model more efficient and easier to deploy in resource-constrained environments.

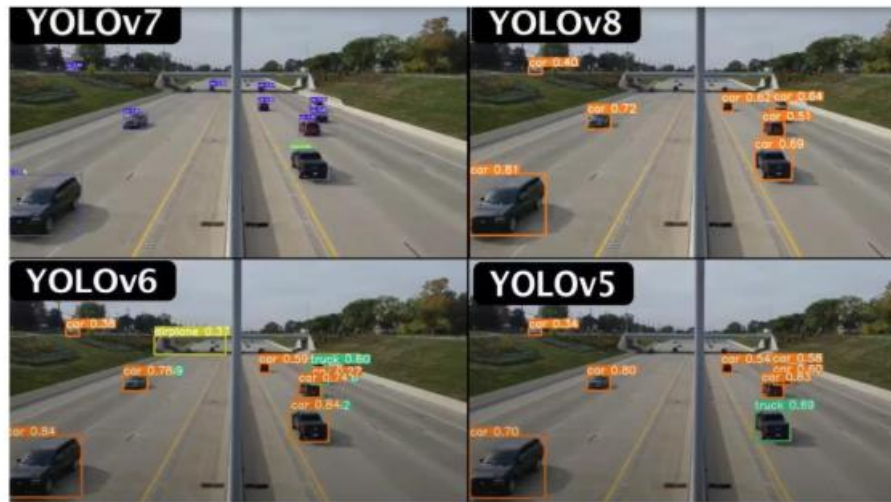


Figure 27 YOLOv8 Comparison with Latest YOLO models

YOLO Algorithms: Fortes

YOLO is widely used in real-world projects because of its accuracy and speed; its main powerful sides can be listed like the following:

Real-time object detection: YOLO is able to detect objects in real-time, making it suitable for applications such as video surveillance or self-driving cars.

High accuracy: YOLO achieves high accuracy by using a convolutional neural network (CNN) to predict both the class and location of objects in an image.

Single-shot detection: YOLO can detect objects in an image with just one forward pass through the network, making it more efficient than other object detection methods that require multiple passes.

Good performance on small objects: YOLO is able to detect small objects in an image because of its grid-based approach.

Efficient use of GPUs: YOLO uses a fully convolutional network architecture, which allows for the efficient use of GPUs during training and inference.

Ability to handle multiple scales: YOLO uses anchor boxes, which allows the model to handle objects of different scales, thus allowing the model to detect objects of different sizes in the same image.

YOLO Algorithm: Limitations

Even though YOLO is a powerful object detection algorithm, it also has some limitations. Some of these limitations include:

Limited to object detection: YOLO is primarily designed for object detection and may not perform as well on other tasks such as image segmentation or instance segmentation.

Less accurate than some other methods: While YOLO is accurate, it may not be as accurate as two-shot object detection methods, such as RetinaNet or Mask R-CNN.

Struggles with very small objects: YOLO's grid-based approach can make it difficult to detect tiny objects, especially if they are located close to other objects.

No tracking capability: YOLO does not provide any tracking capability, so it may not be suitable for video surveillance applications that require tracking of objects over time.

Why should I use a YOLO model?

YOLO models are popular for a variety of reasons. First, YOLO models are fast. This enables you to use YOLO models to process video feeds at a high frames-per-second rate. This is useful if you are planning to run

live inference on a video camera to track something that changes quickly (i.e. the position of a ball on a football court, or the position of a package on a conveyor belt).

Second, YOLO models continue to lead the way in terms of accuracy. The YOLOv7 model, the latest in the family of models as of November 2022, has state-of-the-art performance when measured against the MS COCO object detection dataset. While there are other great models out there, YOLO has a strong reputation for its accuracy.

Third, the YOLO family of models are open source, which has created a vast community of YOLO enthusiasts who are actively working with and discussing these models. This means that there is no shortage of information on the web about the how and why behind these models, and getting help is not difficult if you reach out to the computer vision community.

| | Faster RCNN | SSD | YOLO |
|---|---|---|---|
| 1 | It is a two-stage detector | It is a single stage detector | It is a single stage detector |
| 2 | Localization involves anchor boxes and Unit Box. | Localization involves default boxes or prior boxes. | Localization involves anchor boxes. |
| 3 | Utilizes Regional Proposal Networks to generate anchor boxes | No use of Regional Proposal Networks | Utilizes regression and CNN layers to generate the grids for object detection |
| 4 | Input video size can be of 300 x 300 or 512 x 512 or 800 x 800 pixels | Input video size can be of 224 x 224 or 227 x 227 or 512 x 512 pixels | Input video size can be of 416 x 416 or 512 x 512 or 608 x 608 pixels |
| 5 | It works efficiently in face detection and digital image processing | It works efficiently in video surveillance and tumor detection | It works efficiently in real time vehicle and pedestrian detection. |

| | | | |
|---|---|--|--|
| 6 | It cannot perform well for objects with large shape variations. | It cannot perform well for smaller objects as the layers in SSD are low resolution layers. | It cannot perform well for zoom in object detection where the object size is tiny and the system has limited memory capacity |
| 7 | Here the RPN is used along with the Fast RCNN detector. | Here input image is scanned once and involves multi scale features and default boxes | Here the input image is scanned once only and feature extraction, regression and probability-based classification is used |
| 8 | Accuracy of Faster RCNN algorithm on COCO dataset is 48.2% | Accuracy of SSD algorithm on COCO dataset is 49.8% | Accuracy of YOLOV2 and YOLOV3 algorithms on COCO dataset are 44% and 51.2% respectively. |

Figure 28 Comparison of Faster RCNN , SSD and YOLO algorithms

YOLOv8

YOLOv8 is a cutting-edge deep learning model designed for real-time object detection in computer vision tasks. It's the latest iteration in the popular YOLO (You Only Look Once) family of algorithms, known for their speed and precision. Here's a breakdown of its key features:

State-of-the-Art Performance: YOLOv8 boasts advancements in both accuracy and speed compared to previous YOLO versions.

Real-Time Object Detection: One of its strengths is the ability to detect objects in videos or image streams very quickly, making it suitable for applications that require immediate results.

Beyond Detection: While detection is its core function, YOLOv8 can also be used for image classification and instance segmentation tasks.

Anchor-Free Design: Unlike prior YOLO models, YOLOv8 takes an anchor-free approach. This means it predicts the center of an object directly instead of relying on predefined anchor boxes.

Scalable Architecture: The model's architecture is built for scalability, allowing adjustments to balance speed and accuracy based on specific needs.

In essence, YOLOv8 offers a powerful and versatile tool for various computer vision applications, particularly those requiring real-time object detection with high accuracy.

6.3 Experiment Specifications and Used Materials

The simulation experiments were performed using the following setup:

- **Platform:** Google Colab
- **Hardware:**
 - GPU: A100 with 12 GB memory
 - RAM: 16 GB
 - CPU: Intel Core i7-4610M (8.00 GHz, 1600 MHz, 4 MB L3 Cache, 2 cores, 37W)
- **Software:**
 - Programming Language: Python
 - Libraries/Frameworks: Ultralytics, OpenCV
 - Model: YOLO v8
 - Operating System: windows
- **Download a Pre-Trained Model:**
 - Pre-trained models for various tasks (object detection, classification, segmentation) are available in the Ultralytics repository ([Ultralytics models]). Download the model that suits your needs.
- **Run Inference with Python Script:**
 - The Ultralytics library provides a Python script for inference. You can find examples in the official repository ([Ultralytics examples]). These scripts allow you to: Load the model, Provide an image or video path, Run object detection and visualize the results
- **Alternatively, Use the Command-Line Tool:**

- YOLOv8 comes with a command-line tool called "yolov8". This tool allows you to perform various tasks like:
- Detection: "yolov8 task=detect"
- Setting source ("source=image.jpg") and model ("model=yolov8n.pt") paths
- Specifying confidence threshold ("conf=0.5") and other options

How to build code in YOLO V8?

- While YOLOv8 offers pre-trained models and easy-to-use scripts for inference, building the codebase from scratch isn't the recommended approach for most users. Here's why:
 1. Complexity: YOLOv8's codebase is extensive and involves deep learning concepts. Building it requires a strong understanding of PyTorch and computer vision principles.
 2. Pre-built Availability: The Ultralytics library provides well-tested and optimized code for various tasks. Rebuilding this functionality would be time-consuming and potentially error-prone.
 3. However, if you're interested in understanding the underlying architecture or contributing to the project, here's some guidance:
 - YOLOv8 Source Code: The core code for YOLOv8 is available on the Ultralytics GitHub repository: <https://github.com/ultralytics/ultralytics>. This includes the model definitions, training scripts, and inference code.
 4. Focus on Modifications: Instead of building everything, consider modifying the existing code for your specific needs. The provided scripts allow you to change the model, input data, and output configurations.

How to call libraries in yolo v8 ?

- YOLOv8 doesn't directly call external libraries in the traditional sense. Instead, it leverages functionalities from pre-installed libraries during its execution.
- Here's a breakdown of how libraries work in YOLOv8:
 1. **Dependency Management:** When you install “ultralytics” using “pip install ultralytics” , it fetches the YOLOv8 codebase and its required libraries. These libraries become part of your Python environment.
 2. **Internal Imports:** The YOLOv8 code itself uses “import” statements to access functionalities from these pre-installed libraries. Here are some commonly used libraries:
 - **PyTorch:** The core deep learning framework used for building and training the models.
 - **Torchvision:** A library within PyTorch that provides functionalities for image manipulation and pre-processing.
 - **Hydra:** A configuration management library used for managing model configurations in YOLOv8.
 - **OpenCV (Optional):** While not strictly required, OpenCV is a popular computer vision library that might be used for visualization tasks within YOLOv8 scripts (e.g., drawing bounding boxes).
 3. **Functionality Through Code:** You don't need to explicitly call these libraries in your YOLOv8 scripts. The Ultralytics library provides functions and classes that utilize the functionalities of these underlying libraries. For instance, you might use the “detect()” function from the “ultralytics” library, which internally leverages PyTorch for object detection.

Chapter 7 System Development

7.1 Overview

Our comprehensive solution is designed to assist deaf's patients and their caregivers in managing day-to-day activities. The app is built using Flutter, flask and a .net backend API for processing and managing data and offers two distinct interfaces tailored to different user roles. The first interface is designed for translate words with sign language and enables them to leverage our machine-learning model to predict words based on patient images and videos, as previously mentioned in detail. The second interface is designed for people who interested to learning a sign language and provides a range of features such as search by video of image to better manage their daily life. Our app is designed to enhance patients' quality of life while also empowering caregivers with the necessary tools to support and have trust in their loved ones. We are proud to offer our app in two languages, English and Arabic, to better serve our users and ensure they can access our app in the language of their choice. This feature is designed to make our app more accessible to users around the world and to ensure that language barriers do not prevent users from accessing critical translation resources.

7.2 App's architecture & design patterns

Our app uses a client-server architecture, with the Flutter app as the client and the Java-based backend API acting as the server. The app is built on a foundation of the Model-View-Controller (MVC) design pattern, which provides a clear separation of concerns and enables the effective organization of code. Furthermore, our app employs a RESTful API design, facilitating communication between the client and server via HTTP requests. This architecture and design pattern combination ensures that our app is scalable, maintainable, and easily extensible.

7.3 Methodological assumptions

User and system requirements to activate the system.

7.3.1 User Requirements

- Users should have basic mobile skills in android, Apple.
- Users must have a connection to access the internet.

7.3.2 System Requirements

- Android or apple mobile with internet.

7.4 Used Technologies in mobile application

7.4.1 Dart

Dart is a programming language optimized for building fast apps on any platform. It aims to be a productive language for multi-platform development, with a flexible execution runtime platform for app frameworks. Dart is versatile and well-suited for a range of development tasks, providing tools for building fast, reliable, and high-quality applications.

7.4.2 Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications from a single codebase for any web browser, Android, iOS, Linux, macOS, and Windows. First described in 2015, Flutter was released in May 2017. It provides a simple, powerful, efficient, and easy-to-understand SDK to write mobile applications in Google's own language, Dart. This tutorial walks through the basics of the Flutter framework, installation of Flutter SDK, setting up Android Studio to develop Flutter-based applications. The architecture of the Flutter framework, and the development of all types of mobile applications using the Flutter framework. It is used to develop and create the Mobile Application and show the design correctly and get it working so that we can upload that app to the stores and use it.

7.5 Technologies used in Back-end

7.5.1 Flask

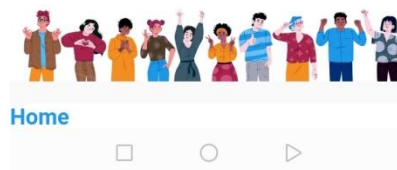
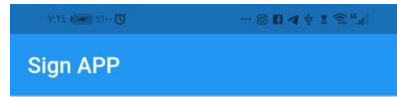
Flask is a popular Python web framework for building web applications. It is a micro-framework, which means that it is designed to be lightweight and flexible, and can be extended with a wide range of third-party libraries and tools. used for ML model Deployment.

7.5.2 ASP.NET

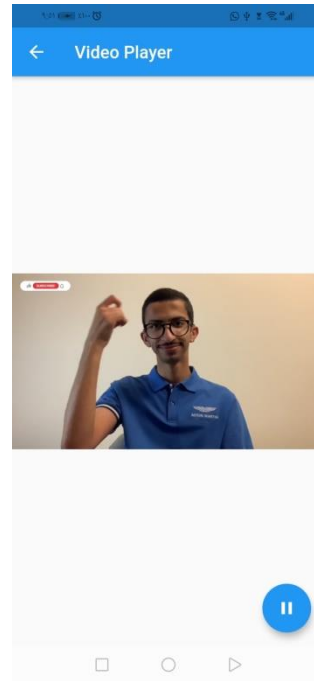
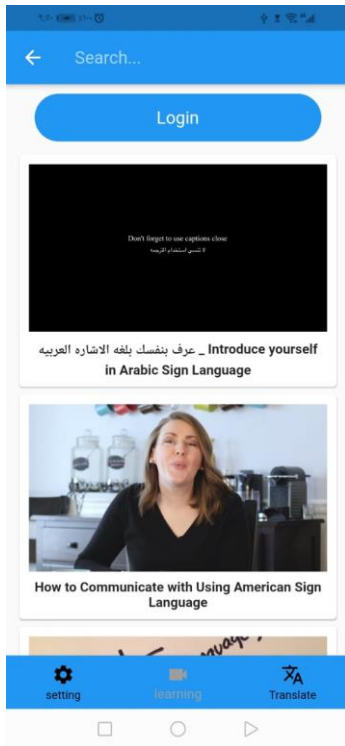
ASP.NET Core Web API can be used to build REST API services. some benefits of working with an ASP.NET Web API: It works the way HTTP works, using standard HTTP verbs like GET, POST, PUT, DELETE for all CRUD operations. Full support for routing We used ASP.NET Core Web API for Authentication and Authorization, API requires authentication and authorization, ASP.NET Core provides various authentication schemes (e.g., JWT, OAuth) and authorization policies that you can configure to secure your API endpoints.

7.6 Mobile development

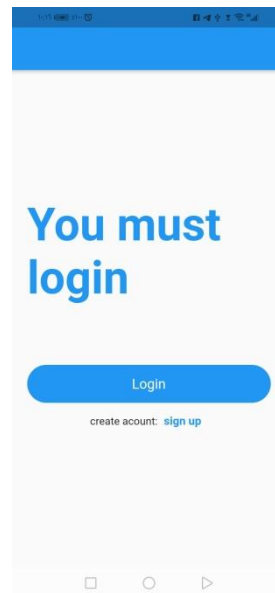
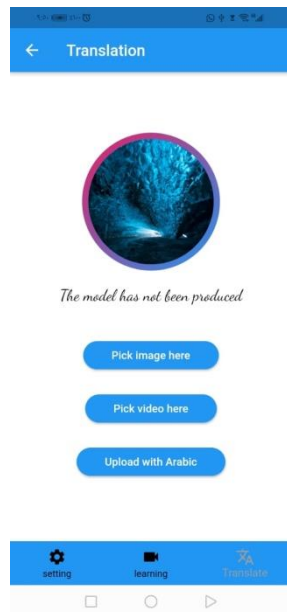
- The home of our application



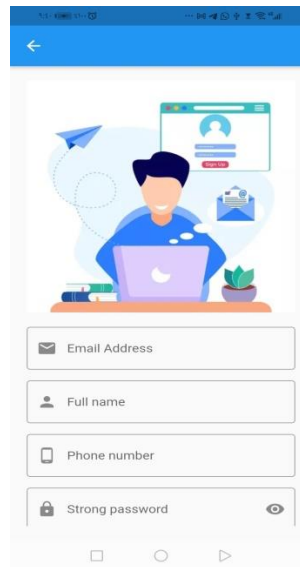
- Firstly we can browse, watch or learn sign language from freely playlists without login



- the user can translate a sign language image or video if he already log in



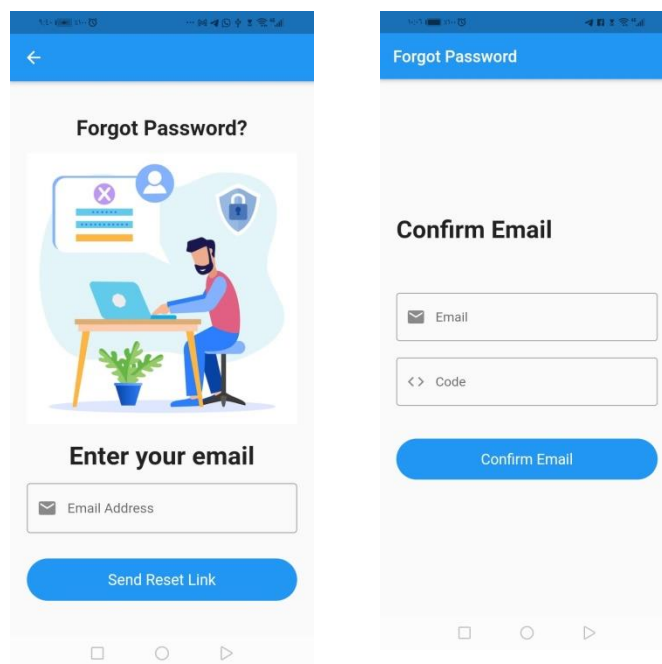
The sign up if he has not account on application yet



Registration form with the following fields:

- Email Address
- Full name
- Phone number
- Strong password (with eye icon for visibility toggle)

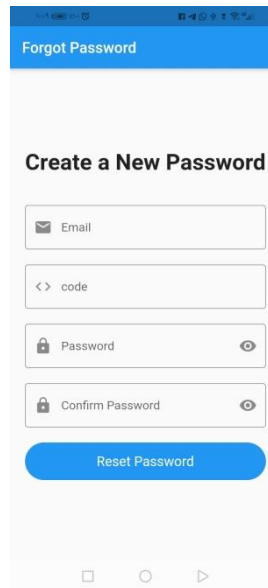
If he forgot the password so need to click forget password and recreate one



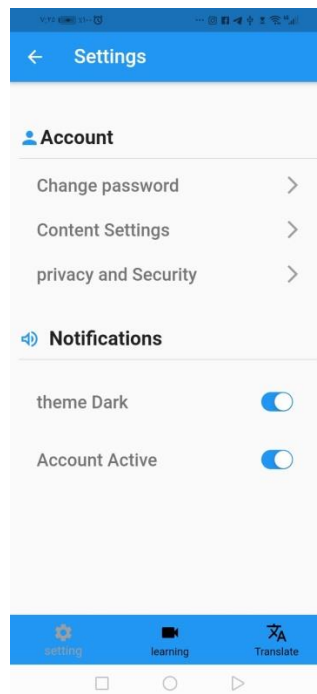
Two screens for password recovery:

- Forgot Password?** Screen:
 - Illustration of a person at a laptop with a password reset icon.
 - Text: **Enter your email**
 - Field: Email Address
 - Button: Send Reset Link
- Forgot Password** Screen:
 - Text: **Confirm Email**
 - Field: Email
 - Field: <> Code
 - Button: Confirm Email

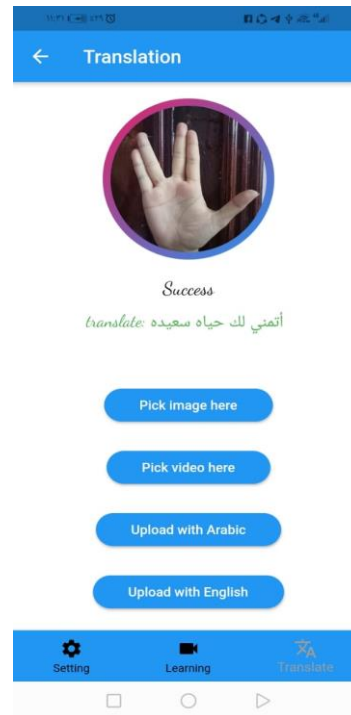
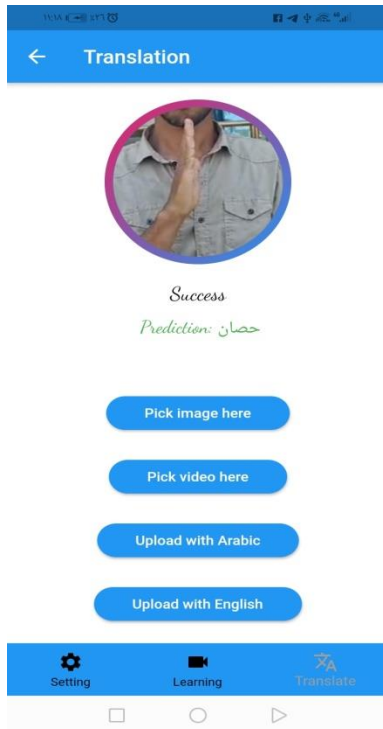
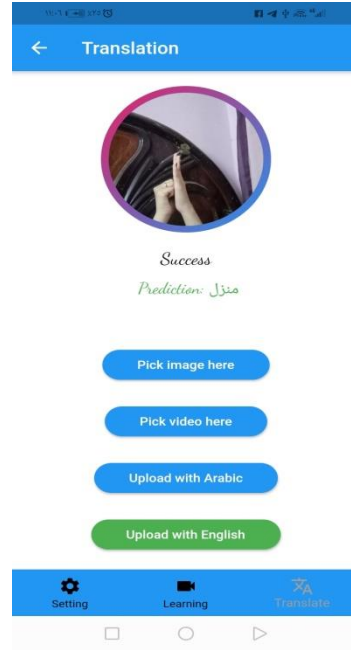
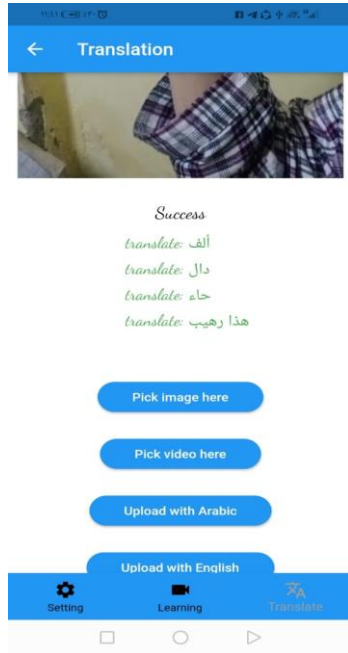
Then can create new password



The user also can change in setting on some features



There is some translation examples from our application



Chapter 8 Conclusions & Future Work

8.1 Conclusions

This sign language application has the potential to break down communication barriers between the deaf and hearing communities. By translating signs into spoken language and vice versa, the app can foster greater inclusivity and understanding. Further development focused on accuracy, fluency, and incorporating diverse sign languages will make this technology even more impactful.

This sign language learning application provides a user-friendly and accessible platform for people to acquire basic or advanced sign language skills. By making learning engaging and interactive, the app empowers users to connect with the deaf community and participate in a rich cultural experience. Continued development can involve adding features like personalized learning paths, gamification elements, and integration with video tutorials from deaf instructors.

8.2 Future Work

Along with the current features in this project, there is a high scope to extend this project so that the mobile application will be more useful for people who want to communicate with hearing-impaired people. Natural language processing (NLP) is very popular field of Artificial Intelligence in which NLP gives the ability to the machines to read, understand and derive meaning from human languages like text or speech. So, adapting this NLP technique will really improve the application as there is high scope to recognize words, sentences using NLP. In this project, the English alphabet is used to recognize the hand gestures for English letters so now there is a future scope that words and sentences will be recognized using NLP technique. The hand gestures for the letters ‘J’

and 'Z' involves hand motions so this project has to deal with image classification which involves spatial details, these two letters 'J' and 'Z' can be better recognized using deep learning techniques like Long short-term memory (LSTM) model. This LSTM model is used for the predictions based on time series data unlike standard straight forward neural networks

Chapter 8 References

8.1 References for American Sign Language Detection using YOLOv5 and YOLOv8

1. World Health Organization. (2023, February 27) Deafness and hearing loss
2. Vedak, Omkar, et al. "Sign language interpreter using image processing and machine learning." International Research Journal of Engineering and Technology (IRJET) (2019).
3. Halvardsson, G., Peterson, J., Soto-Valero, C., & Baudry, B. (2021). Interpretation of swedish sign language using convolutional neural networks and transfer learning. SN Computer Science, 2(3), 207.
4. Daniels, S., Suciati, N., & Fathichah, C. (2021, February). Indonesian sign language recognition using yolo method. In IOP Conference Series: Materials Science and Engineering (Vol. 1077, No. 1, p. 012029). IOP Publishing.
5. Bhattacharya, Abhiruchi, Vidya Zope, Kasturi Kumbhar, Padmaja Borwankar, and Ariscia Mendes. "Classification of sign language gestures using machine learning." Image 8, no. 12 (2019).
6. Starner, J. Weaver and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, pp. 1371-1375, Dec. 1998, doi: 10.1109/34.735811.
7. Aziz, M. S. B. Haji Salam, U. U. Sheikh and S. Ayub, "Exploring Deep Learning-Based Architecture, Strategies, Applications and Current Trends in Generic Object Detection: A Comprehensive Review," in IEEE Access, vol. 8, pp. 170461-170495, 2020, doi: 10.1109/ACCESS.2020.3021508.
8. Naglot and M. Kulkarni, "Real time sign language recognition using the leap motion controller," 2016 International Conference on Inventive Computation Technologies (ICICT), 2016, pp. 1-5, doi: 10.1109/INVENTIVE.2016.7830097.
9. C. -H. Chuan, E. Regina and C. Guardino, "American Sign Language Recognition Using Leap Motion Sensor," 2014 13th International Conference on

Machine Learning and Applications, 2014, pp. 541-544, doi: 10.1109/ICMLA.2014.110.

10. Imagawa, Shan Lu and S. Igi, "Color-based hands tracking system for sign language recognition," Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 462-467, doi: 10.1109/AFGR.1998.670991.

11. Wang et al., "Hear Sign Language: A Real-Time End-to-End Sign Language Recognition System," in IEEE Transactions on Mobile Computing, vol. 21, no. 7, pp. 2398-2410, 1 July 2022, doi: 10.1109/TMC.2020.3038303.

12. Fernando and J. Wijayanayaka, "Low cost approach for real time sign language recognition," 2013 IEEE 8th International Conference on Industrial and Information Systems, 2013, pp. 637-642, doi: 10.1109/ICIInfS.2013.6732059.

13. K. Dutta and S. A. S. Bellary, "Machine Learning Techniques for Indian Sign Language Recognition," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017, pp. 333-336, doi: 10.1109/CTCEEC.2017.8454988.

14. S. Teja Mangamuri, L. Jain and A. Sharmay, "Two Hand Indian Sign Language dataset for benchmarking classification models of Machine Learning," 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2019, pp. 1-5, doi: 10.1109/ICICT46931.2019.8977713.

15. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", in proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016

8.2 References for Video Captioning Based on Sign Language Using YOLOV8 Model

1. Mehta, A., Solanki, K., Trupti Rathod, T.: Automatic translate real-time voice to sign language conversion for deaf and dumb people. Int. J. Eng. Res. Technol. (IJERT) ICRADL – 2021 9(5), 174–177 (2021)

2. Rani, R.S., Rumana, R., Prema, R.: A review paper on sign language recognition for the deaf and dumb. *Int. J. Eng. Res. Technol. (IJERT)* 10(10), (2021)
3. Khallikkunaisa, Kulsoom A.A., Chandan Y.P, Fathima Farheen, F., Halima, N.: Real time sign language recognition and translation to text for vocally and hearing impaired people. *Int. J. Eng. Res. Technol. (IJERT) IETE* 8(11) (2020)
4. Kodandaram, S.R., Kumar, N., Gl, S.: Sign language recognition. *Turkish J. Comput. Math. Educ. (TURCOMAT)*. 12, 994–1009 (2021)
5. Muralidharan, N. T. , R. R. S., R. M. R., S. N. M., H. M. E.: Modelling of sign language smart glove based on bit equivalent implementation using flex sensor. In: 2022 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), Chennai, India, pp. 99–104 (2022).
6. Singh, A.K., John, B.P., Subramanian, S.R.V., Kumar, A.S., Nair, B.B.: A low-cost wearable Indian sign language interpretation system. In: 2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA), Amritapuri, India, , pp. 1–6, (2016) <https://doi.org/10.1109/RAHA.2016.7931873>