

Al-Azhar University

Faculty of Engineering for Girls

Systems and Computers Engineering Department



Intelligent Sign language Manipulation using AI.
“ISMA3NY”

A project is submitted in partial fulfilment for the requirements of the degree of Bachelor of Computer and Systems Engineering.

Presented By:

Asmaa Moayad Attia Muhammed	ID No. 20248624
Omnia Hasan Ahmed Beker	ID No. 20248636
Doaa Muhammed El-Qurany	ID No. 20248674
Salma Magdy Saeed Muhammed	ID No. 20248702
Salma Wael Abd-Elmejeed Hasan	ID No. 20248703
Somia Muhammed Amr-allah Al-sayed	ID No. 20248706

Supervised by:

Dr. Momtaz Saad Alkholy

July, 2024

1 Contents

Acknowledgment	7
Abstract	8
1 Introduction.....	10
1.1 Problem definition.....	10
1.2 Motivation.....	11
1.3 Overview.....	13
1.4 Sign Language and Hand Gesture Recognition	15
1.5 Stakeholders	16
1.6 Literature Review.....	17
1.7 Expected Project Constraints	18
2 Planning and Requirements	20
2.1 Planning.....	20
2.2 Requirements.....	22
2.2.1 Software Requirements.....	22
2.2.2 Hardware Requirements.....	22
2.2.3 Functional Requirements	23
2.2.4 Non-functional Requirements.....	24
2.2.5 User Requirements.....	25
2.3 Use Cases	26
2.3.1 User login.....	26
2.3.2 Open Camera for Detection	27
2.3.3 Sign Language Detection.....	28
2.3.4 Display Recognition Results.....	29
2.4 Domain Diagram	30
3 Project Analysis and Design	32
3.1 Use case Diagram.....	32
3.2 Sequence Diagram.....	34
3.3 Class Diagram	36
3.4 State Diagram.....	37
4 Implementation and Testing.....	40

4.1	Tools & Technologies:.....	40
4.1.1	AI	40
4.1.2	Android	41
4.2	System Overview	41
4.3	Data Selection and Preparation	41
4.3.1	Dataset Selection.....	41
4.3.2	Dataset Description.....	42
4.3.3	Data Preparation.....	43
4.4	Training Module.....	45
4.4.1	Model Construction	45
4.4.2	Model Training.....	46
4.4.3	Model Evaluation and Conversion.....	48
4.4.4	Making Predictions and Evaluating Performance.....	49
4.5	Android Application.....	50
4.6.1	Screens	50
4.5.2	Classes and functionality:	60
4.5.3	Use Firebase.....	64
4.6	Experimental Results.....	68
4.7	Sign Language Recognition Results	70
5	Conclusion	72
5.1	Conclusion.....	72
5.2	Future Work.....	73

List of figures

Figure 1-1: System Architecture	14
Figure 1-2: Arabic sign language	15
Figure 2-1: Gantt chart.....	20
Figure 2-2: Domain Object Diagram	30
Figure 3-1: Use Case Diagram.....	33
Figure 3-2: Sequence Diagram of Sign Language Recognition	34
Figure 3-4: Sequence Diagram of Login and Verification, Open Camera.....	35
Figure 3-5: Sequence Diagram Capture and Process Gestures.....	35
Figure 3-6: Sequence Diagram of Convert Text to Speech	35
Figure 3-7: Class Diagram	36
Figure 3-8: State Diagram.....	37
Figure 3-9: State diagram.....	38
Figure 3-10: State diagram.....	38
Figure 4-1: Sample of Dataset	42
Figure 5-1 :Confusion Matrix	68

List of tables

Table 1: User Login	26
Table 2: Open Camera for Detection	27
Table 3: Sign Language Detection	28
Table 4: Display Recognition Results.....	29

Abbreviations:

TTS	Text-To-Speech
HCL	Human Computer Interface
ASL	American Sign Language
BSL	British Sign Language
ArSL	Arabic Sign Language
HOG	Histogram of Oriented Gradients
SVM	Support Vector Machine
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks

Acknowledgment

Praise be to Allah, who guided us to this, and we would never have been guided if Allah had not guided us." first and before everything we would like to thank Allah for His favors and blessings on us. We wish to express our sincere appreciation to our supervisor: ***Dr.Momtaz El-kholy*** for his expert advice, his keenness on our acquisition of knowledge and keeping pace with the development of modern technology methods and his guidance throughout this project. We would also like to thank our department staff (computers and systems engineering department) for their invaluable assistance that they provided during our educational process. We wish to express our deepest gratitude and great love to our families who have kept us going on and this work would not have been possible without their support. moral support and encouragement. They have stood by us in the most difficult of times.

Abstract

The aim of this project is to develop a real - time system for converting sign language into text and speech, facilitating communication between individuals who are deaf or hard of hearing and those who are not familiar with sign language. The system leverages computer vision and natural language processing techniques to accurately recognize and interpret sign language gestures, converting them into corresponding text and generating synthesized speech output.

The system utilizes a camera - based input system, which captures video data of the user performing sign language gestures. The video frames are processed using computer vision algorithms, employing techniques such as object detection and hand pose estimation to accurately recognize hand gestures. The recognized gestures are then mapped to a predefined sign language dictionary, where each gesture corresponds to a specific word or phrase. This mapping is achieved using machine learning algorithms, which have been trained on a large dataset of sign language gesture to identify patterns and map gestures to their corresponding textual representations. Once the gestures are mapped to text, the system generates synthesized speech output using text - to - speech (TTS) technology. The synthesized speech provides an auditory representation of the sign language, enabling individuals who are not familiar with sign language to understand and respond to the communicated messages. The development of this real - time sign language conversion system has the potential to bridge the communication gap between individuals who use sign language and those who do not. By providing an efficient and accurate conversion of sign language into text and speech, this project aims to empower individuals with hearing impairments to communicate more effectively and inclusively in various social and professional settings.

Chapter 1

Introduction

1 Introduction

Speech impaired people use hand signs and gestures to communicate. Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

1.1 Problem definition

1. Background

Communication between deaf individuals who use sign language and those who rely on spoken language can be challenging. While interpreters and written communication offer solutions, they are not always convenient or accessible, especially in spontaneous or everyday interactions.

2. Problem Statement

Deaf and hard-of-hearing individuals often face significant communication barriers in various social, educational, and professional settings. These barriers can lead to misunderstandings, exclusion, and a lack of equal opportunities. Traditional methods of bridging this communication gap, such as interpreters or written notes, are not always practical or efficient for real-time, dynamic conversations.

3. Key Challenges

- Real-Time Translation: Existing solutions often fail to provide accurate and immediate translation of sign language to speech, limiting fluid conversation.
- Accessibility and Convenience: Many tools are either too complex to use or not readily available to the general public, restricting their widespread adoption.
- Language Variability: Sign languages are not universal; different regions use different sign languages, necessitating a solution that can handle multiple languages and dialects.
- User Personalization: Users have diverse needs and preferences, requiring customizable features to cater to individual requirements.

- Technological Limitations: Ensuring high accuracy and low latency in translation requires advanced technology that can work reliably in varied lighting and background conditions.

4. Solution Vision

Develop a Sign Language to Speech app that leverages cutting-edge computer vision and machine learning technologies to provide real-time, accurate translation of sign language into spoken words. The app should be user-friendly, highly accessible, and customizable, supporting multiple sign languages and offering voice personalization options.

5. Objectives

- Accuracy: Achieve high accuracy in recognizing and translating sign language gestures.
- Speed: Ensure real-time translation with minimal delay to enable natural conversations.
- User Experience: Design an intuitive and accessible user interface.
- Support for Multiple Languages: Include support for various sign languages and dialects.
- Offline Functionality: Provide offline capabilities for use without an internet connection.

1.2 Motivation

1. Bridging Communication Gaps: Deaf and hard-of-hearing individuals face significant communication barriers in daily interactions with those who rely on spoken language. This app aims to bridge this gap, fostering more inclusive and seamless communication.
2. Promoting Inclusivity: Inclusion in social, educational, and professional settings is often hindered by communication difficulties. By translating sign language into speech in real time, the app helps create a more inclusive environment where everyone can participate equally.
3. Enhancing Independence: Many deaf individuals rely on interpreters or written communication, which can limit spontaneity and independence. The app empowers users to communicate directly and autonomously, enhancing their confidence and independence.

4. Supporting Accessibility: Accessibility is a fundamental right. This app aligns with efforts to make communication accessible to all, ensuring that deaf and hard-of-hearing individuals can engage fully in various aspects of life without barriers.

5. Leveraging Technological Advances: Advancements in computer vision, machine learning, and natural language processing have made it feasible to create sophisticated tools for sign language translation. This app leverages these technologies to provide an accurate, reliable, and user-friendly solution.

6. Real-World Applications: The app addresses real-world needs in multiple scenarios:

- Social Interactions: Facilitates easier communication in social settings, reducing feelings of isolation.
- Educational Settings: Supports students in classrooms by translating their contributions into spoken language, aiding their learning experience.
- Work Environments: Enhances workplace communication, enabling full participation of deaf employees in meetings and discussions.
- Public Services: Improves access to essential services, such as healthcare and customer support, where effective communication is crucial.

7. Raising Awareness: By making interactions with sign language users more commonplace, the app can help raise awareness and understanding of the deaf community and sign language among the hearing population.

8. Addressing Technological Gaps: Existing solutions for sign language translation often lack real-time capabilities or are not user-friendly. This app addresses these gaps by providing a practical, efficient, and accessible solution.

9. Personalization and Adaptability: The app's ability to learn and adapt to individual signing styles over time ensures a personalized and improved user experience, increasing its usefulness and acceptance.

10. Driving Social Change: By making communication easier and more natural, the app contributes to breaking down social barriers, fostering better relationships, and promoting equality and respect for the deaf and hard-of-hearing community.

1.3 Overview

The primary goal of the app is to enable effortless and dynamic conversations by converting sign language gestures into audible speech, making it easier for deaf and hard-of-hearing individuals to interact in various social, educational, and professional contexts.

- Core Features:
 - Real-Time Translation: Uses advanced computer vision and machine learning algorithms to instantly translate sign language gestures into spoken words.
 - User-Friendly Interface: Designed with simplicity in mind, ensuring ease of use for people of all ages and technical abilities.
 - Learning and Adaptation: The app can learn and adapt to individual users' signing styles over time, improving accuracy.
- Technological Components:
 - Computer Vision: Detects and tracks hand and body movements to interpret sign language gestures.
 - Machine Learning: Utilizes neural networks trained on vast datasets of sign language to accurately translate gestures into spoken language.
- User Scenarios:
 - Everyday Conversations: Facilitates real-time communication in casual, daily interactions.
 - Educational Settings: Assists in classrooms by translating students' sign language questions or comments into speech.
 - Professional Environments: Enhances workplace communication, ensuring deaf and hard-of-hearing employees can fully participate in meetings and discussions.
 - Public Services: Improves accessibility in public services like healthcare, banking, and customer support.

- Benefits:
 - Inclusivity: Promotes greater inclusion of deaf and hard-of-hearing individuals in various aspects of life.
 - Convenience: Provides a practical and accessible solution for real-time communication.
 - Empowerment: Enhances the independence and confidence of users by enabling more direct and personal interactions.
 - Awareness: Raises awareness and understanding of sign language among the hearing population.

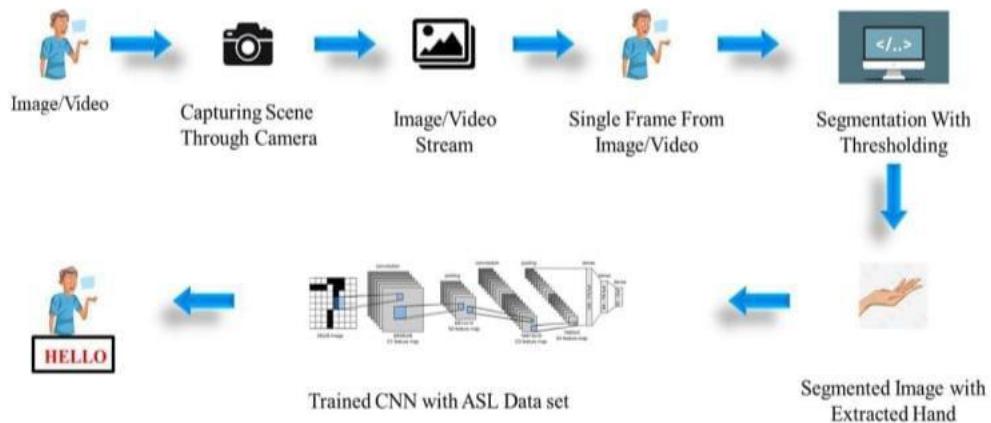


Figure 1-1: System Architecture

1.4 Sign Language and Hand Gesture Recognition

Sign language recognition converts users' signs and gestures into text, bridging communication gaps for the deaf and mute community. It uses image processing and neural networks to map gestures to text, enabling interaction with the general public. This technology addresses the communication challenges faced by deaf and mute individuals, whose interactions are often limited to family or community due to a lack of widespread sign language knowledge. Despite advancements in speech recognition, there are no significant commercial products for sign recognition yet. The project aims to enhance human-computer interaction (HCI) by developing systems that understand gestures using image processing and template matching techniques.

الأبجدية العربية بلغة الإشارة



Figure 1-2: Arabic sign language

1.5 Stakeholders

1. Deaf and Hard of Hearing Community

- Individuals: People who are deaf or hard of hearing who can benefit directly from the app for communication.
- Families and Friends: Relatives and friends who want to communicate more effectively with their deaf or hard of hearing loved ones.

2. Sign Language Interpreters

- Professionals who may use the app to enhance their interpreting services or to aid in situations where an interpreter is not available.

3. Educational Institutions

- Schools for the Deaf: Institutions that cater specifically to deaf students might integrate the app into their curriculum.
- Universities and Colleges: Higher education institutions offering courses in sign language or special education might find the app useful.

4. Healthcare Providers

- Hospitals and Clinics: To facilitate communication between healthcare providers and patients who use sign language.
- Mental Health Professionals: Therapists and counselors who work with deaf and hard of hearing individuals.

5. Government and Non-Governmental Organizations

- Social Services: Government agencies that provide support services to the deaf and hard of hearing community.

1.6 Literature Review

Research on sign language recognition has seen significant advancements over the past few decades, with a focus on various sign languages including American Sign Language (ASL), British Sign Language (BSL), and others. However, Arabic Sign Language (ArSL) recognition has gained attention more recently. This section reviews some of the notable works in the field of ArSL recognition.

Image-Based Recognition

One of the early approaches to ArSL recognition involves the use of image processing techniques to interpret static hand gestures. Al-Rousan et al. (2018) proposed a system that utilizes Histogram of Oriented Gradients (HOG) features and Support Vector Machines (SVM) for recognizing static Arabic alphabet signs. Their system achieved a commendable accuracy, highlighting the potential of image-based methods for ArSL recognition.

Deep Learning Approaches

With the advent of deep learning, Convolutional Neural Networks (CNNs) have become the cornerstone of modern image recognition systems. El-Sawy et al. (2020) developed a deep CNN model that was trained on a large dataset of ArSL gestures. Their approach significantly outperformed traditional image processing techniques, demonstrating the effectiveness of deep learning in capturing the intricate details of hand gestures.

Another notable contribution is by Hussein et al. (2021), who utilized a combination of CNNs and Recurrent Neural Networks (RNNs) to recognize dynamic sign sequences in ArSL. By leveraging the temporal dependencies between consecutive frames, their model achieved higher accuracy in recognizing continuous sign language sentences.

Sensor-Based Recognition

Beyond image and video data, sensor-based approaches have also been explored for ArSL recognition. Abu-Zaiter et al. (2019) employed data from wearable sensors to capture the motion and orientation of the hand. Their system used a combination of accelerometers and gyroscopes to track hand movements and applied machine learning algorithms for gesture

classification. This method proved to be effective in noisy environments where image-based systems might struggle.

Hybrid Systems

Recent studies have explored hybrid approaches that combine image data with sensor data to improve recognition accuracy and robustness. Al-Jarrah et al. (2022) proposed a hybrid system that integrates CNN-based image recognition with sensor data from gloves equipped with flex sensors. This multi-modal approach leveraged the strengths of both data types, resulting in improved performance in various lighting conditions and complex backgrounds.

1.7 Expected Project Constraints

The following are the expected project constraints for the system:

1. Time constraints: The project may be subject to time constraints, such as deadlines for implementation, testing, and deployment.
2. Financial constraints: The project may be subject to financial constraints, such as budget limitations or the need to generate revenue to cover implementation and maintenance costs.
3. Technical constraints: The project may be subject to technical constraints, such as limitations in the availability of hardware, software, or communication networks.

Chapter 2

Planning and Requirements

2 Planning and Requirements

In this chapter, we will talk about the activities definition, sequencing, and duration estimating using a Gantt chart. We will also cover the software and hardware requirements, as well as the functional and non-functional requirements essential for the development and deployment of the Arabic Sign Language (ArSL) Recognition App.

2.1 Planning

(a) Activities Definition, Sequencing and Duration Estimating

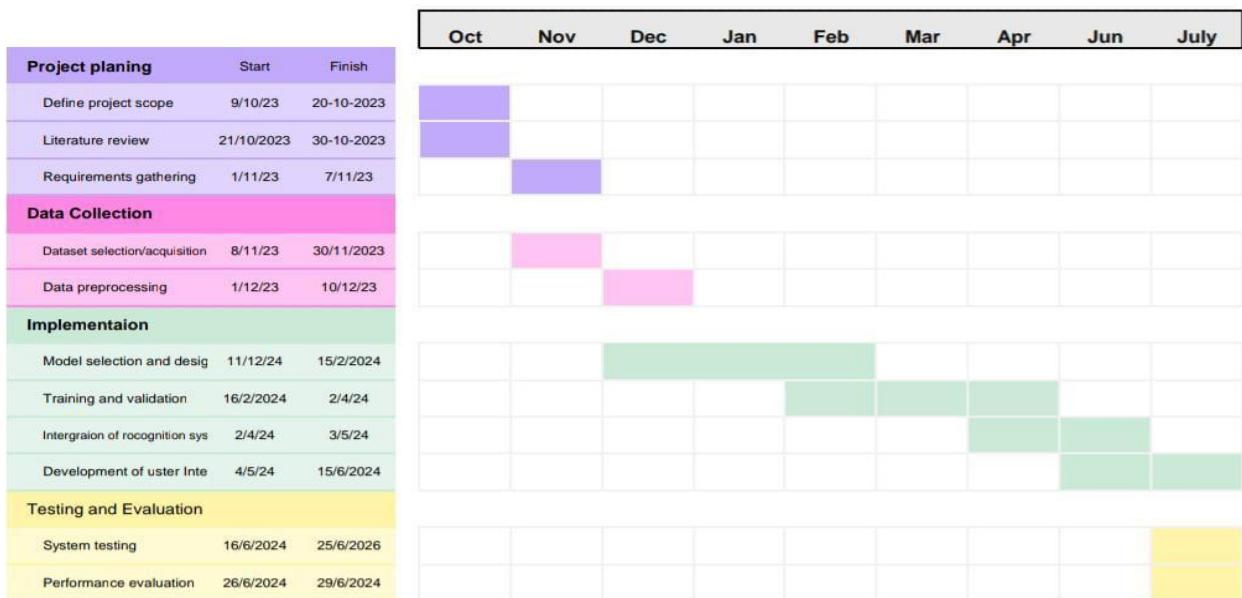


Figure 2-1: Gantt chart

(b) Risk list

- **Technical Risks:**
 - Performance Issues: The app may experience lag or delays, especially on lower-end devices
 - Model Accuracy: The sign language recognition model may not perform accurately across different users and environments.
 - Integration Challenges: Issues might arise when integrating the Python model into the Android app.
- **Data Risks:**
 - Insufficient Data: Lack of sufficient and diverse training data can affect model performance.
 - Data Privacy: Storing and managing user data improperly can lead to privacy violations.
- **Project Management Risks:**
 - Scope Creep: Uncontrolled changes or additions to project scope can lead to delays and budget overruns.
 - Resource Constraints: Limited availability of resources (e.g., developers, hardware) can affect project timelines.
- **External Risks:**
 - Market Competition: Similar apps entering the market can affect the app's success.

2.2 Requirements

2.2.1 Software Requirements

Development Tools

- Integrated Development Environment (IDE) such as VSCode for Python development.
- Android Studio for Android app development.

Programming Languages

- Python for developing the sign language recognition model.
- Java for Android app development.

Frameworks and Libraries

- Machine learning frameworks such as TensorFlow or PyTorch for developing the recognition model.
- Computer vision libraries like OpenCV for video processing.
- Android libraries and SDKs for integrating the model into the app.

2.2.2 Hardware Requirements

Device

- Smartphones or tablets with a built-in camera for capturing video input.
- Sufficient processing power and memory to handle video processing and recognition tasks.

Development Machine

A computer with sufficient resources to run VSCode, Android Studio, and necessary libraries for model development and app deployment.

2.2.3 Functional Requirements

Sign Language Interpretation

- The system shall accurately interpret sign language gestures captured from real time video input.
- It shall support multiple sign language dialects and variations.
- The interpretation process shall be performed in real-time with minimal latency.

Translation:

- 4. The system shall translate interpreted sign language gestures into textual format.
- It shall provide multilingual support for translated text output.
- The translation accuracy shall meet or exceed predefined thresholds.

Speech Synthesis

- The system shall synthesize translated text into audible speech output,
- It shall support multiple languages and accents for speech synthesis.
- Speech synthesis shall be natural-sounding and intelligible.

User Interaction

- 10. The system shall provide an intuitive user interface for initiating and controlling translation processes.
- 11. It shall support touch-based interactions for inputting sign language gestures and navigating the interface.
- 12. The user interface shall be accessible to individuals with disabilities, including support for screen readers and alternative input methods.

2.2.4 Non-functional Requirements

Performance

- The system shall process sign language gestures and deliver translated output in real-time with a maximum latency of X milliseconds.
- It shall be capable of handling concurrent user requests and scaling to accommodate increasing loads.

Security

- The system shall encrypt all communication channels to ensure data confidentiality and integrity.
- It shall implement user authentication mechanisms to prevent unauthorized access to user data.

Accessibility

- The system shall comply with accessibility standards (e.g., WCAG) to ensure usability for individuals with disabilities.
- It shall provide adjustable settings for font size, color contrast, and other accessibility features.

Compatibility

- The system shall be compatible with Android mobile devices running Android OS version X and above.
- It shall support integration with third-party services and platforms for enhanced functionality.

Reliability

- 9. The system shall have a minimum uptime of 99% to ensure continuous availability.
- 10. It shall implement error handling mechanisms to gracefully handle system failures and recover from errors.

2.2.5 User Requirements

User Feedback

- The system shall provide users with the ability to provide feedback on translation accuracy and usability.
- It shall incorporate user feedback to continuously improve translation quality and user experience.

User Training and Support

- The system shall offer user training resources, including tutorials and documentation, to facilitate user onboarding.
- It shall provide responsive customer support channels for addressing user inquiries and technical issues.

2.3 Use Cases

2.3.1 User login

Table 1: User Login

Use Case Name	User Login
Actor	User (anyone who wants to use the ArSL recognition app)
Brief Description	The user logs into the ArSL recognition app using their credentials to access the app's features.
Basic Flow	<ol style="list-style-type: none">1) The user opens the app and is presented with the login page.2) The user enters their username and password.3) The user clicks the "Login" button.4) The system verifies the credentials.5) If the credentials are correct, the user is redirected to the main page.
Precondition	<ol style="list-style-type: none">1) The user has a registered account with the app.2) The app is installed and running on the user's device.
Postcondition	The user is successfully logged in and redirected to the main page of the app.

2.3.2 Open Camera for Detection

Table 2: Open Camera for Detection

Use Case Name	Open Camera for Detection
Actor	User (logged-in user of the ArSL recognition app)
Brief Description	The user navigates to the main page and presses a button to open the camera and start the sign language detection process.
Basic Flow	<ol style="list-style-type: none"> 1) The user is on the main page of the app. 2) The user presses the "Open Camera" button. 3) The app requests permission to access the camera (if not previously granted). 4) The user grants camera access permission. 5) The camera is activated, and the app is ready for detection.
Precondition	<ol style="list-style-type: none"> 1) The user is logged into the app and is on the main page. 2) The device has a working camera.
Postcondition	The camera is activated, and the app is ready to detect and recognize Arabic Sign Language gestures.

2.3.3 Sign Language Detection

Table 3: Sign Language Detection

Use Case Name	Sign Language Detection
Actor	User
Brief Description	The user performs sign language gestures in front of the camera, and the app detects and recognizes these gestures in real-time.
Basic Flow	<ol style="list-style-type: none"> 1) The camera is activated, and the app is in detection mode. 2) The user performs ArSL gestures in front of the camera. 3) The app captures the gestures using the camera. 4) The app processes the captured gestures to recognize the sign language. 5) The app displays the recognized letter corresponding to the gestures in real-time.
Precondition	<ol style="list-style-type: none"> 1) The camera is activated, and the app is in detection mode. 2) The user is within the camera's view and performs clear gestures.
Postcondition	The app successfully recognizes the gestures and displays the corresponding text.

2.3.4 Display Recognition Results

Table 4: Display Recognition Results

Use Case Name	Display Recognition Results
Actor	User
Brief Description	The app displays the recognized sign language gestures as text or speech output to facilitate communication.
Basic Flow	<ol style="list-style-type: none"> 1) The app has successfully detected and recognized sign language gestures. 2) The app converts the recognized gestures into text. 3) The app displays the text on the screen. 4) The app converts the text to speech and plays the audio.
Precondition	The app has successfully detected and recognized sign language gestures.
Postcondition	The recognized text or speech output is displayed to the user, allowing them to communicate effectively.

2.4 Domain Diagram

A domain object diagram, also known as a class diagram in UML, visually represents the classes and their relationships within a system.

For Our project, the key classes and relationships include:

- 1) **User**: Represents the users of the app, with attributes for identification and methods for logging in and out.
- 2) **Login Page**: Handles user authentication, displaying the login interface and validating user credentials.
- 3) **Main Page**: The main interface where the user can access the camera and other features after logging in.
- 4) **Camera**: Manages the device's camera, capturing gestures for recognition.
- 5) **Gesture Recognition**: Processes the images captured by the camera to recognize Arabic Sign Language gestures.
- 6) **Result Display**: Displays the results of the gesture recognition, either as text or spoken output.

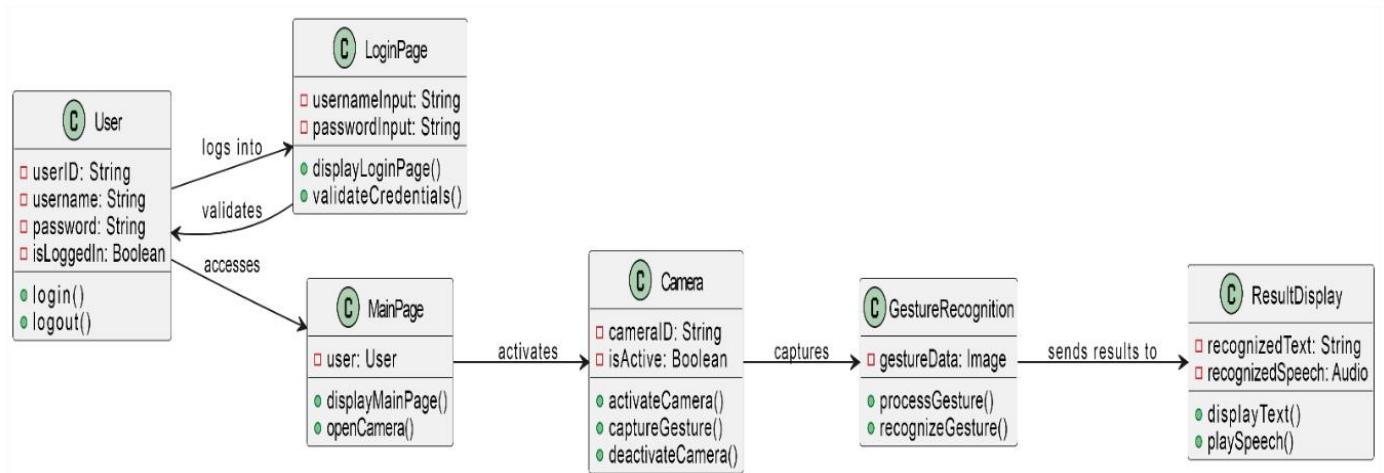


Figure 2-2: Domain Object Diagram

Chapter 3

Project Analysis and Design

3 Project Analysis and Design

3.1 Use case Diagram.

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment.

The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the behavior of the system as seen from the actor 's point of view. It describes the function provided by the system as a set of events that yield a visible result for the actor.

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

we have to use the following guidelines to draw an efficient use case diagram The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed. Give a suitable name for actors. Show relationships and dependencies clearly in the diagram. Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements. Use notes whenever required to clarify some important points.

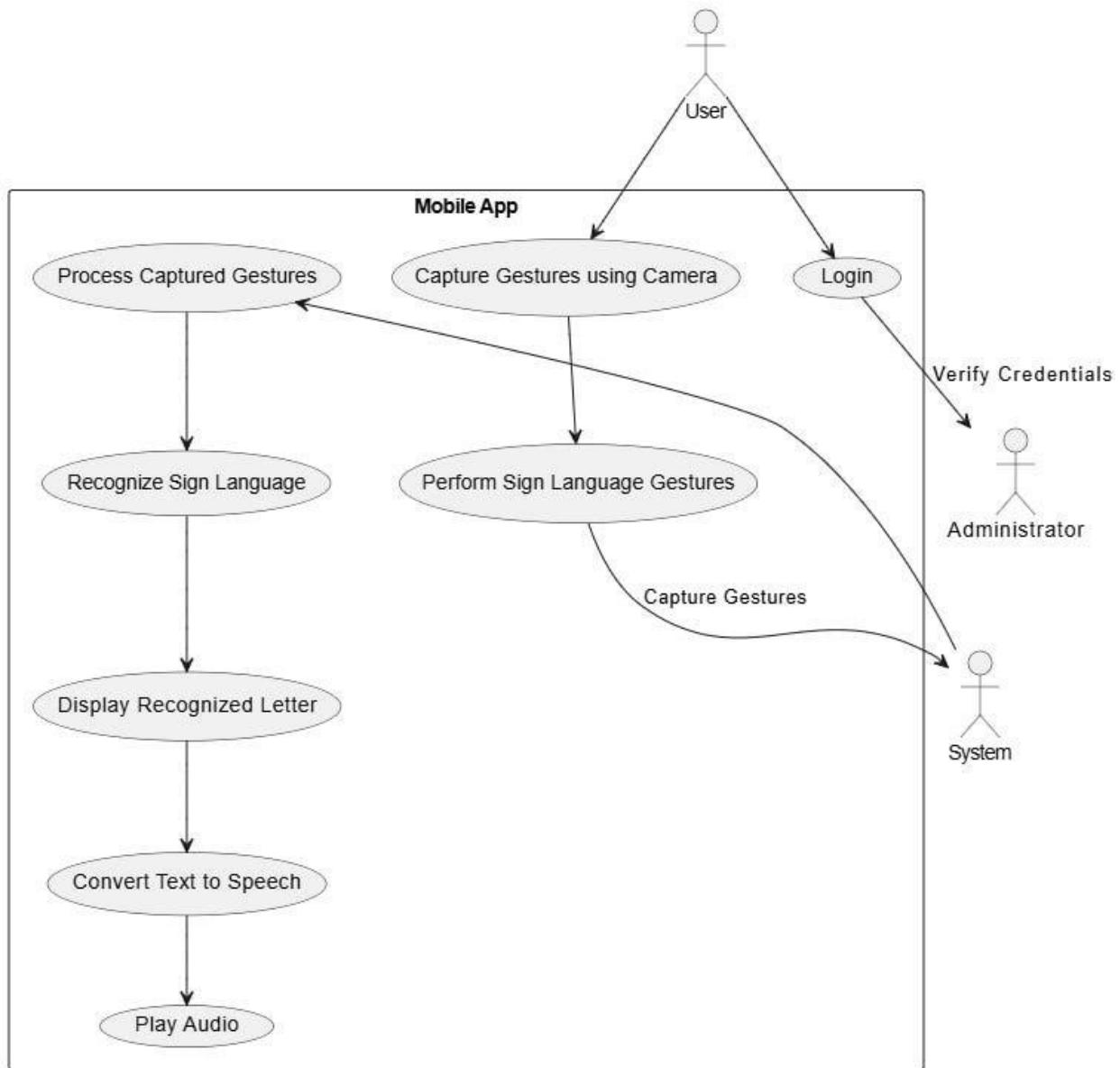


Figure 3-1: Use Case Diagram

3.2 Sequence Diagram

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension(time) and horizontal dimension (different objects).

Objects: Object can be viewed as an entity at a particular point in time with specific value and as a holder of identity. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

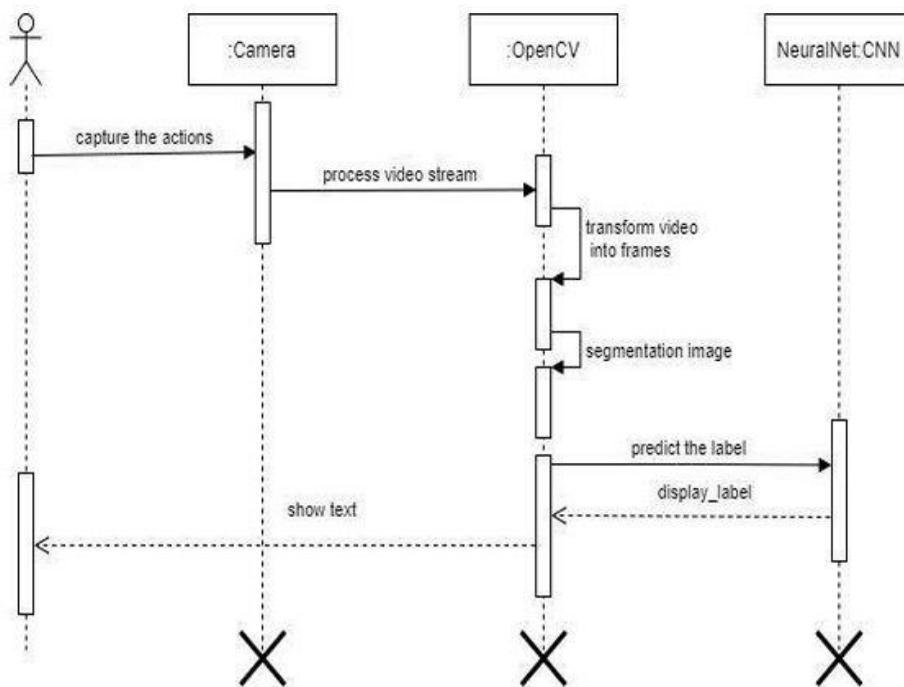


Figure 3-2: Sequence Diagram of Sign Language Recognition

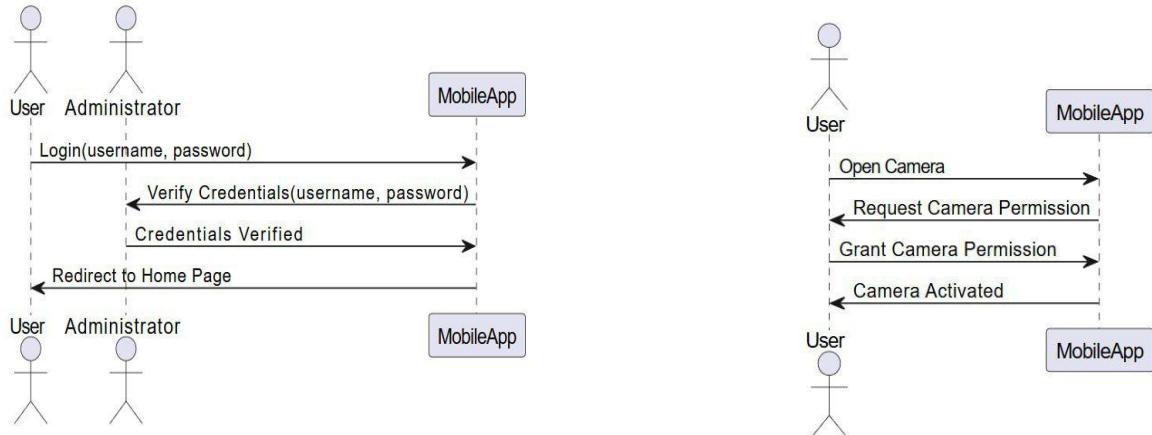


Figure 3-3: Sequence Diagram of Login and Verification, Open Camera

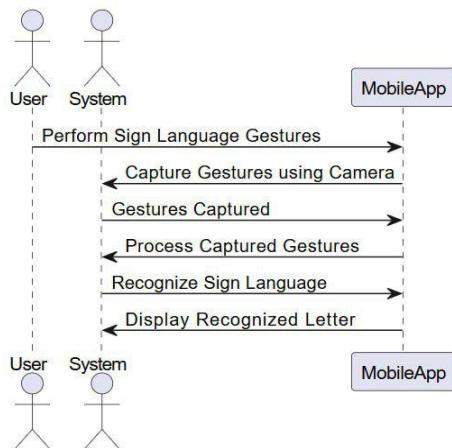


Figure 3-4: Sequence Diagram Capture and Process Gestures

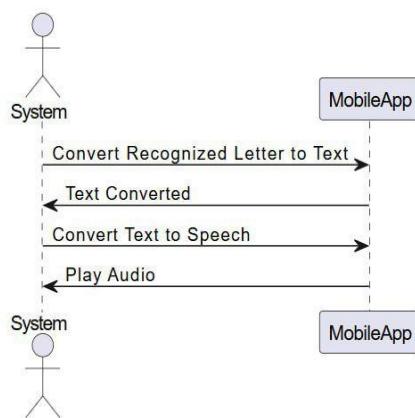


Figure 3-5: Sequence Diagram of Convert Text to Speech

3.3 Class Diagram

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe the different perspective when designing a system conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations.

Class diagram also display relationships such as containment, inheritance, association etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes.

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram.

Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top-level view.

The following points should be remembered while drawing a class diagram – The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance. Responsibility (attributes and methods) of each class should be clearly identified for each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram. At the end of the drawing, it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn.

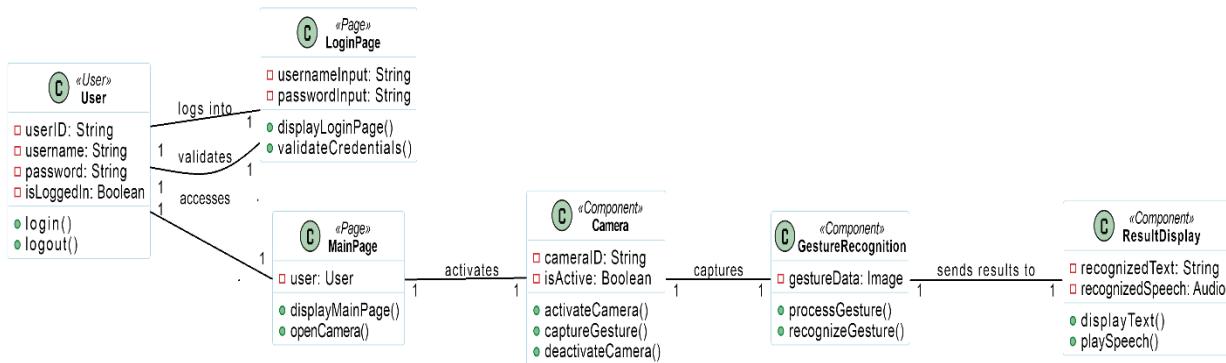


Figure 3-6: Class Diagram

3.4 State Diagram

A state chart diagram describes a state machine which shows the behavior of classes. It shows the actual changes in state not processes or commands that create those changes and is the dynamic behavior of objects over time by modelling the life cycle of objects of each class. It describes how an object is changing from one state to another state.

There are mainly two states in State Chart Diagram: 1. Initial State 2. Final-State. Some of the components of State Chart Diagram are:

State: It is a condition or situation in life cycle of an object during which it's satisfies same condition or performs some activity or waits for some event.

Transition: It is a relationship between two states indicating that object in first state performs some actions and enters into the next state or event.

Event: An event is specification of significant occurrence that has a location in time and space.

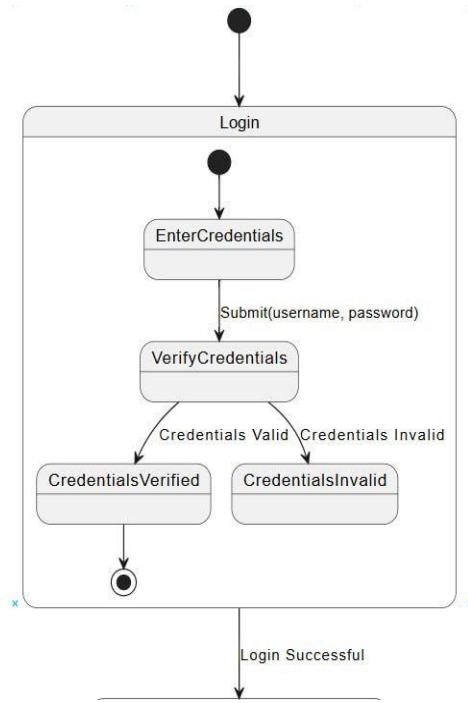


Figure 3-7: State Diagram

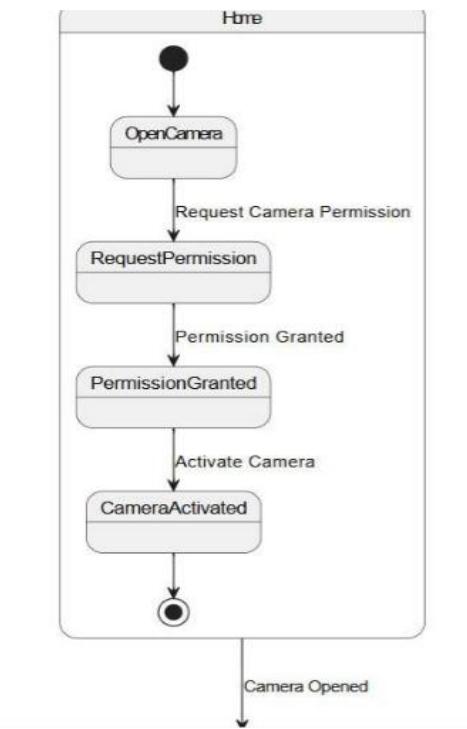


Figure 3-8: State diagram

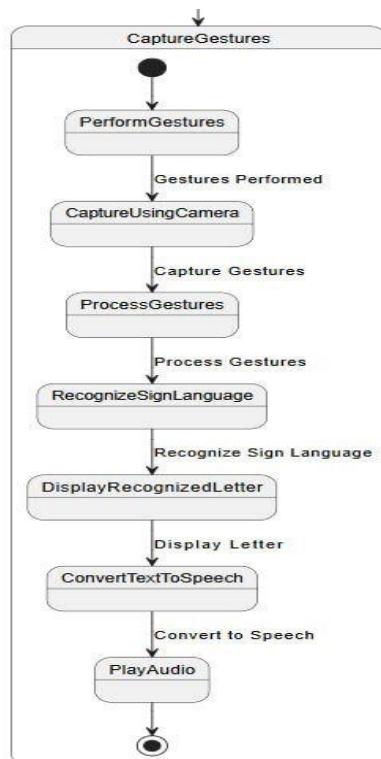


Figure 3-9: State diagram

Chapter 4

Implementation and Testing

4 Implementation and Testing

The purpose of this part is to document the implementation of a sign language recognition system using a Convolutional Neural Network (CNN) model and its deployment on an Android platform. The system aims to facilitate communication for individuals who use Arabic sign language.

4.1 Tools & Technologies:

4.1.1 AI

1. Programming Languages:

- Python: Used for developing and training the CNN model using libraries like TensorFlow or Keras.

2. Machine Learning Libraries:

- TensorFlow: Popular open-source library for developing and training deep learning models.
- Keras: High-level neural networks API, often used with TensorFlow for building and training models.
- scikit-learn: Useful for data preprocessing, splitting datasets, and evaluating models.

3. Deep Learning Models:

- Convolutional Neural Network (CNN): Specifically used for image classification tasks like sign language recognition.
- Pre-trained Models (e.g., EfficientNet): Leveraged for transfer learning to boost model performance with limited data.

4. Data Handling and Augmentation:

- NumPy: Fundamental package for numerical computing in Python, used for handling large arrays of data.

- PIL (Python Imaging Library) / Pillow: Libraries for opening, manipulating, and saving many different image file formats.
- Augmentor: Python package for augmenting image data to increase dataset size and model robustness.

4.1.2 Android

1. Programming Languages:

- Java: Used for developing the Android application to deploy the trained model.

2. Deployment Tools:

- TensorFlow Lite: Lightweight version of TensorFlow designed for mobile and embedded devices, used to deploy machine learning models on Android.
- Android SDK: Software Development Kit providing necessary tools and APIs for Android app development and deployment.

4.2 System Overview

This section provides a high-level overview of the system architecture, including the main components and their interactions.

- **Components**

- Dataset Selection and Preparation
- Model Training and Conversion to TensorFlow Lite
- Android Application for Real-Time Recognition

4.3 Data Selection and Preparation

4.3.1 Dataset Selection

The dataset was obtained from an existing source rather than being collected manually. It comprises images representing 29 different Arabic letters, with each letter having 201 images in the original dataset. Following data augmentation, each class now contains 2005 images.

4.3.2 Dataset Description

The dataset consists of images for 29 different Arabic sign language gestures:

- Classes: 29 Arabic letters.
- Number of Images:
 - Original: 201 images per class, resulting in a total of 5,628 images.
 - Augmented: 2005 images per class, resulting in a total of 56,140 images.
- Image Resolution: 96x96 pixels.
- Image Format: RGB color images saved in JPEG, PNG and JPG format.
- Augmentation Techniques: Rotation, zoom, width and height shifts, shear transformations, and horizontal flips were applied to enhance the dataset and improve model generalization.

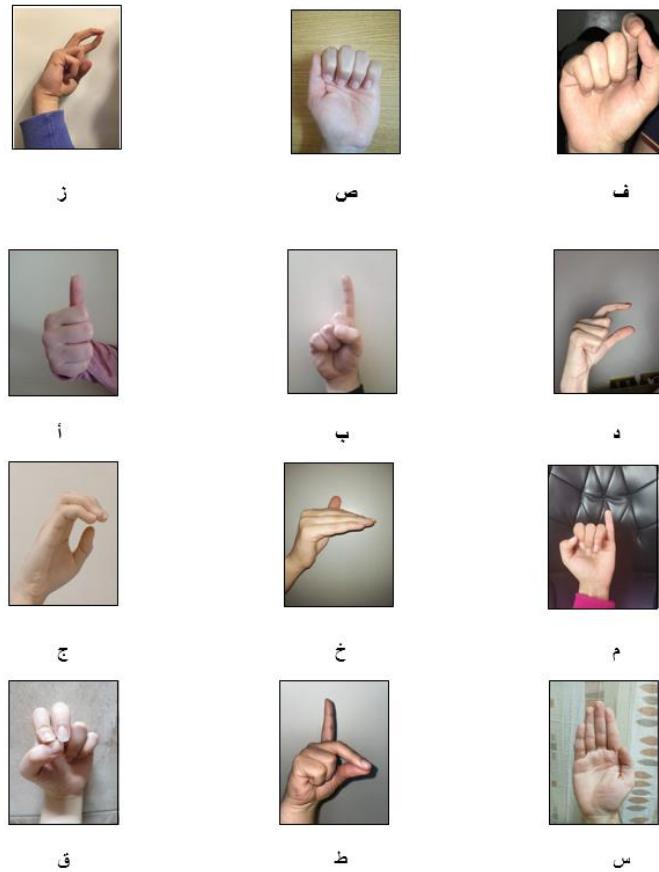


Figure 4-1: Sample of Dataset

4.3.3 Data Preparation

The preparation of the dataset involves several steps to ensure the data is in a suitable format for training the machine learning model. The steps include loading the images, resizing them, converting color formats, normalizing, encoding labels, and splitting the data into training and testing sets.

- **Data Loading and Transformation**

- Library Imports: Import the necessary libraries for data manipulation and image processing.
- Dataset Path: Define the path to the dataset and list all files. List all the files in the dataset directory and sort them alphabetically for consistency.
- Initialize arrays: load and process Images: Create empty lists to store image data and labels and Loop through each file, read images, resize to 96x96 pixels, convert to RGB, and append to lists.
- Convert to Numpy Arrays: Convert image and label lists to numpy arrays.

Code for Data Loading and Transformation

```
import numpy as np
import pandas as pd
import tensorflow as tf
import os
import cv2
import matplotlib.pyplot as plt
from tqdm import tqdm

# now define path to dataset
path=r"F:\0-current_project\DATA"
files=os.listdir(path)
# list of files in path
# sort path from A-Y
files.sort () # not nessesaray
# print to see list
print(files)

# create list of image and label
image_array=[]
label_array=[]
# loop through each file in files
for i in tqdm(range(len(files))):
    sub_file=os.listdir(path+"/"+files[i])
    for j in range (len(sub_file)):
        file_path=path+"/"+files[i]+"/"+sub_file[j]
        image=cv2.imread(file_path)
        image=cv2.resize(image,(96,96))
        image=cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        image_array.append(image)
        label_array.append(i)
```

- **Data Preprocessing**

- **Normalization:** Normalize pixel values to the range [0, 1] by dividing by 255.
- **Resizing:** Resize all images to 96x96 pixels.
- **Augmentation:** Apply augmentation techniques such as rotation, zoom, width and height shifts, shear transformations, and horizontal flips to increase the dataset size and improve model robustness.
- **Data Splitting:** Split the dataset into training and testing sets.

Code for Data Augmentation

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
    rotation_range=20,          #Random rotation between 0 and 45
    width_shift_range=0.2,       %# shift
    height_shift_range=0.2,
    shear_range=0.05,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='reflect', cval=125)  #Also try nearest, constant, reflect, wrap

import numpy as np
import os
import cv2
from PIL import Image

data=[]
lable=[]

categ =["Hah"] # the class name
data_dir=r"F:\0-current_project\DATA"  #path to the class

for i in categ:
    path = os.path.join(data_dir,i)
    for img in os.listdir(path):
        image_path = os.path.join(path,img)
        image = cv2.imread(image_path)
        if image is None:
            print("Error: Image not loaded. Check the path.")
        rgb_img= cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        rgb_img = cv2.resize(rgb_img,(200,200))
        data.append(rgb_img)
        lable.append(i)

i = 0
for batch in datagen.flow(x, batch_size=202,
                           save_to_dir='test', #path to store the generated data
                           save_prefix='aug',
                           save_format='png'):

    i += 1
    print(i)
    if i > 4:
        break
```

4.4 Training Module

Supervised machine learning: It is one of the ways of machine learning where the model is trained by input data and expected output data, to create such model, it is necessary to go through the following phases:

1. Model Construction

2. Model Training

3. Model Testing

4. Model Evaluation

4.4.1 Model Construction

In this project, the model construction relies on neural networks, specifically using transfer learning with a pre-trained model. The process involves several steps:

1. Initialize the Model:

- Begin by creating a Sequential model object.

2. Add Layers:

- Incorporate a pre-trained model (EfficientNetB0) without the top layer, which excludes the original classification layer.
- Add a global average pooling layer to reduce the spatial dimensions of the output from the pre-trained model.
- Include a dropout layer to prevent overfitting by randomly setting a fraction of input units to 0 at each update during training.
- Add a dense layer with 28 units (corresponding to the 28 classes) and a softmax activation function for classification.

3. Build the Model:

- Define the input shape for the model.

4. Summarize the Model:

- Print a summary of the model architecture to review the layers and their parameters.

Code for Model Construction

```
model = Sequential()
pretrained_model=tf.keras.applications.EfficientNetB0(input_shape=(96,96,3),include_top=False)
model.add(pretrained_model)

model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dropout(0.3))

model.add(layers.Dense(28, activation='softmax'))
# model.add(layers.Dense(1))
model.build(input_shape=(None,96,96,3))
model.summary()
```

```
Model: "sequential"
-----  
Layer (type)          Output Shape         Param #  
=====  
efficientnetb0 (Functional (None, 3, 3, 1280)      4049571  
)  
  
global_average_pooling2d ( (None, 1280)           0  
GlobalAveragePooling2D)  
  
dropout (Dropout)        (None, 1280)           0  
  
dense (Dense)          (None, 28)              35868  
  
=====  
Total params: 4085439 (15.58 MB)  
Trainable params: 4043416 (15.42 MB)  
Non-trainable params: 42023 (164.16 KB)
```

4.4.2 Model Training

After constructing the model, the next step is to train it using the training data and their corresponding expected outputs. The training process is initiated with the `model.fit` method. This method takes the training data and the expected output as inputs and trains the model over a specified number of epochs. The progress of the training is displayed on the console, including the training and validation accuracy and loss for each epoch. At the end of the training phase, the final accuracy of the model is reported.

The following code demonstrates the training process:

1. Compile the Model:

- Configure the model with an optimizer, loss function, and metrics.

2. Define Callbacks:

- Use ModelCheckpoint to save the best model based on validation accuracy.
- Use ReduceLROnPlateau to reduce the learning rate when validation accuracy plateaus.

3. Train the Model:

- Train the model using the training data (X_{train} , Y_{train}) and validate it using the validation data (X_{test} , Y_{test}).
- Set the number of epochs and batch size for the training process.

Code for Model Training

```
# Model compilation
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

# Callbacks
ckp_path = "trained_model/model"
model_ckpt = tf.keras.callbacks.ModelCheckpoint(
    filepath=ckp_path,
    monitor="val_accuracy", # Monitoring validation accuracy
    mode="auto",
    save_best_only=True,
    save_weights_only=True
)

reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(
    factor=0.9,
    monitor="val_accuracy", # Monitoring validation accuracy
    mode="auto",
    cooldown=0,
    patience=5,
    verbose=1,
    min_lr=1e-6
)

# Training the model
validation_data=(X_test, Y_test),
Epochs = 10
Batch_size = 32

history = model.fit(
    X_train, Y_train,
    validation_split=0.2,
    batch_size=Batch_size,
    epochs=Epochs,
    callbacks=[model_ckpt, reduce_lr]
)
```

```
Epoch 1/10
929/929 [=====] - 1193s 1s/step - loss: 0.2971 - accuracy: 0.9060 - val_loss: 0.2633 - val_accuracy: 0.9173 - lr: 0.0010
Epoch 2/10
929/929 [=====] - 1126s 1s/step - loss: 0.1808 - accuracy: 0.9418 - val_loss: 0.1950 - val_accuracy: 0.9421 - lr: 0.0010
Epoch 3/10
929/929 [=====] - 11123s 12s/step - loss: 0.1446 - accuracy: 0.9537 - val_loss: 0.1595 - val_accuracy: 0.9514 - lr: 0.0010
Epoch 4/10
929/929 [=====] - 1124s 1s/step - loss: 0.1163 - accuracy: 0.9639 - val_loss: 0.2100 - val_accuracy: 0.9342 - lr: 0.0010
Epoch 5/10
929/929 [=====] - 1123s 1s/step - loss: 0.1048 - accuracy: 0.9668 - val_loss: 0.1454 - val_accuracy: 0.9550 - lr: 0.0010
Epoch 6/10
929/929 [=====] - 1126s 1s/step - loss: 0.0889 - accuracy: 0.9721 - val_loss: 0.3080 - val_accuracy: 0.9257 - lr: 0.0010
Epoch 7/10
929/929 [=====] - 1110s 1s/step - loss: 0.0928 - accuracy: 0.9697 - val_loss: 0.1611 - val_accuracy: 0.9541 - lr: 0.0010
Epoch 8/10
929/929 [=====] - 1113s 1s/step - loss: 0.0768 - accuracy: 0.9755 - val_loss: 0.2142 - val_accuracy: 0.9438 - lr: 0.0010
Epoch 9/10
929/929 [=====] - 1107s 1s/step - loss: 0.0801 - accuracy: 0.9747 - val_loss: 0.1794 - val_accuracy: 0.9483 - lr: 0.0010
Epoch 10/10
929/929 [=====] - 1117s 1s/step - loss: 0.0608 - accuracy: 0.9806 - val_loss: 0.1458 - val_accuracy: 0.9617 - lr: 0.0010
```

4.4.3 Model Evaluation and Conversion

1. Evaluate the Model:

We evaluates the model on the test dataset X_test and Y_test with a batch size of 128. It returns the loss value and any metrics specified during the model compilation.

2. Load Model Weights:

We loads the model weights from the checkpoint file specified by ckp_path.

3. Convert the Model to TensorFlow Lite:

Then we convert the Keras model to TensorFlow Lite format, which is suitable for deployment on mobile and embedded devices.

4. Save the TensorFlow Lite Model:

Saves the converted TensorFlow Lite model to a file named model.tflite.

Code for Model Evaluation and Conversion

```
results = model.evaluate(x_test,y_test, batch_size=128)
✓ 38.5s

41/41 [=====] - 38s 835ms/step - loss: 0.1458 - accuracy: 0.9617
```

```
model.load_weights(ckp_path)
converter=tflite.TFLiteConverter.from_keras_model(model)
tflite_model=converter.convert()

with open("model_letter_accuracy_new.tflite","wb") as f:
    f.write(tflite_model)
```

4.4.4 Making Predictions and Evaluating Performance

```
predict_val=model.predict(x_test,batch_size=32)
predicted_classes = np.argmax(predict_val, axis=1)
```

True values

```
class_indices = np.argmax(Y_test, axis=1)
class_indices[:10]
✓ 0.0s

array([16, 12, 15,  3,  2,  1, 15, 19,  2, 19], dtype=int64)
```

predicted Values

```
predicted_classes[:10]
✓ 0.0s

array([16, 12, 15,  3,  2,  1, 15, 18,  2, 19], dtype=int64)
```

4.5 Android Application

4.6.1 Screens

4.5.1.1 screen one:



Intro Screen: It handles the initialization and user interaction for the introductory screen of the application. This screen navigates to Login screen when a specific TextView(GO) is clicked.

```
1 package com.elmarwacathlab.myapplication;
2 import android.content.Intent;
3 import android.os.Bundle;
4 import android.view.View;
5 import android.widget.TextView;
6
7 import androidx.activity.EdgeToEdge;
8 import androidx.appcompat.app.AppCompatActivity;
9 import androidx.core.graphics.Insets;
10 import androidx.core.view.ViewCompat;
11 import androidx.core.view.WindowInsetsCompat;
12
13 public class IntroActivity extends AppCompatActivity {
14
15     TextView textView5;  2 usages
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         EdgeToEdge.enable( $this$enableEdgeToEdge: this );
21         setContentView(R.layout.activity_intro);
22     }
23 }
```

```

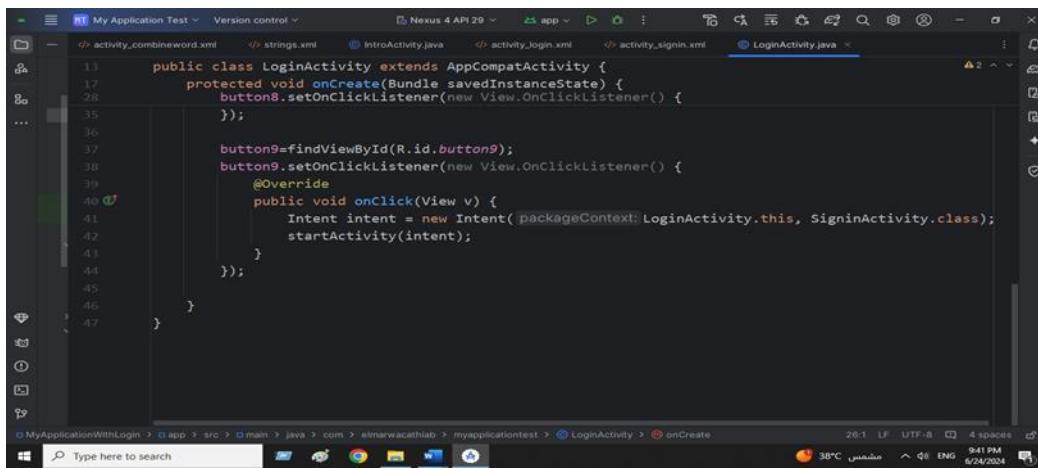
13     public class IntroActivity extends AppCompatActivity {
14         ...
15         @Override
16         protected void onCreate(Bundle savedInstanceState) {
17             super.onCreate(savedInstanceState);
18             EdgeToEdge.enable( $this$enableEdgeToEdge: this );
19             setContentView( R.layout.activity_intro );
20             ViewCompat.setOnApplyWindowInsetsListener( findViewById( R.id.main ), ( v, insets ) -> {
21                 Insets systemBars = insets.getInsets( WindowInsetsCompat.Type.systemBars() );
22                 v.setPadding( systemBars.left, systemBars.top, systemBars.right, systemBars.bottom );
23             } );
24             ...
25         }
26     }
27
28     textView5 = findViewById( R.id.textView5 );
29     textView5.setOnClickListener( new View.OnClickListener() {
30         ...
31         @Override
32         public void onClick( View v ) {
33             Intent intent = new Intent( packageContext: IntroActivity.this, LoginActivity.class );
34             startActivity( intent );
35         }
36     } );
37 }

```

4.5.1.2 screen two:



Login Screen. It initializes the login screen of the application, configures edge-to-edge display, applies window insets to adjust padding for system bars, and sets up click listeners on two buttons to navigate to Definition Screen and Sign up Screen.



The screenshot shows the Android Studio interface with the LoginActivity.java file open in the editor. The code implements the onCreate method of an AppCompatActivity, setting up click listeners for two buttons. The first button's onClick listener starts an activity named SigninActivity. The second button's onClick listener starts an activity named IntroActivity. The code uses findViewById to get the button references and Intent to start new activities.

```
public class LoginActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        button8.setOnClickListener(new View.OnClickListener() {
            ...
        });
        button9=findViewById(R.id.button9);
        button9.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), LoginActivity.this, SigninActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

4.5.1.3 screen three:



Sign up Screen.: It initializes the sign-up screen of the application, configures edge-to-edge display, applies window insets to adjust padding for system bars, and sets up a click listener on a button to navigate to Definition Screen.

My Application Test Version control Nexus 4 API 29 app IntroActivity.java activity_login.xml activity_signin.xml LoginActivity.java SigninActivity.java

```
1 package com.elmarwacathlab.myapplicationtest;
2 import android.content.Intent;
3 import android.os.Bundle;
4 import android.view.View;
5 import android.widget.Button;
6
7 import androidx.activity.EdgeToEdge;
8 import androidx.appcompat.app.AppCompatActivity;
9 import androidx.core.graphics.Insets;
10 import androidx.core.view.ViewCompat;
11 import androidx.core.view.WindowInsetsCompat;
12
13 public class SigninActivity extends AppCompatActivity {
14     Button button8; 2 usages
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         EdgeToEdge.enable($this$enableEdgeToEdge: this);
19         setContentView(R.layout.activity_signin);
20         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
21             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
22
23         });
24
25         button8 = findViewById(R.id.button8);
26         button8.setOnClickListener(new View.OnClickListener() {
27             @Override
28             public void onClick(View v) {
29                 Intent intent = new Intent(packageContext: SigninActivity.this,
30                     DefinActivity.class);
31                 startActivity(intent);
32             }
33         });
34     }
35
36 }
```

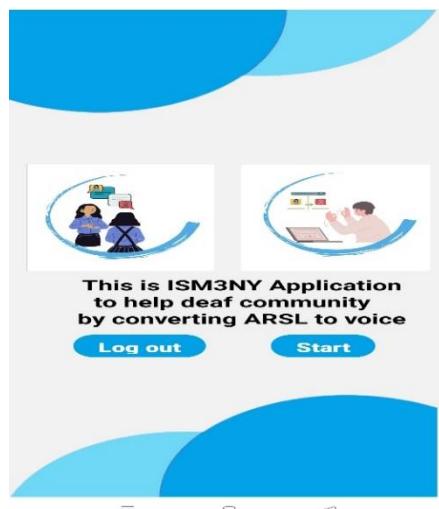
Type here to search 38°C 10:04 PM 6/24/2024

My Application Test Version control Nexus 4 API 29 app IntroActivity.java activity_login.xml activity_signin.xml LoginActivity.java SigninActivity.java

```
13 public class SigninActivity extends AppCompatActivity {
14     protected void onCreate(Bundle savedInstanceState) {
15         EdgeToEdge.enable($this$enableEdgeToEdge: this);
16         setContentView(R.layout.activity_signin);
17         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
18             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
19             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
20
21         });
22
23         button8 = findViewById(R.id.button8);
24         button8.setOnClickListener(new View.OnClickListener() {
25             @Override
26             public void onClick(View v) {
27                 Intent intent = new Intent(packageContext: SigninActivity.this,
28                     DefinActivity.class);
29                 startActivity(intent);
30             }
31         });
32     }
33
34 }
```

Type here to search 24:12 LF UTF-8 4 spaces 38°C 10:05 PM 6/24/2024

4.5.1.4 screen four:

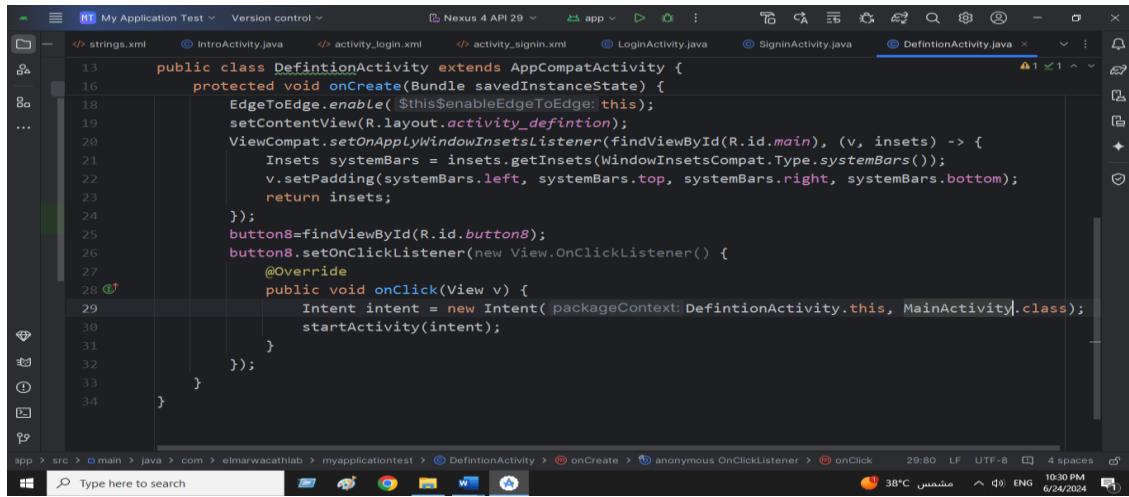


Definition Screen: It initializes the definition screen of the application, configures edge-to-edge display, applies window insets to adjust padding for system bars, and sets up a click listener on a button to navigate to Start application and other button to log out from application.

A screenshot of the Android Studio IDE showing the code for DefinitionActivity.java. The code is as follows:

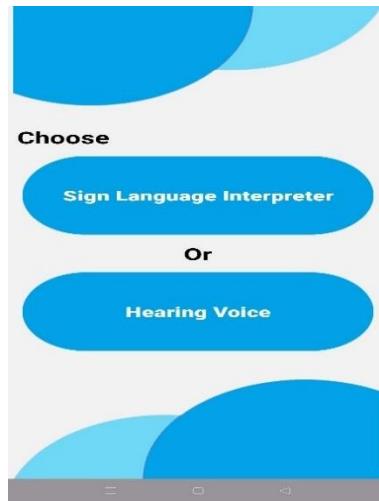
```
1 package com.elmarwacathlab.myapplicationtest;
2 import android.content.Intent;
3 import android.os.Bundle;
4 import android.view.View;
5 import android.widget.Button;
6
7 import androidx.activity.EdgeToEdge;
8 import androidx.appcompat.app.AppCompatActivity;
9 import androidx.core.graphics.Insets;
10 import androidx.core.view.ViewCompat;
11 import androidx.core.view.WindowInsetsCompat;
12
13 public class DefinitionActivity extends AppCompatActivity {
14     Button button8; 2 usages
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         EdgeToEdge.enable( $this$enableEdgeToEdge: this);
19         setContentView(R.layout.activity_definition);
20         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
21             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
22         });
23     }
24 }
```

The code includes imports for various Android components and defines an activity that enables edge-to-edge display and handles window insets.



```
public class DefintionActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_defintion);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        button8=findViewById(R.id.button8);
        button8.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: DefintionActivity.this, MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

4.5.1.5 screen five:



Start application screen: It initializes OpenCV, sets up the user interface with two buttons, one of them it's sign language interpreter used for only detection, the other it's hearing voice used for combine word and sentence and give us the voice , and defines their actions to navigate to different screen (Camera screen and combine word screen).

```
1 package com.elmarwacathlab.myapplicationtest;
2 import androidx.appcompat.app.AppCompatActivity;
3
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.view.View;
8 import android.widget.Button;
9
10 import org.opencv.android.OpenCVLoader;
11
12 import java.io.IOException;
13
14 public class MainActivity extends AppCompatActivity {
15     static {
16         if(OpenCVLoader.initDebug()){
17             Log.d( tag: "MainActivity: ", msg: "Opencv is loaded");
18         } else {
19             Log.d( tag: "MainActivity: ", msg: "Opencv failed to load");
20         }
21     }
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         camera_button=findViewById(R.id.camera_button);
29         camera_button.setOnClickListener(new View.OnClickListener() {
30             @Override
31             public void onClick(View v) {
32                 startActivity(new Intent( packageContext: MainActivity.this,
33                                         CameraActivity.class).
34                             addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));
35             }
36         });
37
38         combine_word_button=findViewById(R.id.combine_word_button);
39         combine_word_button.setOnClickListener(new View.OnClickListener() {
40             @Override
41             public void onClick(View v) {
42                 startActivity(new Intent( packageContext: MainActivity.this,
43                                         combinewordActivity.class).
44                             addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));
45             }
46         });
47     }
48
49 }
```

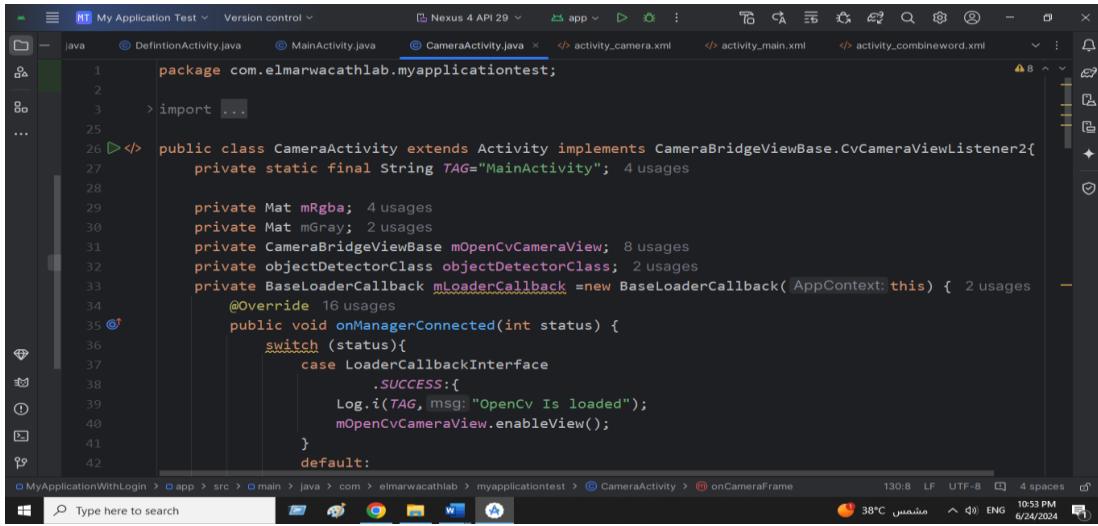
```
14
15     private Button camera_button; 2 usages
16     private Button combine_word_button; 2 usages
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         camera_button=findViewById(R.id.camera_button);
24         camera_button.setOnClickListener(new View.OnClickListener() {
25             @Override
26             public void onClick(View v) {
27                 startActivity(new Intent( packageContext: MainActivity.this,
28                                         CameraActivity.class).
29                             addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));
30             }
31         });
32
33         combine_word_button=findViewById(R.id.combine_word_button);
34         combine_word_button.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 startActivity(new Intent( packageContext: MainActivity.this,
38                                         combinewordActivity.class).
39                             addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));
40             }
41         });
42     }
43
44 }
```

```
14
15     public class MainActivity extends AppCompatActivity {
16         protected void onCreate(Bundle savedInstanceState) {
17             camera_button.setOnClickListener(new View.OnClickListener() {
18                 @Override
19                 public void onClick(View v) {
20                     startActivity(new Intent( packageContext: MainActivity.this,
21                                         CameraActivity.class).
22                             addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));
23                 }
24             });
25
26             combine_word_button=findViewById(R.id.combine_word_button);
27             combine_word_button.setOnClickListener(new View.OnClickListener() {
28                 @Override
29                 public void onClick(View v) {
30                     startActivity(new Intent( packageContext: MainActivity.this,
31                                         combinewordActivity.class).
32                             addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));
33                 }
34             });
35         }
36     }
37
38 }
```

4.5.1.6 screen six



Camera screen: that integrates OpenCV for real-time camera processing. It sets up the camera view, handles permissions, initializes the OpenCV library, and processes camera frames using a TensorFlow Lite model for object detection.

A screenshot of an Android Studio code editor. The main window shows the Java code for 'CameraActivity.java'. The code implements the 'CvCameraViewListener2' interface and handles various camera-related variables and methods. The code editor includes syntax highlighting, code completion suggestions, and a search bar at the bottom. The status bar at the bottom right shows system information like battery level, temperature, and network connection.

```

26     public class CameraActivity extends Activity implements CameraBridgeViewBase.CvCameraViewListener2 {
27         private BaseLoaderCallback mLoaderCallback =new BaseLoaderCallback( ApplicationContext.this ) { 2 usages
28             public void onManagerConnected(int status) {
29                 ...
30             }
31             super.onManagerConnected(status);
32         }
33         public void onManagerDisconnected() {
34             ...
35         }
36         public void onCameraViewCreated(CvCameraView cvCameraView) {
37             ...
38         }
39         public void onCameraViewDestroyed() {
40             ...
41         }
42     };
43     public CameraActivity() { Log.i(TAG, msg:"Instantiated new "+this.getClass()); }
44
45     @Override
46     protected void onCreate(Bundle savedInstanceState) {
47         super.onCreate(savedInstanceState);
48         requestWindowFeature(Window.FEATURE_NO_TITLE);
49         getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
50
51         int MY_PERMISSIONS_REQUEST_CAMERA=0;
52     }
53
54     ...
55
56     ...
57     ...
58
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
249
249
250
251
252
253
254
255
255
256
257
257
258
259
259
260
261
262
263
263
264
265
265
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
14
```

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** My Application Test - Version control - Nexus 4 API 29 - app - TO CA EB - 16
- Project Structure:** Main Activity.java, CameraActivity.java, activity_camera.xml, activity_main.xml, activity_combineword.xml, combinewordActivity.java
- Code Editor:** The file combinewordActivity.java is open. The code implements CameraBridgeViewBase.CvCameraViewListener2 and defines several private fields including TAG, STORAGE_PERMISSION_CODE, and various Mat and Button objects.
- Toolbars and Status Bar:** Includes icons for file operations, navigation, and search. The status bar shows 38°C, 107:21, LF, UTF-8, ENG, and 6/24/2024.
- Bottom Navigation:** Shows the navigation bar with icons for Home, Back, Forward, and Stop.

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** My Application Test > Version control > Nexus 4 API 29 > app > onCreate
- Toolbar:** Back, Forward, Stop, Run, Build, Refresh, Find, Settings, Help, Minimize, Maximize, Close.
- Side Navigation:** Project, Files, Recent, Settings, Help.
- Code Editor:** The code is for `com.elmarwacathlab.myapplicationtest.combinewordActivity`. The current file is `combinewordActivity.java`. The code implements `CameraBridgeViewBase.CvCameraViewListener2` and contains logic for managing a camera connection.

```
public class combinewordActivity extends Activity implements CameraBridgeViewBase.CvCameraViewListener2 {
    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(ApplicationContext.this) { 2 usages
        @Override 16 usages
        public void onManagerConnected(int status) {
            switch (status) {
                case LoaderCallbackInterface.SUCCESS: {
                    Log.i(TAG, msg:"OpenCv Is loaded");
                    mOpenCvCameraView.enableView();
                }
                default: {
                    super.onManagerConnected(status);
                }
            }
            break;
        }
    };
}
```

- Status Bar:** 38°C, مشمس ENG, 11:08 PM, 6/24/2024.
- Bottom Navigation:** Windows icon, Search bar, Recent files.

The screenshot shows the Android Studio interface with the code for `com.elmarwacathlab.myapplicationtest.CombinewordActivity`. The code implements `CameraBridgeViewBase.CvCameraViewListener2` and handles camera permissions and window features.

```
public class CombinewordActivity extends Activity implements CameraBridgeViewBase.CvCameraViewListener2 {
    ...
    public CombinewordActivity() { Log.i(TAG, msg: "Instantiated new " + this.getClass()); }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

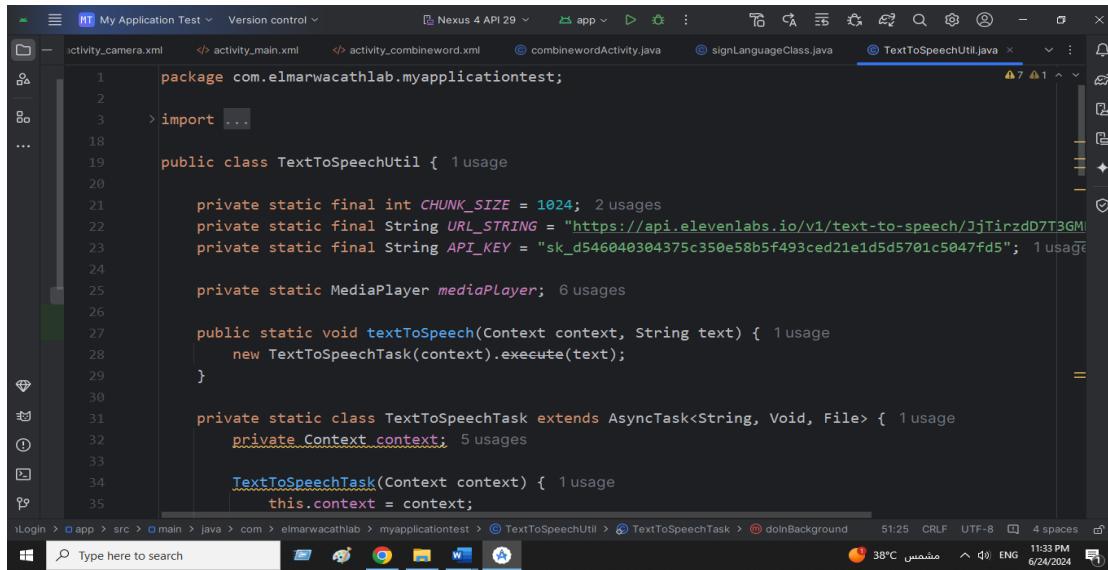
        int MY_PERMISSIONS_REQUEST_CAMERA = 0;
        // if camera permission is not given it will ask for it on device
        if (ContextCompat.checkSelfPermission(context: CombinewordActivity.this,
                Manifest.permission.CAMERA)
            == PackageManager.PERMISSION_DENIED) {
            ActivityCompat.requestPermissions(activity: CombinewordActivity.this,
                new String[]{Manifest.permission.CAMERA}, MY_PERMISSIONS_REQUEST_CAMERA);
        }
    }

    if (ContextCompat.checkSelfPermission(context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
        ...
}
```

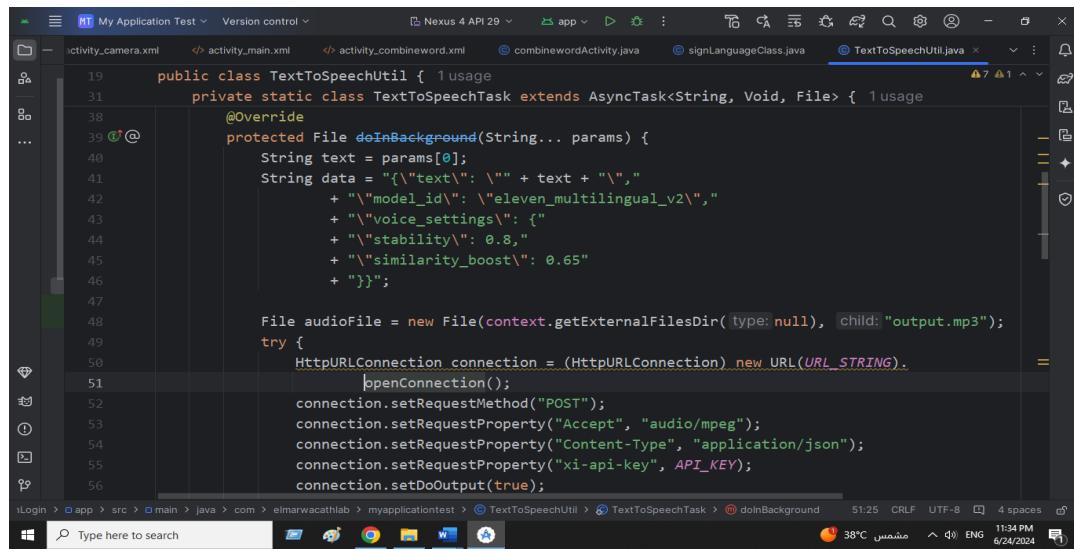
4.5.2 Classes and functionality:

4.5.2.1 Text to speech:

The `TextToSpeechUtil` class provides functionality to convert text to speech using a remote API endpoint. It asynchronously downloads the generated audio file and plays it using Android's `MediaPlayer`.



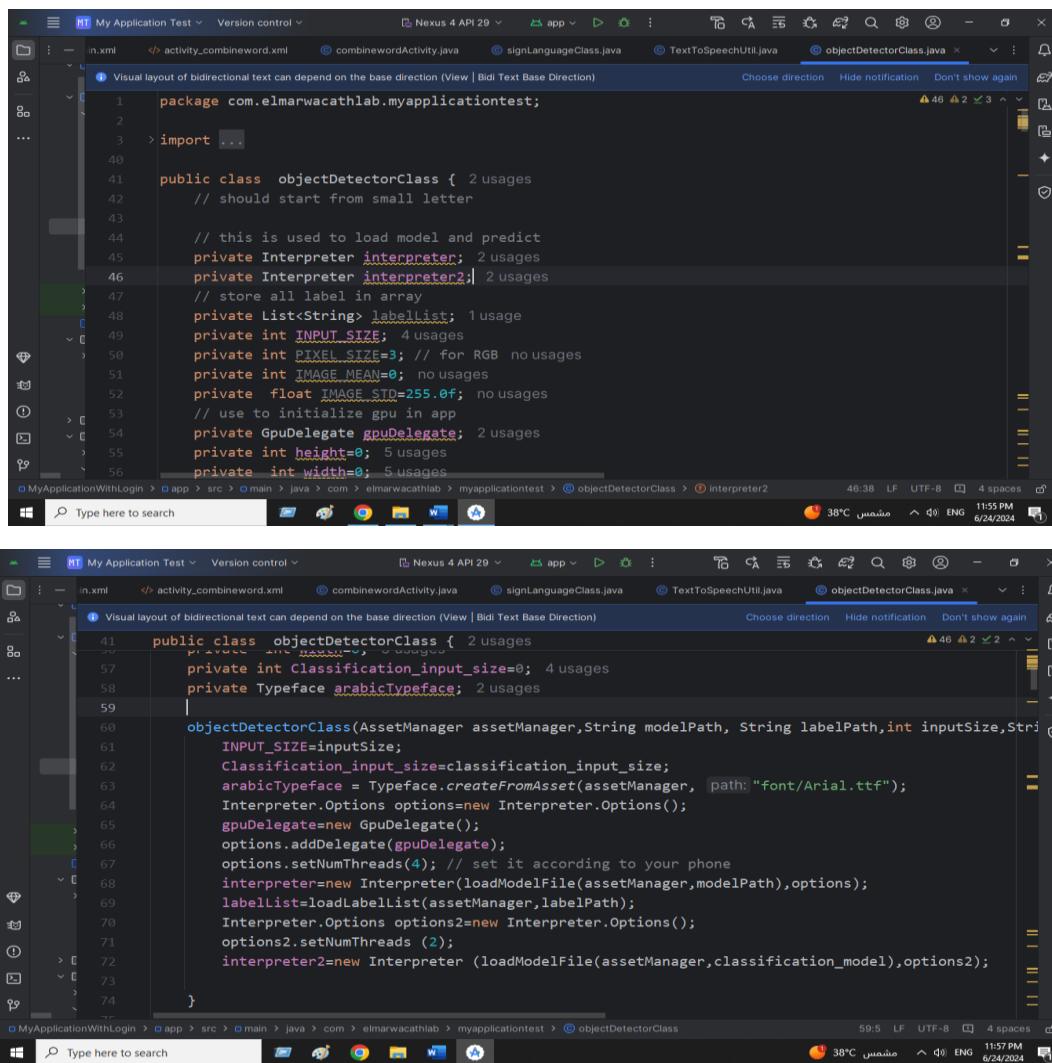
```
1 package com.elmarwacathlab.myapplicationtest;
2
3 > import ...
4
5 public class TextToSpeechUtil { 1 usage
6
7     private static final int CHUNK_SIZE = 1024; 2 usages
8     private static final String URL_STRING = "https://api.elevenlabs.io/v1/text-to-speech/JjTirzdD7T3GM";
9     private static final String API_KEY = "sk_d546040304375c350e58b5f493ced21e1d5d5701c5047fd5"; 1 usage
10
11     private static MediaPlayer mediaPlayer; 6 usages
12
13     public static void textToSpeech(Context context, String text) { 1 usage
14         new TextToSpeechTask(context).execute(text);
15     }
16
17     private static class TextToSpeechTask extends AsyncTask<String, Void, File> { 1 usage
18         private Context context; 5 usages
19
20             TextToSpeechTask(Context context) { 1 usage
21                 this.context = context;
22             }
23
24             @Override
25             protected File doInBackground(String... params) {
26                 String text = params[0];
27                 String data = "{\"text\": \"\" + text + "\","
28                     + "\"model_id\": \"eleven_multilingual_v2\","
29                     + "\"voice_settings\": {"
30                     + "\"stability\": 0.8,"
31                     + "\"similarity_boost\": 0.65"
32                     + "}";
33
34                 File audioFile = new File(context.getExternalFilesDir( type: null), child: "output.mp3");
35                 try {
36                     HttpURLConnection connection = (HttpURLConnection) new URL(URL_STRING).
37                         openConnection();
38                     connection.setRequestMethod("POST");
39                     connection.setRequestProperty("Accept", "audio/mpeg");
40                     connection.setRequestProperty("Content-Type", "application/json");
41                     connection.setRequestProperty("xi-api-key", API_KEY);
42                     connection.setDoOutput(true);
43
44                     connection.connect();
45                     connection.getOutputStream().write(data.getBytes());
46                     connection.getOutputStream().flush();
47                     connection.getOutputStream().close();
48
49                     FileDescriptor fd = connection.getInputStream().getFD();
50                     mediaPlayer.setDataSource(fd);
51                     mediaPlayer.prepare();
52                     mediaPlayer.start();
53
54                     connection.disconnect();
55
56                 } catch (IOException e) {
57                     e.printStackTrace();
58                 }
59             }
60
61             @Override
62             protected void onPostExecute(File result) {
63                 if (result != null) {
64                     mediaPlayer.setDataSource(result.getAbsolutePath());
65                     mediaPlayer.prepare();
66                     mediaPlayer.start();
67                 }
68             }
69
70         }
71
72     }
73
74 }
```



```
19     public class TextToSpeechUtil { 1 usage
20         private static class TextToSpeechTask extends AsyncTask<String, Void, File> { 1 usage
21             ...
22             @Override
23             protected File doInBackground(String... params) {
24                 String text = params[0];
25                 String data = "{\"text\": \"\" + text + "\","
26                     + "\"model_id\": \"eleven_multilingual_v2\","
27                     + "\"voice_settings\": {"
28                     + "\"stability\": 0.8,"
29                     + "\"similarity_boost\": 0.65"
30                     + "}";
31
32                 File audioFile = new File(context.getExternalFilesDir( type: null), child: "output.mp3");
33                 try {
34                     HttpURLConnection connection = (HttpURLConnection) new URL(URL_STRING).
35                         openConnection();
36                     connection.setRequestMethod("POST");
37                     connection.setRequestProperty("Accept", "audio/mpeg");
38                     connection.setRequestProperty("Content-Type", "application/json");
39                     connection.setRequestProperty("xi-api-key", API_KEY);
40                     connection.setDoOutput(true);
41
42                     connection.connect();
43                     connection.getOutputStream().write(data.getBytes());
44                     connection.getOutputStream().flush();
45                     connection.getOutputStream().close();
46
47                     FileDescriptor fd = connection.getInputStream().getFD();
48                     mediaPlayer.setDataSource(fd);
49                     mediaPlayer.prepare();
50                     mediaPlayer.start();
51
52                     connection.disconnect();
53
54                 } catch (IOException e) {
55                     e.printStackTrace();
56                 }
57             }
58
59             @Override
60             protected void onPostExecute(File result) {
61                 if (result != null) {
62                     mediaPlayer.setDataSource(result.getAbsolutePath());
63                     mediaPlayer.prepare();
64                     mediaPlayer.start();
65                 }
66             }
67
68         }
69
70     }
71
72 }
```

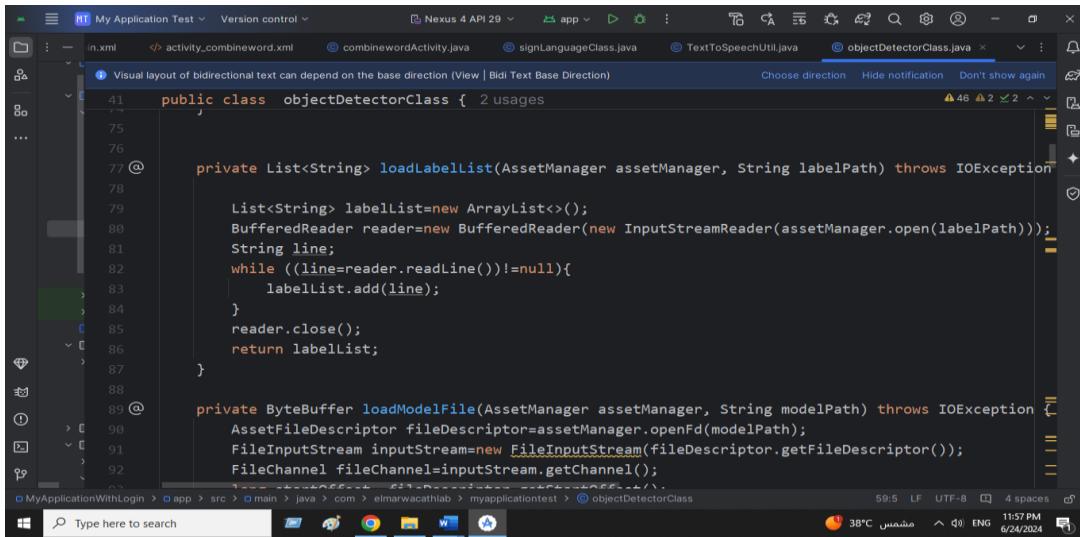
4.5.2.2 object detection

objectDetectorClass integrates TensorFlow Lite models for object detection and classification into an Android application using OpenCV for image manipulation. It demonstrates how to load models, perform inference, process results, and annotate images with detected objects and their classifications. This setup is tailored for applications requiring real-time object detection and classification on Android devices.



```
1 package com.elmarwacathlab.myapplicationtest;
2
3 > import ...
4
5 public class objectDetectorClass { 2 usages
6     // should start from small letter
7
8     // this is used to load model and predict
9     private Interpreter interpreter; 2 usages
10    private Interpreter interpreter2; 2 usages
11
12    // store all label in array
13    private List<String> labelList; 1 usage
14    private int INPUT_SIZE; 4 usages
15    private int PIXEL_SIZE=3; // for RGB no usages
16    private int IMAGE_MEAN=0; no usages
17    private float IMAGE_STD=255.0f; no usages
18    // use to initialize gpu in app
19    private GpuDelegate gpuDelegate; 2 usages
20    private int height=0; 5 usages
21    private int width=0; 5 usages
22
23 }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

```
1 package com.elmarwacathlab.myapplicationtest;
2
3 > import ...
4
5 public class objectDetectorClass { 2 usages
6     private AssetManager assetManager;
7     private int Classification_input_size=0; 4 usages
8     private Typeface arabicTypeface; 2 usages
9
10    objectDetectorClass(AssetManager assetManager, String modelPath, String labelPath, int inputSize, String
11        INPUT_SIZE=inputSize;
12        Classification_input_size=classification_input_size;
13        arabicTypeface = Typeface.createFromAsset(assetManager, path: "font/Arial.ttf");
14        Interpreter.Options options=new Interpreter.Options();
15        gpuDelegate=new GpuDelegate();
16        options.addDelegate(gpuDelegate);
17        options.setNumThreads(4); // set it according to your phone
18        interpreter=new Interpreter(loadModelFile(assetManager, modelPath),options);
19        labelList=loadLabelList(assetManager, labelPath);
20        Interpreter.Options options2=new Interpreter.Options();
21        options2.setNumThreads (2);
22        interpreter2=new Interpreter (loadModelFile(assetManager, classification_model),options2);
23    }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
39
40
41
42
43
44
45
46
47
48
49
49
50
51
52
53
54
55
56
```



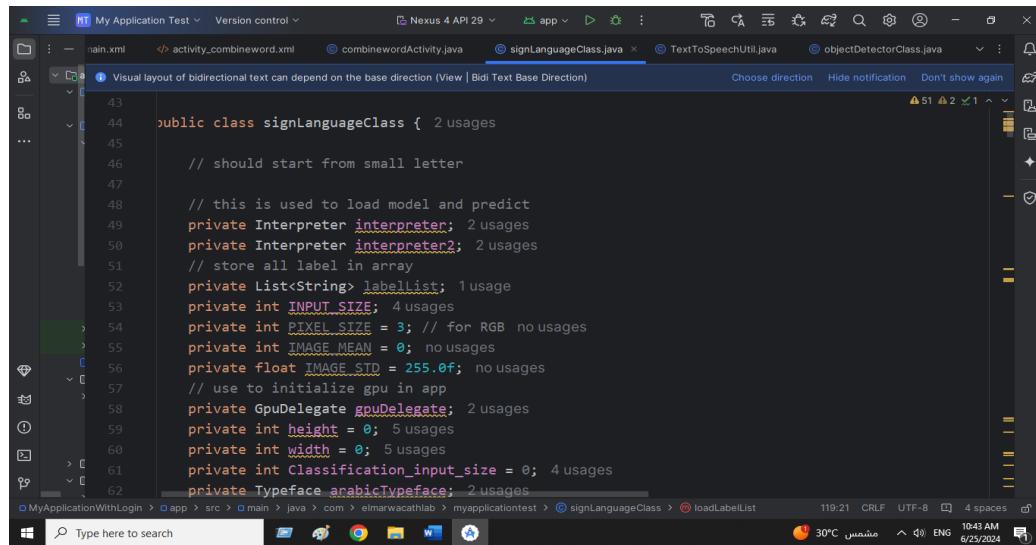
```
public class objectDetectorClass { 2 usages

    private List<String> loadLabelList(AssetManager assetManager, String labelPath) throws IOException {
        List<String> labelList=new ArrayList<>();
        BufferedReader reader=new BufferedReader(new InputStreamReader(assetManager.open(labelPath)));
        String line;
        while ((line=reader.readLine())!=null){
            labelList.add(line);
        }
        reader.close();
        return labelList;
    }

    private ByteBuffer loadModelFile(AssetManager assetManager, String modelPath) throws IOException {
        AssetFileDescriptor fileDescriptor=assetManager.openFd(modelPath);
        FileInputStream inputStream=new FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel=inputStream.getChannel();
        long startOffset=fileDescriptor.getStartOffset();
        long declaredSize=fileDescriptor.getDeclaredSize();
        return fileChannel.read(ByteBuffer.allocate(declaredSize+startOffset));
    }
}
```

4.5.2.3 Sign language:

The signLanguageClass encapsulates functionality for recognizing sign language gestures using TensorFlow Lite models integrated with OpenCV for Android. It handles loading models, processing images, predicting gestures, and rendering results on screen.

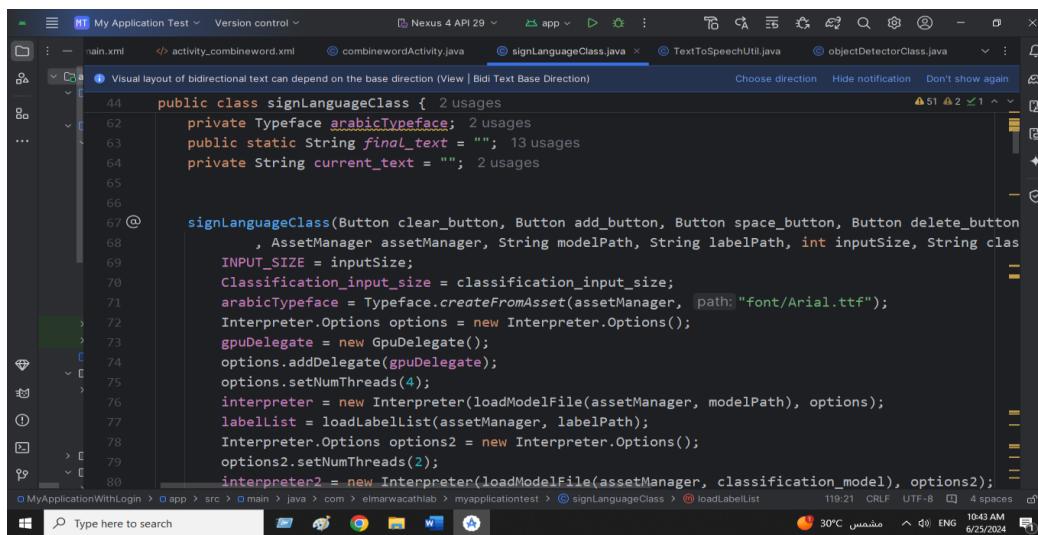


```
public class signLanguageClass { 2 usages

    // should start from small letter

    // this is used to load model and predict
    private Interpreter interpreter; 2 usages
    private Interpreter interpreter2; 2 usages
    // store all label in array
    private List<String> labellist; 1 usage
    private int INPUT_SIZE; 4 usages
    private int PIXEL_SIZE = 3; // for RGB no usages
    private int IMAGE_MEAN = 0; no usages
    private float IMAGE_STD = 255.0f; no usages
    // use to initialize gpu in app
    private GpuDelegate gpuDelegate; 2 usages
    private int height = 0; 5 usages
    private int width = 0; 5 usages
    private int Classification_input_size = 0; 4 usages
    private Typeface arabicTypeface; 2 usages
}

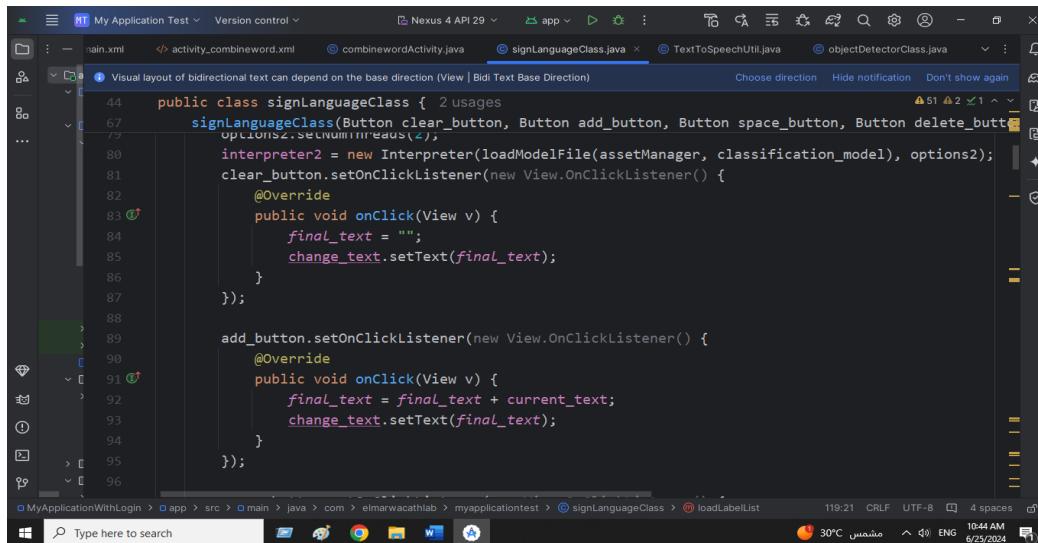
private void loadLabelList() {
    List<String> labelList = new ArrayList<>();
    try {
        AssetFileDescriptor fileDescriptor = getAssets().openFd("labelList.txt");
        FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel = inputStream.getChannel();
        long startOffset = fileDescriptor.getStartOffset();
        long declaredSize = fileDescriptor.getDeclaredSize();
        ByteBuffer buffer = fileChannel.read(ByteBuffer.allocate(declaredSize + startOffset));
        buffer.rewind();
        Charset charset = StandardCharsets.UTF_8;
        CharsetDecoder decoder = charset.newDecoder();
        while (buffer.hasRemaining()) {
            String line = decoder.decode(ByteBuffer.wrap(buffer.slice()));
            labelList.add(line);
        }
        fileDescriptor.close();
        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```



```
public class signLanguageClass { 2 usages
    private Typeface arabicTypeface; 2 usages
    public static String final_text = ""; 13 usages
    private String current_text = ""; 2 usages

    signLanguageClass(Button clear_button, Button add_button, Button space_button, Button delete_button
                      , AssetManager assetManager, String modelPath, String labelPath, int inputSize, String clas
                      INPUT_SIZE = inputSize;
    Classification_input_size = classification_input_size;
    arabicTypeface = Typeface.createFromAsset(assetManager, path: "font/Arial.ttf");
    Interpreter.Options options = new Interpreter.Options();
    gpuDelegate = new GpuDelegate();
    options.addDelegate(gpuDelegate);
    options.setNumThreads(4);
    interpreter = new Interpreter(loadModelFile(assetManager, modelPath), options);
    labelList = loadLabelList(assetManager, labelPath);
    Interpreter.Options options2 = new Interpreter.Options();
    options2.setNumThreads(2);
    interpreter2 = new Interpreter(loadModelFile(assetManager, classification_model), options2);
}

MyApplicationWithLogin > app > src > main > java > com > elmarwacathlab > myapplicationtest > signLanguageClass > @loadLabelList
```



```
public class signLanguageClass { 2 usages
    signLanguageClass(Button clear_button, Button add_button, Button space_button, Button delete_button
                      , options2.setNumThreads(2);
    interpreter2 = new Interpreter(loadModelFile(assetManager, classification_model), options2);
    clear_button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final_text = "";
            change_text.setText(final_text);
        }
    });

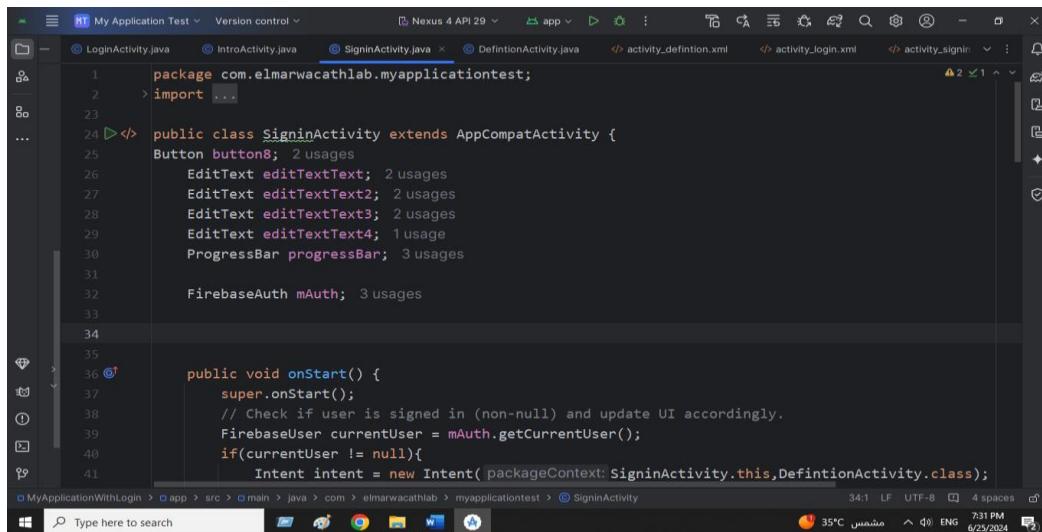
    add_button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final_text = final_text + current_text;
            change_text.setText(final_text);
        }
    });
}

MyApplicationWithLogin > app > src > main > java > com > elmarwacathlab > myapplicationtest > signLanguageClass > @loadLabelList
```

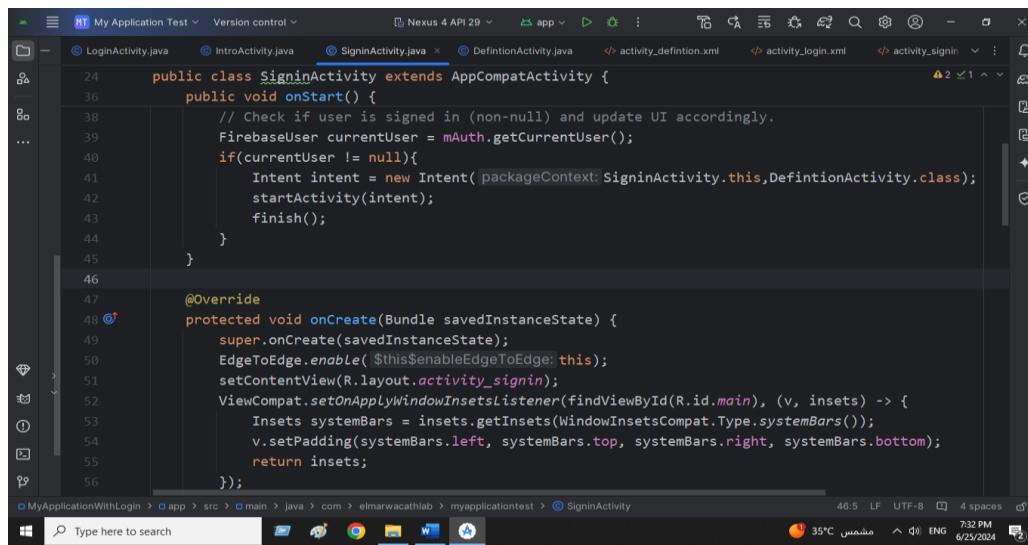
4.5.3 Use Firebase

It's platform by google it's provide a variety and service . we use in our application Authentication to provides backend service for easy authentication including sign in methods like email and password.

4.5.3.1 To sign up:

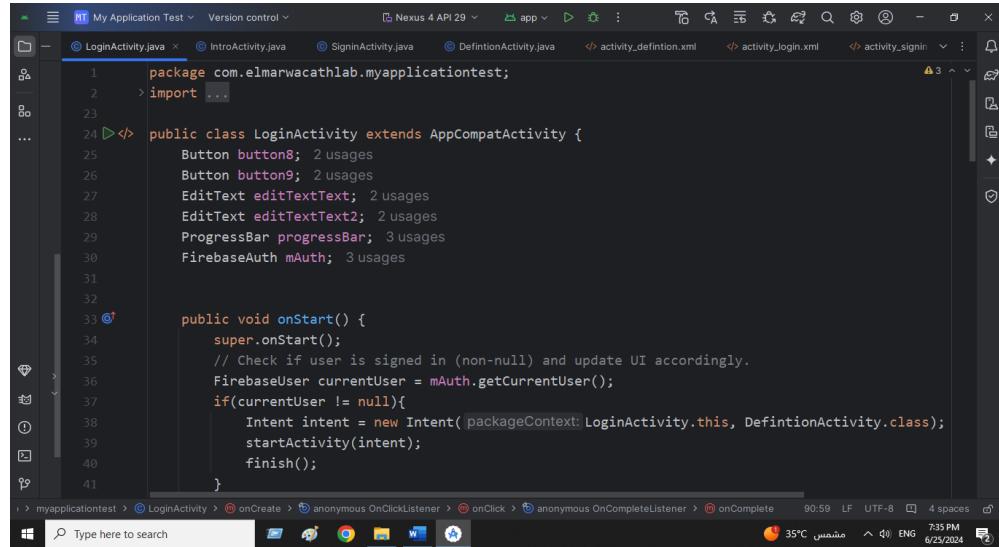


```
1 package com.elmarwacathlab.myapplicationtest;
2 import ...
3 ...
4 public class SigninActivity extends AppCompatActivity {
5     Button button8; 2 usages
6     EditText editTextText; 2 usages
7     EditText editTextText2; 2 usages
8     EditText editTextText3; 2 usages
9     EditText editTextText4; 1 usage
10    ProgressBar progressBar; 3 usages
11
12    FirebaseAuth mAuth; 3 usages
13
14
15    public void onStart() {
16        super.onStart();
17        // Check if user is signed in (non-null) and update UI accordingly.
18        FirebaseUser currentUser = mAuth.getCurrentUser();
19        if(currentUser != null){
20            Intent intent = new Intent(getApplicationContext(),SigninActivity.this,DefintionActivity.class);
21        }
22    }
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
```

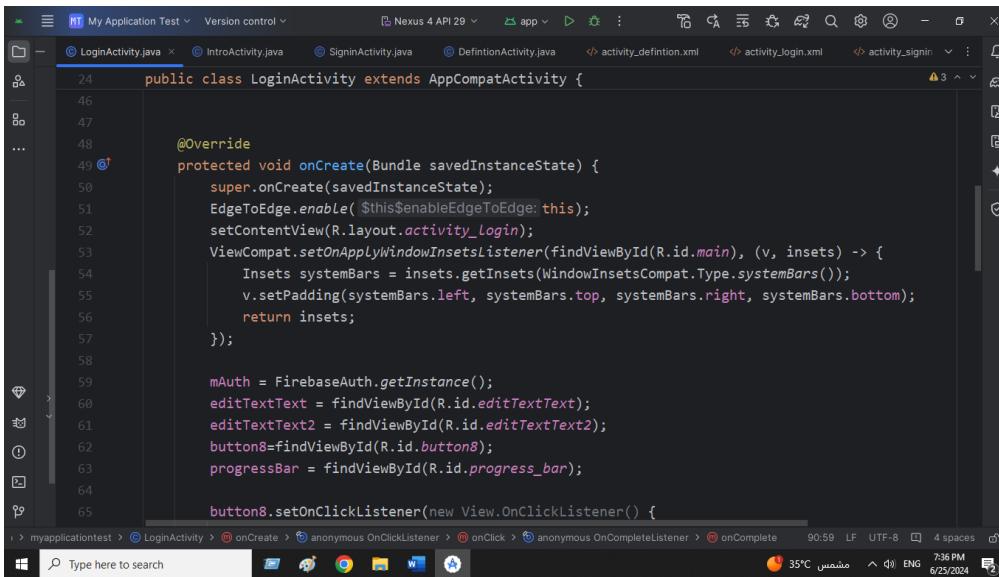


```
24    public class SigninActivity extends AppCompatActivity {
25        public void onStart() {
26            // Check if user is signed in (non-null) and update UI accordingly.
27            FirebaseUser currentUser = mAuth.getCurrentUser();
28            if(currentUser != null){
29                Intent intent = new Intent(getApplicationContext(),SigninActivity.this,DefintionActivity.class);
30                startActivity(intent);
31                finish();
32            }
33        }
34
35        @Override
36        protected void onCreate(Bundle savedInstanceState) {
37            super.onCreate(savedInstanceState);
38            EdgeToEdge.enable($this$enableEdgeToEdge: this);
39            setContentView(R.layout.activity_signin);
40            ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
41                Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
42                v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
43                return insets;
44            });
45        }
46
47
48
49
50
51
52
53
54
55
56
```

4.5.3.2 To log in:

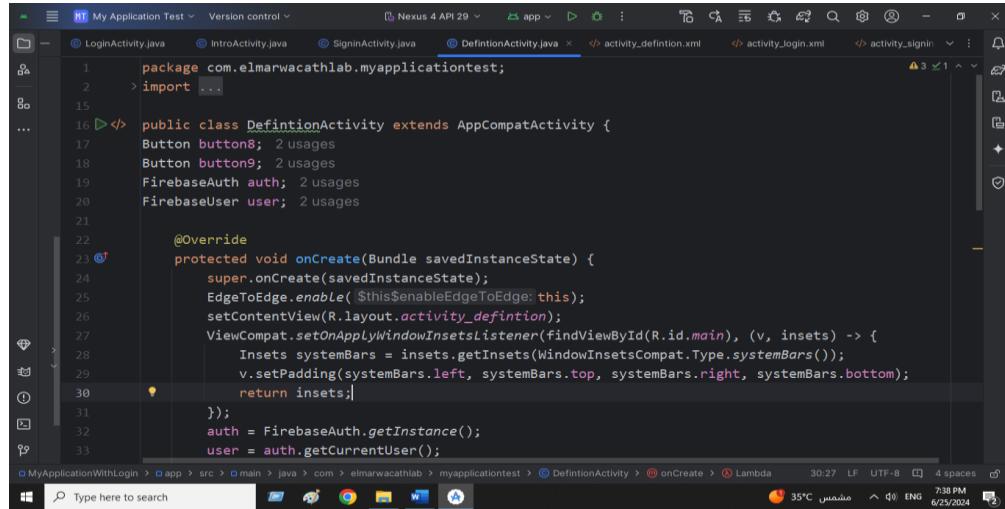


```
1 package com.elmarwacathlab.myapplicationtest;
2 import ...
3 ...
4 public class LoginActivity extends AppCompatActivity {
5     Button button8; 2 usages
6     Button button9; 2 usages
7     EditText editTextText; 2 usages
8     EditText editTextText2; 2 usages
9     ProgressBar progressBar; 3 usages
10    FirebaseAuth mAuth; 3 usages
11
12
13    public void onStart() {
14        super.onStart();
15        // Check if user is signed in (non-null) and update UI accordingly.
16        FirebaseUser currentUser = mAuth.getCurrentUser();
17        if(currentUser != null){
18            Intent intent = new Intent(getApplicationContext(), DefintionActivity.class);
19            startActivity(intent);
20            finish();
21        }
22    }
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
```

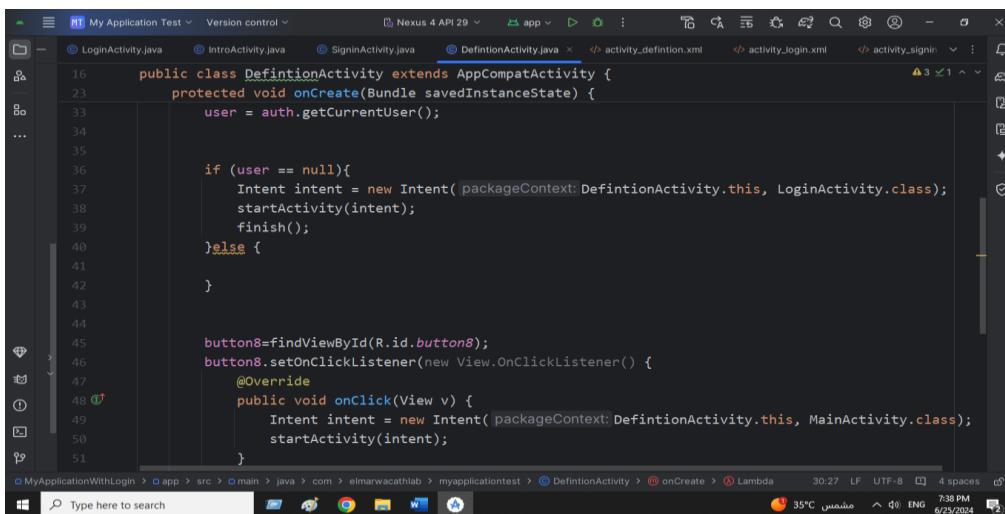


```
24 public class LoginActivity extends AppCompatActivity {
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44 @Override
45 protected void onCreate(Bundle savedInstanceState) {
46     super.onCreate(savedInstanceState);
47     EdgeToEdge.enable(EdgeToEdge.this);
48     setContentView(R.layout.activity_login);
49     ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
50         Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
51         v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
52         return insets;
53     });
54
55
56
57
58
59
60
61
62
63
64
65     mAuth = FirebaseAuth.getInstance();
66     editTextText = findViewById(R.id.editTextText);
67     editTextText2 = findViewById(R.id.editTextText2);
68     button8 = findViewById(R.id.button8);
69     progressBar = findViewById(R.id.progress_bar);
70
71     button8.setOnClickListener(new View.OnClickListener() {
```

4.5.3.3 To log out:



```
1 package com.elmarwacathlab.myapplicationtest;
2 > import ...
3 ...
4 public class DefintionActivity extends AppCompatActivity {
5     Button button8; 2 usages
6     Button button9; 2 usages
7     FirebaseAuth auth; 2 usages
8     FirebaseUser user; 2 usages
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
14        setContentView(R.layout.activity_defintion);
15        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
16            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
17            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
18        });
19        auth = FirebaseAuth.getInstance();
20        user = auth.getCurrentUser();
21    }
22}
```



```
16     public class DefintionActivity extends AppCompatActivity {
17         protected void onCreate(Bundle savedInstanceState) {
18             user = auth.getCurrentUser();
19
20             if (user == null){
21                 Intent intent = new Intent( packageContext:DefintionActivity.this, LoginActivity.class);
22                 startActivity(intent);
23                 finish();
24             }else {
25
26             }
27
28             button8=findViewById(R.id.button8);
29             button8.setOnClickListener(new View.OnClickListener() {
30                 @Override
31                 public void onClick(View v) {
32                     Intent intent = new Intent( packageContext:DefintionActivity.this, MainActivity.class);
33                     startActivity(intent);
34                 }
35             });
36         }
37     }
38 }
```

<<we have cloud contain the all account create in our application

The screenshot shows the Firebase Authentication console interface. At the top, there is a header bar with icons for battery, signal, and network. Below the header, the URL is .firebase.google.com. The main title is "Authentication". Below the title, there are tabs for "Users", "Sign-in method", "Templates", and "Usage". A prominent yellow warning banner at the top states: "Cross origin redirect sign-in on Google Chrome M115+ is no longer supported, and will stop working on June 24, 2024." Below the banner is a search bar with placeholder text "Search by email address, phone number, ...". There is a blue "Add user" button. The main content area displays a table of users:

User Email	Action	More
vv@gmail.com	<input type="checkbox"/>	<input type="checkbox"/>
salma.dd@gmail.com	<input type="checkbox"/>	<input type="checkbox"/>
omar@gmail.com	<input type="checkbox"/>	<input type="checkbox"/>
salma@gmail.com	<input type="checkbox"/>	<input type="checkbox"/>

Below the table, there are controls for "Rows per page" (set to 50) and navigation buttons for "1 - 4 of 4", "<", and ">".

4.6 Experimental Results

In this section, the experimental results of the sign language recognition system are presented and analyzed. The evaluation metrics include a confusion matrix, training and validation curves, as well as precision, recall, and F1-score calculations. Additionally, visual representation of the recognition results is provided below:

Confusion matrix

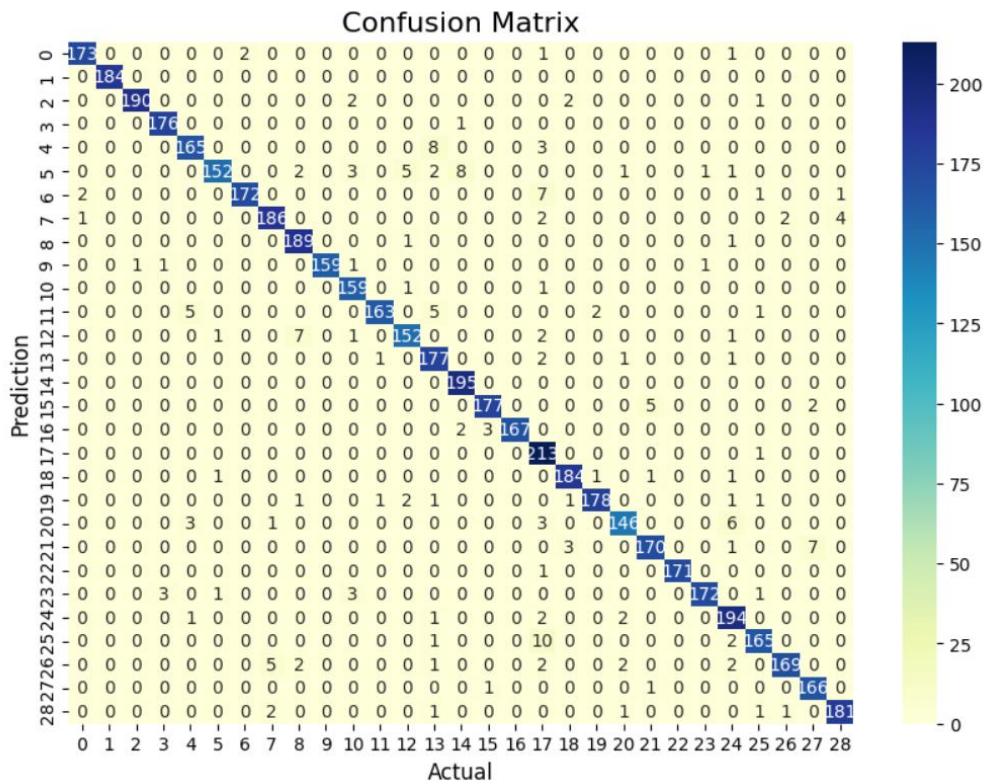


Figure 4-2 :Confusion Matrix

Classification Report

The classification report provides key metrics for evaluating the performance of ArSL recognition model: precision, recall, f1-score, and support for each class.

Precision: The ratio of correctly predicted positive observations to the total predicted positives.

High precision indicates a low false positive rate.

Ranges from 0.86 to 1.00, indicating that the model generally predicts well with low false positives.

Recall: The ratio of correctly predicted positive observations to all observations in the actual class. High recall indicates a low false negative rate.

Ranges from 0.87 to 1.00, suggesting the model is effective at capturing the true positives for most classes.

F1-score: The weighted average of precision and recall. It is useful for comparing the balance between precision and recall.

Ranging from 0.92 to 1.00, which shows a consistent performance across classes.

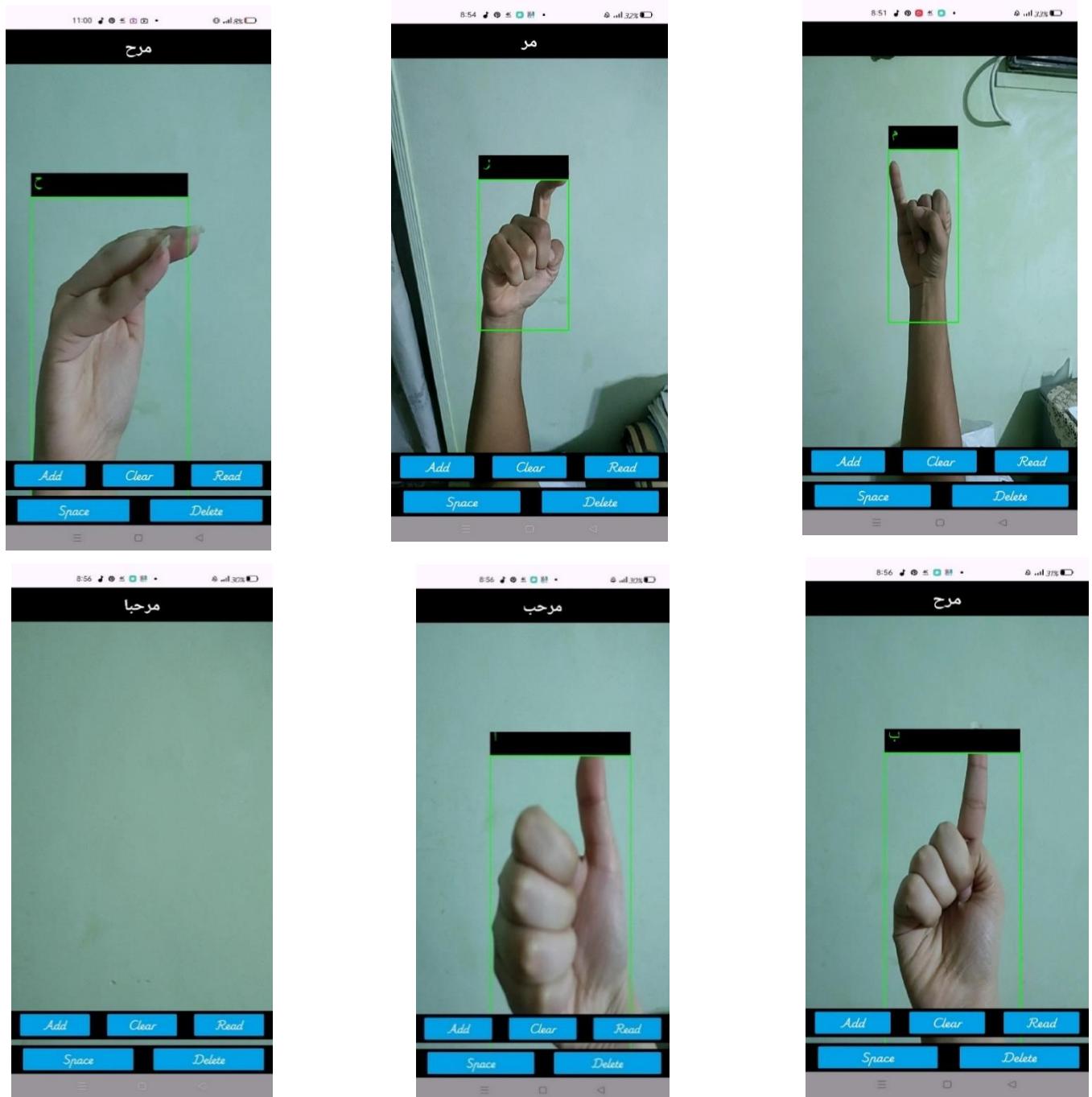
	precision	recall	f1-score	support
0	0.98	0.98	0.98	177
1	1.00	1.00	1.00	184
2	0.99	0.97	0.98	195
3	0.98	0.99	0.99	177
4	0.95	0.94	0.94	176
5	0.98	0.87	0.92	175
6	0.99	0.94	0.96	183
7	0.96	0.95	0.96	195
8	0.94	0.99	0.96	191
9	1.00	0.98	0.99	163
10	0.94	0.99	0.96	161
11	0.99	0.93	0.96	176
12	0.94	0.93	0.94	164
13	0.90	0.97	0.93	182
14	0.95	1.00	0.97	195
15	0.98	0.96	0.97	184
16	1.00	0.97	0.99	172
17	0.86	1.00	0.92	214
18	0.97	0.98	0.97	188
19	0.98	0.96	0.97	186
20	0.95	0.92	0.94	159
21	0.96	0.94	0.95	181
22	1.00	0.99	1.00	172
23	0.99	0.96	0.97	180
24	0.92	0.97	0.94	200
25	0.96	0.93	0.94	178
26	0.98	0.92	0.95	183
27	0.95	0.99	0.97	168
28	0.97	0.97	0.97	187
accuracy			0.96	5246
macro avg	0.96	0.96	0.96	5246
weighted avg	0.96	0.96	0.96	5246

Figure 4-2: Precision, Recall, and F1-Score

4.7 Sign Language Recognition Results

Results: We have gesture signs for the letter that we combine together to form word.

Test Results: We test the word <>مرحبا<>



Chapter 5

Conclusion and Future Work

5 Conclusion

5.1 Conclusion

The goal of this project was to develop an effective Arabic Sign Language (ArSL) recognition system using a Convolutional Neural Network (CNN) based on the EfficientNet architecture. We successfully trained and implemented a model capable of recognizing 29 Arabic sign language gestures with high accuracy. The trained model was integrated into an Android application to facilitate real-time gesture recognition.

The use of EfficientNet, a state-of-the-art CNN architecture known for its balance of accuracy and efficiency, significantly contributed to the high performance of our system. The model achieved an impressive accuracy of 95% on the test dataset, demonstrating its capability to generalize well across various gestures and conditions. Extensive data augmentation techniques were applied, which enhanced the model's robustness and ability to handle diverse inputs.

Despite these successes, the project faced challenges such as occasional misclassifications due to similar gestures, variations in hand positioning, and different lighting conditions. Additionally, the dataset size was a limitation, suggesting that a more extensive dataset could further improve model performance.

This project has significant implications for enhancing communication between the deaf community and others. The developed ArSL recognition system offers a practical tool for real-time sign language interpretation, potentially benefiting educational environments, assistive technologies, and inclusive communication platforms.

5.2 Future Work

While this project has made substantial progress, several avenues for future research and development can further enhance the system's performance and applicability:

1. Dataset Expansion:

Increase the size and diversity of the dataset by incorporating more variations of each gesture, including different lighting conditions, hand orientations, and backgrounds. Collaborating with sign language communities to gather more data can improve the model's robustness.

2. Advanced Data Augmentation:

Implement more sophisticated data augmentation techniques, such as Generative Adversarial Networks (GANs), to create synthetic data that can further enhance the training dataset and improve model performance.

3. Enhanced Model Architecture:

Experiment with advanced variations of EfficientNet or other cutting-edge architectures, such as EfficientNetV2 or transformer-based models, to achieve even higher recognition accuracy and efficiency.

4. User Interface (UI) and User Experience (UX) Improvements:

Enhancing the UI and UX to make the application more intuitive and user-friendly. This could involve redesigning the layout, improving navigation, and adding features like gesture tutorials and interactive guides.

5. Customizable Settings:

Allowing users to customize various settings, such as gesture sensitivity, language preferences, and display options, to cater to individual needs and preferences.

Appendix



We received your application and it is currently under review by our panel of experts. They will be carefully evaluating each submission based on several criteria, including the potential impact of your solution, the feasibility of your business model, and the strength of your team.

We would like to remind you that the registration for the challenge is still open, and you can refer the application to your network! We will be in touch with you after the review process is completed. We anticipate announcing the accepted startups in the next few weeks.

In the meantime, we encourage you to stay connected with us by following our social media channels and signing up for our newsletter. We will be sharing updates and resources to help you continue to grow and develop your Graduation Project.

Thank you for your interest in the Egypt IoT & AI Challenge, and we wish you the best of luck.

Sincerely,
Egypt IoT & AI Challenge 2023 Organizing Committee

In case you've any inquiries, please contact us at
Egypt@ArabIoTAI.org

A regional capacity building and pre-incubation program
for innovative ideas in the areas of IoT & AI and related

Reference

1. https://sist.sathyabama.ac.in/sist_naac/documents/1.3.4/18-b.e-cse-batchno-111.pdf
2. <https://www.scribd.com/document/595521693/Software-Requirements-Specification-Sign-Language-to-Text>
3. <https://www.slideshare.net/slideshow/real-time-conversion-of-sign-language-to-text-and-speech/259552542>
4. <https://ijrpr.com/uploads/V4ISSUE4/IJRPR12225.pdf>
5. “**Jabatan Kebajikan Masyarakat.**”
<https://www.jkm.gov.my/jkm/index.php?r=portal/full&id=ZUFHVTB1NnJWM0EreGtwNC9Vb1hvdz09>
 - a. (Accessed Jun. 09, 2023).
6. https://www.researchgate.net/publication/360384816_Sign_language_Re cognition_using_Deep_CNN
7. <https://www.atlantis-press.com/article/125986284.pdf>
8. <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-ToUnderstanding->
9. <https://www.kaggle.com/code/gauthamupadhyaya/image-classification-using-cnn-and-efficientnetb0>
10. [https://keras.io/examples/vision/image classification efficientnet fine t uning/](https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/)
11. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

ملخص

الهدف من هذا المشروع هو تطوير نظام فوري لتحويل لغة الإشارة إلى نص وكلام، وتسهيل التواصل بين الأفراد الصم أو ضعاف السمع وأولئك الذين ليسوا على دراية بلغة الإشارة. ويستفيد النظام من رؤية الكمبيوتر للتعرف بدقة على إيماءات لغة الإشارة وتفسيرها، وتحوילها إلى نص مناظر وإنشاء مخرجات كلامية مركبة.

يستخدم النظام نظام إدخال يعتمد على الكاميرا، والذي يلتقط بيانات الفيديو المستخدم الذي يقوم بإيماءات لغة الإشارة. تتم معالجة إطارات الفيديو باستخدام خوارزميات الرؤية الحاسوبية، باستخدام تقنيات مثل اكتشاف الأشياء وتقدير وضعية اليد للتعرف بدقة على إيماءات اليد. يتم بعد ذلك ربط الإيماءات المعترف بها بقاموس لغة الإشارة المحدد مسبقاً، حيث تتوافق كل إيماءة مع كلمة أو عبارة معينة. يتم تحقيق هذا التعيين باستخدام خوارزميات التعلم الآلي، التي تم تدريبها على مجموعة كبيرة من بيانات إيماءات لغة الإشارة لتحديد الأنماط وتعيين الإيماءات لتمثيلاتها النصية المقابلة. بمجرد تعيين الإيماءات إلى نص، يقوم النظام بإنشاء مخرجات كلام مركبة باستخدام تقنية تحويل النص إلى كلام.

يوفر الكلام المركب تمثيلاً سمعياً للغة الإشارة، مما يمكن الأفراد الذين ليسوا على دراية بلغة الإشارة من فهم الرسائل المرسلة والرد عليها. إن تطوير نظام تحويل لغة الإشارة في الوقت الفعلي لديه القدرة على سد فجوة التواصل بين الأفراد الذين يستخدمون لغة الإشارة وأولئك الذين لا يستخدموها. من خلال توفير تحويل فعال ودقيق للغة الإشارة إلى نص وكلام، يهدف هذا المشروع إلى تمكين الأفراد ذوي الإعاقة السمعية من التواصل بشكل أكثر فعالية وشمولية في مختلف البيئات الاجتماعية والمهنية.

شکر و تقدیر

الحمد لله الذي هدانا لهذا وما كنا لننهي لولا أن هدانا الله." أولاً وقبل كل شيء نود أن نحمد الله على نعمه علينا. ونود أن نعرب عن صادق تمنياتنا الشكر والتقدير لمشرفنا: الدكتور ممتاز الخولي على نصائحه المتخصصة، وحرصه على اكتسابنا المعرفة ومواكبة تطور أساليب التكنولوجيا الحديثة وتوجيهاته طوال هذا المشروع.

نود تقديم الشكر لأعضاء القسم (قسم هندسة النظم والحواسيب) لمساعدتهم التي لا تقدر بثمن والتي قدموها خلال العملية التعليمية لدينا، ونود أن نعرب عن عميق امتناننا وحبنا الكبير لعائلاتنا الذين جعلونا نستمر، ولم يكن هذا العمل ممكناً بدون دعمهم النفسي وتشجيعهم لنا، لقد وقفوا إلى جانبنا في أصعب الأوقات.

جامعة الأزهر

كلية الهندسة للبنات

قسم النظم و الحاسوبات



معالجة لغة الاشارة بإستخدام الأنظمة الذكية.

" اسمعني "

يقدم هذا المشروع استكمالا جزئيا لمتطلبات درجة البكالوريوس في هندسة النظم و الحاسوبات.

مقدم من الطالبات:

رقم الجلوس: 20248624

أسماء مؤيد عطية محمد

رقم الجلوس: 20248636

أمنية حسن أحمد بكر

رقم الجلوس: 20248674

دعاء محمد القرني

رقم الجلوس: 20248702

سلمي مجدي سعيد محمد

رقم الجلوس: 20248703

سلمي وائل عبد المجيد حسن

رقم الجلوس: 20248706

سمية محمد أمر الله السيد

تحت اشراف :

د. ممتاز سعد الخولي

يوليو، 2024