School of Science and Engineering

Implementation of a mobile application to translate sign language using Mediapipe and a Multilayered Perceptron

Capstone Design Interim Report

March 2024

M'hamed Alghali Joudary

Supervised by:

Dr. Chtouki Yousra

Student conduct and copyright

The supervisor and the student (the Capstoner) agree that:

1. Permission has been obtained for any third party content (eg Data, illustrations, photographs, charts or maps).

2. The results described in this report have not previously been published

Student's name and signature: **M'hamed Alghali Joudary** [illegible]

Supervisor's name and signature: **Yousra Chtouki** ***Yousra Chtouki***

Implementation of a mobile application to translate sign language using Mediapipe and a Multilayered Perceptron

Capstone Report

Student Statement:

I, M'hamed Alghali Joudary, affirm that I have applied ethics to the design process and in the selection of the final proposed design. I have held the safety of the public to be paramount and have addressed this in the presented design wherever may be applicable.

[illegible]

M'hamed Alghali Joudary

Approved by the Supervisor

Yousra Chtouki

Dr. Chtouki Yousra

ACKNOWLEDGEMENTS

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxx.

CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT (ENGLISH)

- This paper presents an innovative approach aimed at facilitating communication for hearing-impaired individuals through the development of a mobile application. The app is designed to translate Arabic Sign Language (ArSL) into text, and speech into text, enabling a seamless two-way communication channel. Using Google's powerful MediaPipe tool, specifically its Hand Landmark Detection solution, the application meticulously extracts the coordinates of 21 hand landmarks from images within the RGB Arabic Alphabets Sign Language Dataset. This critical data is the foundation of the training of a Multilayered Perceptron model, which has achieved an accuracy rate of 99.993%. To cater to the constraints of mobile environments, the model was subsequently optimized into a TensorFlow Lite version, ensuring real-time, efficient translation capabilities and accessibility on mobile devices without the necessity for an internet connection.

- The development of this application is carried out using Java, a widely used programming language known for its robustness, versatility, and extensive support within Android Studio, Google's official integrated development environment (IDE) for Android app development. This choice of technology stack not only facilitates the development process but also enhances the application's performance, reliability, and compatibility across a wide range of Android devices. By integrating Java's robust features and Android Studio's comprehensive development tools, the project emphasizes creating a user-centric design that offers an intuitive interface and instant feedback. This ensures the app's accessibility and ease of use, catering effectively to the needs of its target audience in any Android environment. The aim is to bridge the communication gap experienced by hearing-impaired individuals, offering them a tool that empowers them to communicate freely and effectively in their daily lives.

Keywords: Mediapipe, Android, Multilayered Perceptron, Java

RESUME (FRENCH)

- Ce rapport présente une approche innovante visant à faciliter la communication des personnes malentendantes grâce au développement d'une application mobile. L'application est conçue pour traduire la langue des signes arabe (ArSL) en texte et la parole en texte, permettant ainsi un canal de communication bidirectionnel transparent. À l'aide du puissant outil MediaPipe de Google, en particulier de sa solution de détection des repères manuels, l'application extrait méticuleusement les coordonnées de 21 repères manuels à partir d'images contenues dans l'ensemble de données de langue des signes des alphabets arabes RVB. Ces données critiques constituent la base de la formation d'un modèle Multilayer Perceptron, qui a atteint un taux de précision de 99,993 %. Pour répondre aux contraintes des environnements mobiles, le modèle a ensuite été optimisé dans une version TensorFlow Lite, garantissant des capacités de traduction efficaces et en temps réel et une accessibilité sur les appareils mobiles sans avoir besoin d'une connexion Internet.
- Le développement de cette application est réalisé à l'aide de Java, un langage de programmation largement utilisé connu pour sa robustesse, sa polyvalence et sa prise en charge étendue au sein d'Android Studio, l'environnement de développement intégré (IDE) officiel de Google pour le développement d'applications Android. Ce choix de pile technologique facilite non seulement le processus de développement, mais améliore également les performances, la fiabilité et la compatibilité de l'application sur une large gamme d'appareils Android. En intégrant les fonctionnalités robustes de Java et les outils de développement complets d'Android Studio, le projet met l'accent sur la création d'une conception centrée sur l'utilisateur qui offre une interface intuitive et un retour instantané. Cela garantit l'accessibilité et la facilité d'utilisation de l'application, répondant efficacement aux besoins de son public cible dans n'importe quel environnement Android. L'objectif est de combler le déficit de communication rencontré par les personnes malentendantes, en leur offrant un outil qui leur permet de communiquer librement et efficacement dans leur vie quotidienne.

Mots clés: Mediapipe, Android, Multilayered Perceptron, Java

1. Introduction

In the age where we are surrounded by technology that redefines the ways we communicate and interact, the importance of inclusivity has now become an essential part of these development efforts.. This project’s goal is to produce an innovative mobile app that will break through the communication barriers dealt with by the community of hearing impaired individuals in Morocco and Arabic speakers as a whole. Using advanced machine learning algorithms along with the natural language processing technology the project is aimed at filling the gap between signed language and spoken language, this will bring effortless communication among the people with different linguistic and sensory background.

This project, involving TensorFlow for the creation of a machine learning model, MediaPipe for the detection of real-time hand gestures, TensorFlow Lite for the optimization for mobile devices, and Vosk API for offline Arabic language translation, aims to build a mobile application that tackles multiple facets of communication challenges. Our application's interface, built using Android Studio, is simple and easy to operate, which is a guarantee that anyone can use it without prior familiarity with digital tools.

This report underlines the process from the idea to the realization of the project comprising the methodologies, planning, the activities, and the expected outcomes. This is an illustration of the ability of technology to go beyond innovation, and to use it responsibly to promote more inclusive social interactions and communications.

2. Problem Statement

In Morocco, a significant segment of our global population, particularly the hearing-impaired community, face substantial communication barriers. The situation is further complicated for Arabic speakers, as the tools and technologies designed to bridge these communication gaps are often not available in Arabic. This discrepancy not only isolates this demographic but also limits their access to education, employment, and social participation, thereby hindering their ability to contribute to society fully. This project aims to address these challenges by developing a mobile application that translates Arabic sign language into written text and converts spoken Arabic into text, providing a two-way bridge to facilitate more inclusive and effective communication.

3. Project Specifications

The project aims to design and implement a comprehensive mobile application that serves as a communication aid for both the hearing-impaired community and Arabic speakers. The application will harness advanced technologies, including machine learning and natural language processing, to accurately recognize and translate sign language into Arabic text and vice versa. Specifically, the project will incorporate the following specifications:

1. **Machine Learning Model**: Develop a robust machine learning model using TensorFlow, trained on a dataset of Arabic sign language gestures. This model will be the core of the application, enabling the recognition and translation of sign language into text in real-time.
2. **MediaPipe for Gesture Detection**: Utilize MediaPipe, a versatile framework for building applied machine learning pipelines, for the real-time detection of hand gestures. This technology will allow the application to capture and interpret sign language accurately.
3. **TensorFlow Lite for Mobile Optimization**: Convert the TensorFlow model to TensorFlow Lite to ensure better performance on Android devices.
4. **Vosk API Integration for Offline Conversion**: Integrate the Vosk API to enable offline speech-to-text conversion in Arabic. This feature will ensure that users can convert spoken language into text without the need for an internet connection.
5. **User-friendly Interface**: Design and develop a user-friendly interface using Android Studio, focusing on ease of navigation and accessibility. The interface will cater to users with varying levels of digital knowhow.
6. **Data Preprocessing and Augmentation**: Apply data preprocessing and augmentation techniques to enhance the machine learning model's training process.

7. **Comprehensive Testing and Validation**: Conduct thorough testing and validation of the application to ensure its effectiveness in real-world scenarios. This will include beta testing with potential users from the target demographic to gather feedback and make necessary adjustments.

4. Steeple Analysis

Social Factors:

This app could significantly enhance social inclusion for hearing-impaired individuals in Morocco, promoting greater societal awareness and integration of sign language in mainstream mediums. This would also enable the deaf community in Morocco to have access to better personal, educational, and professional opportunities that they otherwise wouldn't be able to reach.

Technological Factors:

According to the International Trade Administration (ITA), Morocco's mobile phone user penetration rate is above 100 percent (49.42 million mobile phone subscribers, out of a population estimated at 36 million people), we can see that the technological infrastructure does support the adoption of such an application. However, there needs to be thorough testing in across a range of devices to ensure that the application works properly in every one of them.

Economic Factors:

Economic disparities in the users of this app could affect its accessibility. This app will be free to own to ensure accessibility across different socioeconomic groups.

Environmental Factors:

This project helps in promoting digital solutions in various fields which reduces the need for physical materials and travel, which contributes to sustainability.

Political Factors:

In 2019, various governmental campaigns were initiated to help better standardize sign language in Morocco, as well as, better integrate disabled people into society. These initiatives align with this project which can garner support and facilitate the widespread use of our app.

Legal Factors:

We developed the app with Morocco's Data Protection laws in mind, and specifically: Article 23 of the Data Protection Law in Morocco provides that an organization is required to implement all technical and organizational measures to protect personal data in order to prevent it being damaged, altered or used by a third party who is not authorized to have access, as well as to protect it against any form of illicit processing.

Ethical Factors:

This app strives to be as inclusive as possible considering the various needs of its users, including those with varying levels of hearing impairment.

5. Engineering Standards

Relevant engineering standards for this project include:

ISO/IEC 25023:2016 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality: This standard provides as a guideline and criteria for assessing the quality of software and systems. It includes functional, performance, reliability, and usability factors. Quality assurance is a must for making sure that our application gets a high level of performance and provides a tool that is safe and usable to the users. By adhering to ISO/IEC 25023:2016 the project promises to create an app which is not only functional but also user-friendly, accessible, and reliable. Implementation of the quality indicators and evaluation in line with the standard will be a stepping stone in the identification of the areas of improvement so as to to confirm that the application achieve the goals of improving communication for the hearing impaired community.

IEEE 29148-2018 - IEEE Standard for Software and Systems Requirements Engineering: This standard identifies the processes and methods for managing systems and software requirements; that is, identifying the requirements is worthwhile as it gives applications that meet the users' needs and expectations a good foundation. It stresses the issue that solid, fully achieved and unambiguous requirements are the foundation for successful software development. The project relies on the implementation of IEEE 29148-2018 norms which implies a methodical way to retrieve and describe the requirements for the sign language translation app. This requires knowing what the community of hearing-impaired people in Morocco needs specifically, the technical necessities for a speech-to-text conversion, and how well the app should be usable for it to be accessible. The adherence to the standard enables us to generate a very detailed requirements specification which may be used as a kind of a blueprint for the software development, testing, and validation processes that helps us to design an application that fully caters for both user expectations and, of course, technical excellence.

6. Logic Model Framework

6.1. Target Population

Initially, this project aims to target every Moroccan individual. The reason for this is that hearing-impaired people in Morocco are usually not offered adequate opportunities to strive and build a career. This is further aggravated by the lack of resources and support for sign language among the general population, which further increases the gap that is felt by the deaf community from the rest of society.

6.2. Underlying Assumptions

To ensure the effective use of the functionality provided by the application, we assume that the hearing-impaired people who use the application know the Arabic alphabet in sign language as well as the fact that they can read Arabic, since the app will also convert speech to Arabic text. We also assume that the user has a functioning Android phone, where the version is anything above 8.0 (Oreo).

6.3. Resources/Challenges

Some of the resources used for this capstone project include:

RGB Arabic Alphabets Sign Language Dataset: This publicly available dataset in Kaggle comprises 7,857 fully labeled RGB images of Arabic sign language alphabets. The content available in this dataset has been collected from more than 200 participants with various settings such as lighting, background, image orientation, image size, and image resolution.

MediaPipe Documentation: The team behind the Open Source MediaPipe project compiled a very comprehensive documentation on the use of their solutions in various technologies which enabled us to smoothly implement it in this project and go to them for help when needed on their GitHub issues page.

6.4. Activities

The development process of our mobile application, which aims to translate sign language to text and provide offline speech-to-text conversion, went through several steps. Below is a detailed overview of the activities involved in the app development process:

1. **Collecting and Preprocessing Data**: Gathering RGB images of Arabic alphabet sign language gestures for model training then utilizing MediaPipe for image data preprocessing to extract hand gesture keypoints.
2. **Model Development and Optimization**: Creating a machine learning model with TensorFlow to recognize and translate sign language, then converting it to TensorFlow Lite format for efficient mobile deployment.
3. **App Integration**: Implementing MediaPipe and TensorFlow Lite in the Android app for real-time translation and gesture recognition.
4. **Vosk API Integration**: Incorporating Vosk API into the app for offline Arabic speech-to-text conversion.
5. **Interface Design**: Developing a user-friendly app interface with Android Studio for easy navigation and accessibility.

6.5. Outputs of the Project

The primary output of this project is the successful deployment of an Android mobile application capable of translating Arabic sign language into text in real-time and converting speech into text offline. The app features a simple and user-friendly interface designed in Android Studio, ensuring ease of use. It integrates advanced machine learning models optimized with TensorFlow Lite for mobile efficiency and utilizing MediaPipe for accurate hand gesture detection. Furthermore, the incorporation of Vosk API facilitates reliable offline Arabic speech-to-text conversion. Through this app, we are hoping to deliver a comprehensive communication tool designed to enhance accessibility for hearing-impaired individuals and Arabic speakers.

6.6. Outcomes

The outcomes of this project are anticipated to significantly enhance communication accessibility for the hearing-impaired community in Morocco and Arabic speakers in general. By translating sign language into text and converting speech to text offline, the app aims to bridge communication gaps, facilitating smoother interactions in educational, professional, and social settings. The expected long-term outcomes include improved social inclusion for hearing-impaired individuals, increased awareness and use of sign language among the wider public, and the promotion of equal opportunities in information access. We hope to contribute to the overarching goal of creating a more inclusive society where barriers to communication are minimized.

7. Literature Review

Sign Language Recognition Using Keypoints Through DTW

S. Pawar, K. Pandith, M. G. Rao, N. N. Chiplunkar and R. Samaga (2022) discussed in their paper the use of Dynamic Time Warping (DTW) for Sign Language Recognition (SLR) by analyzing video clips for sign language interpretation. The way their method works is by using Mediapipe to extract body and hand skeletal features extracted from RGB videos, focusing on capturing deterministic skeletal data for gesture identification. Their use of landmarks as holistic features for sign identification shows the system's capability to achieve accurate SLR through visual data alone, making it a promising foundation for a project such as mine where a mobile application is aimed at translating sign language.

Dynamic Sign Language Translator

S. Chandra, Venkatarangan MJ and Jyothi TN (2022) discussed further applications of MediaPipe for coordinates extraction in the field of automated sign language translation. This study highlights the advantages of deep learning, particularly Convolutional Neural Networks (CNNs), in enhancing feature extraction and reliable modeling for SLR. By combining MediaPipe with Ensemble and One-Shot techniques, the research demonstrates the potential to achieve high accuracy in translating dynamic sign language phrases, effectively showing the feasibility of developing an efficient mobile translation application.

Costa Rican Sign Language Recognition using MediaPipe

J. Zamora-Mora and M. Chacón-Rivas (2022) offer a focused study on the application of MediaPipe for the recognition of Costa Rican Sign Language. By extracting hand landmarks and employing various image processing techniques, the study showcases how MediaPipe serves as a robust tool for detecting crucial gesture components. The research suggests that with appropriate preprocessing and model selection, applications can be developed to cater to specific sign languages, offering personalized translation services.

8. Methodology & Capstone Design

8.1. Methodology

8.1.1. Data Collection & Preprocessing

The effectiveness of our sign language translation model relies on the quality of the data we use to train it. This section outlines the steps taken to gather a good enough dataset to train our model.

The initial dataset we used was the **RGB Arabic Alphabets Sign Language Dataset**, which comprises 7,857 labeled images of the Arabic sign language alphabets. These images were taken from over 200 participants in different settings, such as lighting, background, image orientation, image size, and resolution.

We then used Mediapipe, specifically its hand tracking solution, to extract 21 landmarks from each image. These landmarks offer a comprehensive representation of the hand gestures by capturing the nuances associated with sign language.

These coordinates were then structured into a csv file for better analysis and formatting. Each row corresponds to an image, while the columns were dedicated to each of the 21 landmarks, with the final column being reserved for the label of the sign depicted in the image.

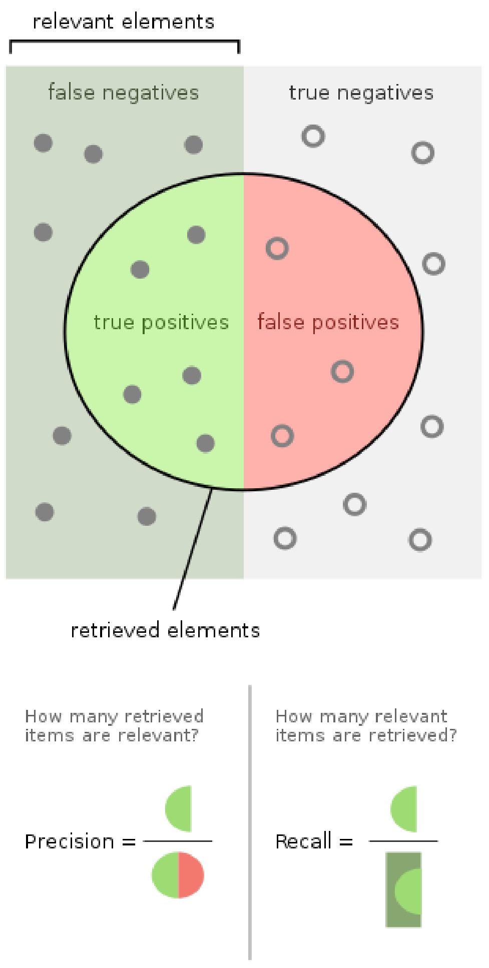
To address the challenge of having a limited dataset, we employed the Python library SMOTE (Synthetic Minority Over-Sampling Technique). This allowed us to artificially augment the dataset by generating synthetic data based on the rows already existing, especially for under-represented letters in the original set. This allowed the model later on to generalize over a broader spectrum of signs.

8.1.2. Analysis Techniques

To evaluate and analyze our model, we employed a variety of metrics, including precision, recall, f1-score, and support. Precision measures how often our model correctly predicts the positive class, which lets us measure the proportion of true positive results. Recall measures how often our model correctly identifies positive instances from all the actual positive samples, which answers the question of whether our model can find all instances of the positive class. F1-score combines the precision and recall scores of the model. It computes how many times the model predicted correctly across the entire dataset. Support shows the number of actual occurrences of each class in the dataset, which grounds our performance metrics in the context of our dataset's composition.

We then used a heatmap of the confusion matrix to better visualize the model's performance across the different classes.

A better representation of the difference between Precision and Recall can be seen here:



8.2. Capstone Design

8.2.1. Design Overview

This application, which was developed in Java for Android, is built with dual-sided functionality to help establish a two-way communication and bridge the communication gap for deaf and hard of hearing individuals.

On one side, it features a sign language recognition and translation module, that uses Mediapipe and a Multilayer Perceptron model to interpret Arabic sign language into text.

On the other, we have a speech-to-text conversion module, dedicated to translating spoken Arabic into text. This allows individuals who do not understand sign language to engage with those who primarily use it.

8.2.2. Software

Visual Studio Code with Jupyter Notebook Extension:

Visual Studio Code (VSCode) equipped with the Jupyter Notebook Extension served as the main Integrated Development Environment (IDE) used to clean and preprocess our data, as well as designing and training our model. VSCode's seamless integration of the Jupyter Notebook environment allowed for a very efficient coding experience, facilitating real-time testing and iteration of code snippets.

Android Studio

Android Studio was the main IDE used in the development of our mobile application. Since its specifically designed for Android development, Android Studio provides a large array of tools that helped streamline the coding, testing, and debugging processes. Its integrated Android emulator allowed for rapid testing across a variety of device configurations, ensuring that our application delivered a consistent user experience across different Android devices. Its seamless deployment capabilities to external mobile devices allowed for easy testing on real devices as well.

Libraries Used

NumPy: An imperative package for numerical computations in Python. NumPy was used to handle large, multi-dimensional arrays and matrices used in the project. It also offered an impressive collection of mathematical functions to operate on such structures.

Open Source Computer Vision (OpenCV): OpenCV was used for image processing tasks. Its functions allowed us to manipulate and prepare images for the dataset, including the application of various transformations and augmentations applied by Mediapipe.

Pandas: This library is crucial for data manipulation and analysis as it provides high-performance and easy-to-use data structures. Pandas made it simple to manage the dataset, as it helps load and preprocess data.

Scikit-Learn: Scikit-learn offered a wide range of tools to evaluate the performance of our machine learning model,. Its metrics module provided functions to measure the accuracy, precision, recall, and F1 score, enabling a thorough assessment of our model's effectiveness.

Mediapipe: Mediapipe provides a range of libraries and tools to help in various AI and

ML projects. These solutions are very customizable and easy to implement. The one we used was the **Hand Landmark Detection Solution**, which detects hands in images/videos/live feeds and applies 21 3d landmarks, each with their coordinates. This helped the model gather a good enough understanding of what each sign usually looks like.

TensorFlow and TensorFlow Lite: TensorFlow was used in the development and training of our Multilayer Perceptron model, using its flexible libraries and tools. We then used TensorFlow Lite to convert our model to a format compatible with mobile devices.

SMOTE (Synthetic Minority Over-Sampling Technique): This library helped deal with the imbalances in our dataset, as well as to artificially augment its content. By generating synthetic rows, we were able to provide a balanced and more representative dataset for our model.

Vosk API: Vosk is an offline open source speech recognition toolkit which allows us to convert Arabic speech into text for our app. The reason we opted for Vosk is to keep our application functional without the need for internet connectivity.

8.2.3. Models

This project uses a custom-built Multilayer Perceptron model to recognize sign language gestures from images. This model is integral to translating these gestures into corresponding textual output. The architecture of our model is designed to process the extracted features from hand landmarks and classify them into predefined sign language gestures.

Model Architecture

The model is constructed using TensorFlow's Keras API, known for its flexibility and easy-to-use features for building neural networks. The architecture consists of a sequential model with four dense layers, designed to process a flattened array of hand landmarks extracted from images.

- **Input Layer**: The model takes an input shape corresponding to the number of features extracted from each image. Given that we extract 21 hand landmarks, each with x, y, and z coordinates, the total number of features per image is 63 (21 landmarks * 3 coordinates).
- **Hidden Layers**: Following the input layer are three dense layers, each consisting of 64 neurons and employing the ReLU (Rectified Linear Unit) activation function. The ReLU function is chosen for its efficiency and ability to prevent vanishing gradient problems, thus ensuring a faster and more effective learning process. These layers are responsible for

learning the complex patterns in the data by transforming the input features into representations that the model can use for classification.

- **Output Layer**: The final layer is a dense layer with neurons equal to the number of classes (num_classes), which in this case is 31, corresponding to the 31 different sign language gestures we aim to recognize. The sigmoid activation function is used for multi-class classification, providing a probability distribution across all possible classes.

Compilation and Training

The model is compiled using the Adam optimizer, a popular choice for deep learning applications due to its adaptive learning rate capabilities, which help converge to the minimum loss faster. The learning rate is set to 0.005, indicating how much the model's weights should be updated during training. A lower learning rate requires more training epochs but can lead to better optimization.

For the loss function, we use Sparse Categorical Crossentropy, ideal for multi-class classification problems where each example belongs to exactly one class. It measures the difference between the distribution of the predictions and the true distribution, with the goal of minimizing this difference.

Training is conducted over 100 epochs with a batch size of 32, where an epoch represents a full pass through the entire training dataset, and the batch size specifies the number of samples to work through before updating the internal model parameters. A validation split of 20% is used to monitor the model's performance on unseen data, ensuring that we maintain a balance between bias and variance, preventing overfitting.

8.2.4. Algorithms

8.2.4.1. Sign Language Recognition Algorithm

Our sign language recognition application employs a real-time algorithm that leverages both MediaPipe for hand landmark detection and a TensorFlow Lite model for gesture classification. This section outlines the operational flow of our recognition algorithm, demonstrating how it integrates these technologies to interpret sign language gestures.

Real-time Landmark Detection with MediaPipe

The algorithm begins with a live video feed from the user's camera. As sign language is performed in front of the camera, MediaPipe is applied in real-time to detect and track hand landmarks. MediaPipe's hand tracking solution offers an approach to identify the positions of the hand's key points, providing a detailed map of hand gestures as they happen live.

Gesture Recognition with TensorFlow Lite

Once MediaPipe has detected that the hand landmarks have stabilized—meaning that the gesture was performed—the current landmarks are captured and processed as input for our TensorFlow Lite model. This step is pivotal: it translates the spatial information of hand landmarks into a specific sign language gesture.

The TensorFlow Lite model, optimized for efficiency and speed, infers the potential meaning behind the captured hand landmarks. The choice of TensorFlow Lite is important, ensuring that our application can perform complex machine learning inferences on mobile devices with limited computing resources. It also enables us to keep the application internet independent, as it does not require communication with a separate server to make the inferences, and instead does them on device.

Displaying Recognized Gestures

Upon making an inference, the application translates the model's output into the Arabic text of the recognized sign. This text is then displayed to the user in a dedicated box located beneath the camera footage. This immediate visual feedback allows users to see the translation of their sign language gestures in real-time, fostering an interactive and engaging user experience.

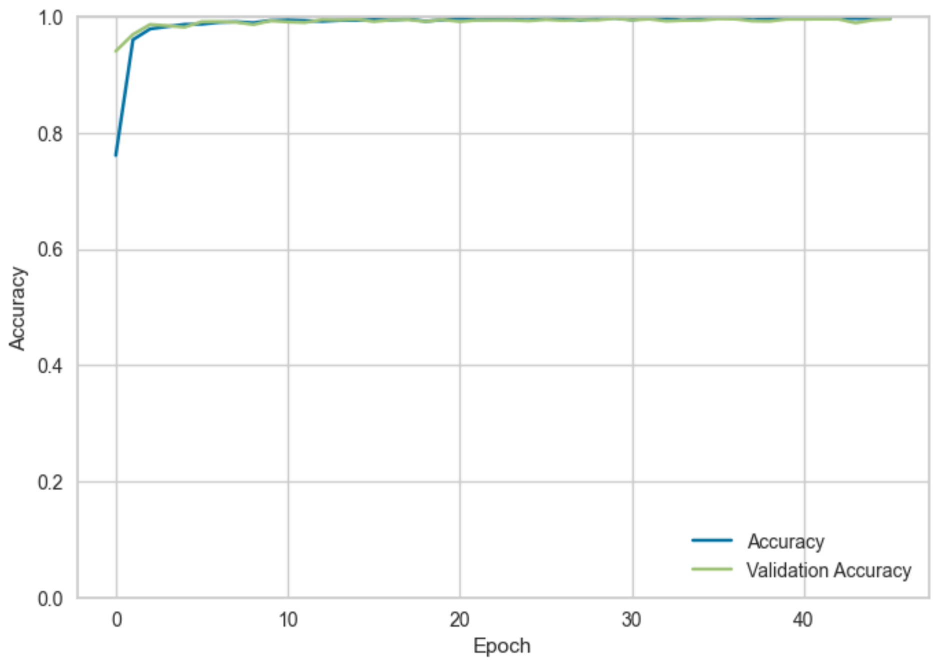
8.2.4.2. Speech-to-text Algorithm

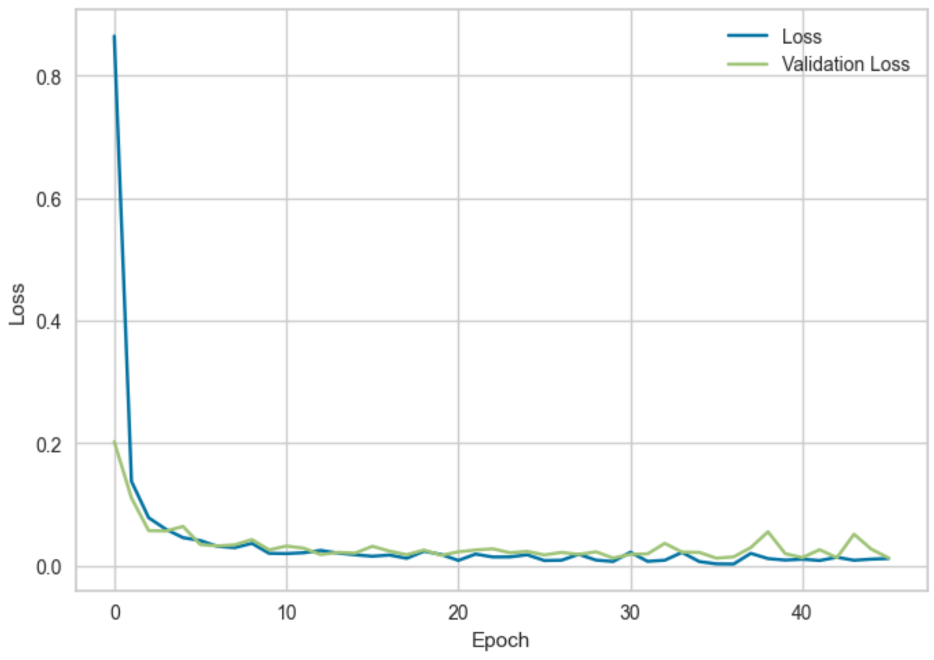
For the Speech-to-text aspect of our application, we opted to go for Vosk API, which is an offline speech recognition toolkit. This API supports Arabic, and since the models are imported locally, the latency is very minimal. The application enables the use of the microphone to capture live speech input, which is then processed through the Vosk speech recognition engine to convert spoken works into text. This process involves the extraction of audio features from the input, which Vosk's deep neural networks analyze to predict corresponding text. The resulting transcription is then displayed in the app's interface, providing users with a written record of their speech.

9. Data Presentation

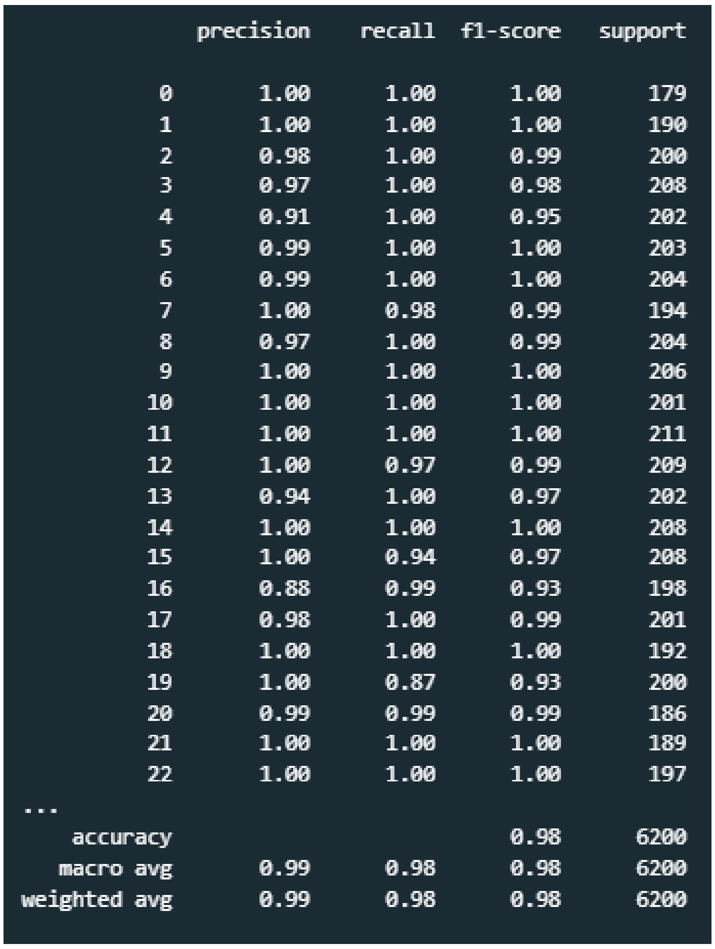
10. Simulations, Results, and Interpretation

I used a Python library called Smote to create synthetic data based on the original dataset I made from the Mediapipe landmarking tool. This allowed me to gather a set of test data that I could use to evaluate the performance of my model with unknown data. These are the results of that simulation:





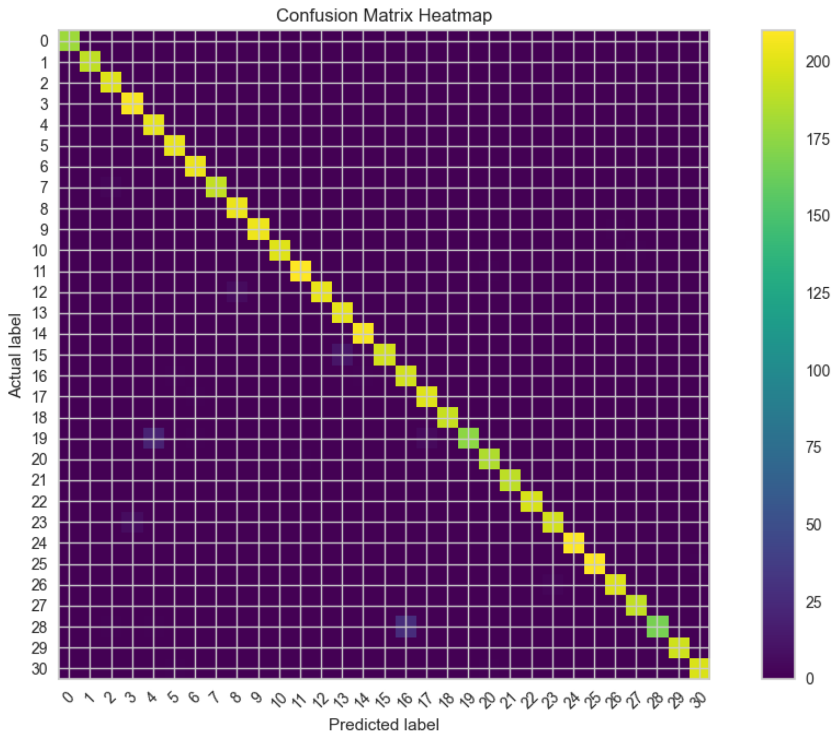
This is a classification report for the performance of the model using 4 different metrics.



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       179
           1       1.00      1.00      1.00       190
           2       0.98      1.00      0.99       200
           3       0.97      1.00      0.98       208
           4       0.91      1.00      0.95       202
           5       0.99      1.00      1.00       203
           6       0.99      1.00      1.00       204
           7       1.00      0.98      0.99       194
           8       0.97      1.00      0.99       204
           9       1.00      1.00      1.00       206
          10       1.00      1.00      1.00       201
          11       1.00      1.00      1.00       211
          12       1.00      0.97      0.99       209
          13       0.94      1.00      0.97       202
          14       1.00      1.00      1.00       208
          15       1.00      0.94      0.97       208
          16       0.88      0.99      0.93       198
          17       0.98      1.00      0.99       201
          18       1.00      1.00      1.00       192
          19       1.00      0.87      0.93       200
          20       0.99      0.99      0.99       186
          21       1.00      1.00      1.00       189
          22       1.00      1.00      1.00       197
...
    accuracy                           0.98      6200
   macro avg       0.99      0.98      0.98      6200
weighted avg       0.99      0.98      0.98      6200
```

And this is a visualization of the heatmap of the confusion matrix produced by the model:



11.Learning Strategies

12.Conclusion

References (APA)

Steeple tech https://www.trade.gov/country-commercial-guides/morocco-telecommunications

Precision/Recall https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

F1-score https://www.v7labs.com/blog/f1-score-guide

Please use

APA Citation Guide (7th edition) : In-Text Citation
https://columbiacollege-ca.libguides.com/c.php?g=713274&p=5082934

APPENDIX A : Code

APPENDIX B : CONFERENCE PROPOSAL X

APPENDIX C : XXXXX