PFE Code : I-258-23

Republic of Tunisia

Ministry of Higher Education and Scientific Research

Manouba University

Higher Institute of Multimedia Arts of Manouba

Graduation Project

Submitted for the purpose of obtaining a

National Engineer's Degree in Applied and Technological Sciences

Major : Digital Image and Virtual Reality (INREV)

By

Thouraya

ROMDHANI

Sign AI : Real-Time Sign Language Translation and Teaching Tool

Professional Supervisor: **Firas BARGUI** Talan Tunisie Consulting

Academic Supervisor: **Kalthoum REZGUI** ISAMM

Realized within: Talan Tunisia

Academic Year 2022 - 2023

Dedications

Dear God

You have remained my unwavering source of strength, guidance, and boundless blessings throughout my life's voyage.

To my beloved family

You have been the steadfast wellspring of my support, motivation, and the perpetual spark of inspiration throughout my existence. To you, my family—the bedrock of my life, my fortitude, my guiding star— I extend my heartfelt dedication on this journey, with profound gratitude and infinite love, every day and every night.

To my dearest friends

For your enduring friendship, unwavering support, and unceasing presence in my life, you have illuminated this path of education and personal growth, making it all the more enriching and enjoyable.

To all those who hold me dear. To all those whom I hold dear.

Thouraya ROMDHANI

Acknowledgements

I wish to convey my profound appreciation to all those who played a role, regardless of how small, in bringing my project to fruition and providing me with unwavering support during this enriching journey.

Above all, I want to express my deepest appreciation to Mrs. **Imen AYARI**, the director of the Talan Innovation Factory department, for generously affording me an extraordinary opportunity to embark on my internship journey within their team. Her unwavering faith in my capabilities and her priceless guidance have played a pivotal role in enriching my learning experience and acquiring priceless skills.

I extend my heartfelt gratitude to all the collaborators at **Innovation Factory** and **Talan Tunisia**, with special recognition for Mrs. **Racha FRIJI**, for their priceless support, technical and functional expertise, and their unwavering dedication, enthusiasm, and wholehearted engagement in our shared endeavors.

I also extend my sincere gratitude to Mrs. **Kalthoum REZGUI**, my supervisor, for her unwavering support and diligent oversight of my work. Her constructive feedback and guidance allowed me to delve deeper into my research and advance my thinking.

Furthermore, I would like to express my appreciation to Mr. **Firas BARGUI**, my internship mentor, for his continuous support, insightful advice, and ready availability throughout the project. His enthusiasm for innovation and his willingness to share his knowledge served as a perpetual wellspring of inspiration for me.

Contents

List of Figures

List of Tables

List of abbreviations

ML: Machine Learning

DL: Deep learning

TL: Transfer Learning

CNN: Convolutional Neural Network

DNN: Deep Neural Network

AI: Artificial Intelligence

PCA: Cascading Style Sheets

NLP: Natural language processing

NLU: Natural language understanding

NLG: Natural language generation

LLMS: Large language models

OCR: Optical character recognition

Crisp Dm: Cross Industry Standard Process for Data Mining

General Introduction

Sign language is a powerful means of communication that has been used by deaf and hard-of-hearing individuals for centuries. Unlike spoken language, sign language relies on visual cues and gestures to convey meaning. It is a complex and nuanced language that varies from country to country, with different grammatical rules and vocabulary.

For many deaf individuals, sign language is not only their primary mode of communication, but it is also a critical aspect of their cultural identity. Sign language allows deaf individuals to communicate effectively with their peers, express their thoughts and emotions, and participate fully in social, educational, and professional settings.

Despite its importance, sign language has often been overlooked or undervalued in mainstream society. Many people are unaware of the rich linguistic and cultural traditions of sign language and view it as a mere supplement to spoken language. This has led to a lack of access to education, employment, and social opportunities for deaf individuals, as well as a lack of recognition of their unique cultural identity.

However, in recent years, there has been a growing recognition of the importance of sign language as a means of communication and an essential part of deaf culture. According to the World Federation of the Deaf, there are more than 70 million deaf people worldwide. More than 80% of them live in developing countries. Collectively, they use more than 300 different sign languages [1]. Laws and policies have been enacted to ensure that deaf individuals have access to sign language interpreters in public settings, and there has been a push to recognize sign language as an official language in many countries. To make sign language more accessible and inclusive for everyone, the development of AI tools to translate sign language in real-time has been crucial. These tools can break down communication barriers by enabling real-time translation of sign language into spoken or written language, while also assisting in the learning of new sign languages, promoting cross-cultural communication, and understanding, and providing accessible and inclusive communication for the deaf and hard-of-hearing community to bridge the gap and promote equal opportunities for all.

Thus, our graduation project is part of this scope aiming to develop **Sign AI** an artificial

intelligence tool where information and analysis will be performed. It consists of data collection, data storage and pre-processing, and then utilizing the latest advanced machine learning models to create a deep learning based sign language interpretation tool in real-time.

This report is structured into five chapters that detail the implementation process of **Sign AI**. The first chapter aims to enhance understanding of the project's framework, encompassing a preliminary study, a critical analysis of existing solutions, and the proposed solution. The second chapter delves into an in-depth literature review covering the fundamental concepts of Deep Learning, as well as an examination of prior research work in this field. Meanwhile, the third chapter focuses on specifying requirements and project design. Following the exposition of the project's implementation steps and results achieved in the fourth chapter, the fifth chapter centers on presenting our solution by describing the hardware and software environments utilized.

Chapter 1

Scope of the Project

Contents

Introduction

In this chapter we will give an overview on the overall context of the project. Starting with the general scope of project including the host company presentation. Next, we will make a study on the existing system, criticize them, and then we propose our solution. Lastly, we compare various methodologies and describe the one used in our project.

1.1 General scope

This section provides an overview of the host company and its main activities.

1.1.1 Presentation of the host company : "Talan Tunisie Consulting"

"Talan group" is an international consulting group in innovation and transformation through technology created by Mehdi Houas, Eric Benamou and Philippe Cassoulat. Since 2002, "Talan" has been advising companies and administrations. The group supports them and implements their transformation and innovation projects in internationally.

With a presence on five continents, the group expects to reach a turnover of 600 million euros in 2023 for more than 6,000 consultants and aims to surpass sales by a billion euros by 2025. The group places innovation at the heart of its development and intervenes in areas related to the technological evolution of major groups.

In order to speed-up its development, "Talan Group" launched in 2008 "Talan Tunisia" as a Near-Shore Development Center in Tunisia, bringing together today over 500 engineers from the development of new technologies from the most prestigious Tunisian and European engineering schools, working for the largest European customers. [2].

1.1.2 Activities and areas of expertise

"Talan" concentrates on the following areas of expertise:

- Finance,
- Insurance,
- Energy&.Environment,
- Public sector,
- Telecom,

- Transport&.Logstic,

1.1.3 Talan Innovation Factory

Our end of studies internship was in the department "Talan innovation factory", Which is the R&.D and innovation department of "Talan Tunisia".

"Talan Innovation Factory" has been active since 2017. It explores and provides expertise on disruptive technologies such as "Blockchain", "Artificial Intelligence", "Internet of Things", "Data science", "Metaverse" and "Web3". [2].

1.1.4 Bootcamp

"Boot camp training is a way to train developers and other technical professionals to acquire deep skills in new technologies in a short period of time"[3].

At "Talan Innovation Factory", there is a strong belief that investing in their interns is crucial for creating a capable and competent workforce. To achieve this goal, they have developed a comprehensive 5-week Bootcamp program that is designed to provide their interns with the necessary knowledge, skills, and practical experience required for success in their future careers. The Bootcamp covers a broad range of essential topics.

At the end of each week, the interns are required to make a presentation on the topics covered and implement a simple use case.

- Week One: «AI anchors (bases)» :
  - AI projects methodologies (CRISP-DM, Kanban, etc).
  - Data collection : data sources, data labeling, web scraping, etc
  - Exploratory data analysis.
  - Data preprocessing: techniques to handle imbalance or incomplete datasets, dimensionality reduction(e.g PCA, LDA), Feature selection(e.g Chi-square,...), image preprocessing techniques.
  - Regulations in AI: RGDP (data privacy, regulation, etc.), other regulations worldwide.
  - AI and ethical challenges.
  - Different roles in AI (Machine learning engineer, data scientist, data engineer, etc..).
- Week Two: «Benchmarking of Machine Learning algorithms (architecture, use cases and tools) and Deep learning algorithms»:

  - Learning algorithms categories.
  - Evaluation metrics (Accuracy, Precision, Recall, F1 score, IoU (Intersection over Union), etc).
  - How Deep Neral Networks DNN learn: backpropagation vs forward-forward.

- Week Three: «NLP (NLU+NLG)» :
  - NLP techniques.
  - Benchmark of conversational interfaces and architectures.
  - LLMs and use cases (GPT-3, chatGPT, BERT, BARD, RoBERTa, ALBERTA ,etc).

- Week Four: « Computer Vision » :
  - Image pre-processing techniques.
  - Image analysis techniques.
  - Optical character recognition (OCR) .
  - Object detection and object tracking techniques (Yolo, Deepsort, etc).

- Week Five: « OpenAI new realises ChatGPT and Whisper APIs »
  - Domain of usage.
  - Benchmark of AI language models.

This approach ensures that the interns not only learn the theoretical concepts but also put them into practice, which helps to reinforce their understanding of the material. Overall, the «Talan Innovation Factory» Bootcamp is an effective and efficient way to help interns develop the skills they need to thrive in their future careers.

1.2 General context of the project

In this section, we give a global view on our project.

1.2.1 Project scope

While sign language is a valuable tool for deaf individuals to communicate, they may still face several challenges in their daily lives. One problem is the lack of awareness and recognition of sign language by the public, which can lead to misunderstandings and discrimination.

Another challenge is the limited availability of sign language interpreters in various settings, such as in hospitals, courts, or educational institutions. Without access to an interpreter, deaf individuals may struggle to communicate effectively, which can lead to misunderstandings, inadequate care, or even legal issues.

It is within this context our end-of-studies project named "sign language detection" give us the hand to maintain a conversation between the deaf or hard hearing community and hearing people. And give them the opportunity to learn sign language.

1.2.2 Study of the existing systems

Studying existing solutions is an important step in any project, whether the product already exists or needs to be developed. By studying existing solutions, potential limitations can be identified, effectiveness can be analyzed, opportunities for improvement can be found, compatibility can be ensured, and past experiences can be learned from. If the product already exists, this can lead to more successful solutions that meet changing needs. If the product needs to be developed, studying existing solutions can identify best practices, common challenges, and potential solutions. Overall, studying existing solutions is critical for better outcomes and success in any project.

In our scenario, we've identified two solutions, namely 'Ace Asl' and 'Slait.' By amalgamating these two solutions into a single application, we could potentially create a solution that closely resembles our own.

- «Ace Asl» :Ace ASL is an innovative AI-powered American Sign Language (ASL) learning app that offers a dynamic and immersive educational experience. The core functionality relies on real-time sign recognition technology, facilitated by a camera, providing instant feedback on ASL signing gestures. This technology also enables seamless translation between ASL and English. The app features a diverse curriculum, including lessons on fingerspelling and numerals, and incorporates intuitive hand recognition for user-specific signs.

  The development of Ace ASL involved collaboration among linguists, native ASL users, and tech experts, ensuring linguistic precision and technical excellence. Additionally, post-lesson quizzes enable users to assess their progress rapidly. Impressively, Ace ASL's AI technology recognizes a vast array of over 100+ distinct signs, demonstrating its versatility. Notably, the app was crafted by both deaf and hearing developers, emphasizing inclusivity and accessibility.

Table 1.1: Advantages and disadvantages of «Ace Asl»

Advantages	Disadvantages
• Learn alphabet, numbers and digit. • Practice your skills. • Easy to use. • Free.	• Used for alphabet, numbers and digit only, no words or sentences.

- «Slait» : SLAIT is a real-time sign language translation app that employs AI technology to convert sign language gestures into text. It recognizes over 200 ASL gestures and simple sentences with 92% accuracy, using a trained neural network model. The app uses Google AI's MediaPipe algorithm to track hand, arm, and facial expressions. SLAIT is available on web, Android, and iOS platforms, aiming to break communication barriers between deaf and hearing individuals, allowing natural video communication.

  As of 2023, the app's current availability remains uncertain. While the company's website is active and offers a demo for specific services, there's no clear information on its status. Despite this uncertainty, SLAIT's potential to enhance communication for the deaf and hearing communities is evident, and further research may reveal its current status and impact.SLAIT was initiated in 2021 with its primary focus on sign language AI translation, hence its name. It was developed as a real-time video chat and translation tool capable of identifying prevalent sign language gestures, facilitating effective communication between ASL speakers and individuals unfamiliar with the language.

Table 1.2: Advantages and disadvantages of «Slait»

Advantages	Disadvantages
• Provide automatic translations from ASL to English. • Provide transcription from speech to text.	• Still under test. • not available for use.

1.2.3 Proposed solution

After conducting extensive research and identifying the limitations of current solutions available in the market, we have identified a critical problem that currently lacks a viable solution in the market. The inability to track sign language in real-time and translate it to text or speech is a significant challenge for the deaf and hard of hearing community. Additionally, there is a lack of solutions that can translate from speech or text and vice versa in real-time. Moreover, it's important to note that there is a glaring absence of solutions tailored to Arabic and Tunisian sign languages, further compounding the communication barriers faced by individuals in these communities.

To address these challenges, we propose a hybrid mobile application **"Sign AI"** that can translate conversations from sign language to text or speech and from speech to text in real-time in both language American English and Tunisian Arabic.

Our primary goal is to facilitate seamless communication between the deaf and hard of hearing community and hearing individuals.

Moreover, our application includes an educational space where users can learn new sign language, such as Arabic sign language. Our proposed solution aims to bridge the communication gap between the deaf and hearing community and provide an inclusive environment for all individuals.

1.3 Project Management Methodologies

Project management methodologies are important because they provide a structured approach to managing projects. A project management methodology is a framework that guides project managers and their teams through the process of planning, executing, monitoring, controlling, and closing a project.

1.3.1 CRISP-DM Methodology

CRISP-DM, which stands for Cross-Industry Standard Process for Data Mining, is an industry-proven way to guide your data mining efforts.

- As a methodology, it includes descriptions of the typical phases of a project, the tasks involved with each phase, and an explanation of the relationships between these tasks.

- As a process model, CRISP-DM provides an overview of the data mining life cycle.

This approach includes six steps, as shown in Figure 1.1. [4].

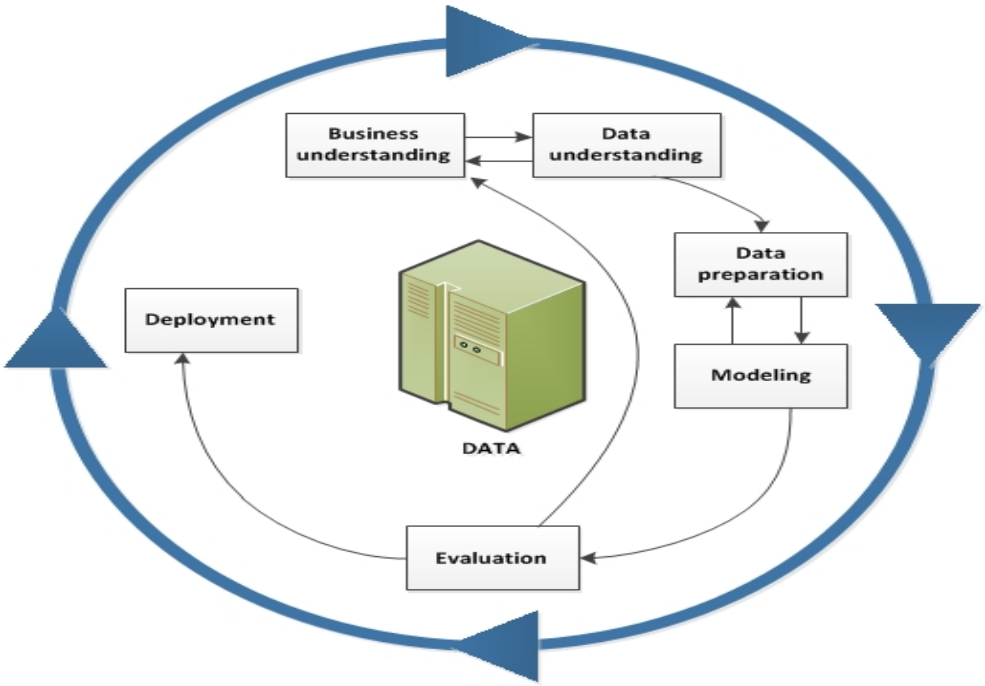


Figure 1.1: CRISP-DM Methodology

- Understanding the Activity: The first crucial step in any project involves understanding the requirements and establishing standards to achieve the objectives.

- Understanding Data: The next phase focuses on gathering, analyzing, and exploring data essential to the project.

- Data Preparation: During this stage, we select and clean the data for subsequent use in the analysis.

- Modeling: This phase defines the algorithm that will be applied to the prepared data.

- Evaluation: A rigorous evaluation compares the data models created with the task at hand and selects the most suitable model.

- Deployment: The final stage involves preparing the obtained results for presentation and submission to the client's decision-making process.

1.3.2 IBM Master Plan Methodology

The IBM Master Plan is a project management methodology developed by IBM for managing large-scale software development projects. The method involves breaking down the project into smaller, more manageable components, with each component having its own project plan and development team. The methodology emphasizes the use of iterative development, frequent

testing, and continuous integration to ensure that the project stays on track and that any issues are identified and resolved early in the development process.

This approach includes ten steps, as shown in Figure 1.2. [5].

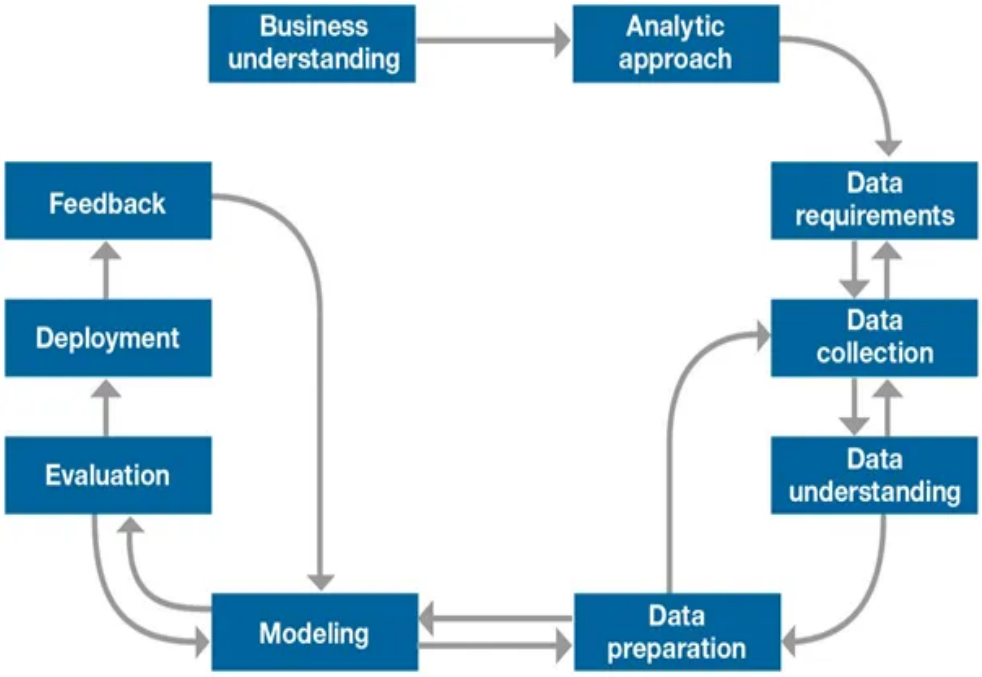


Figure 1.2: IBM Master Plan Methodology

- Business Understanding : this phase is for getting clarity on the problem allows you to determine which data will be used to answer the core question

- Analytic approach : this phase helps limit the algorithm(s) that will be used later.

- Data requirements : Once we chose our analytic approach, we identify the necessary data content, formats and sources for initial data collection

- Data collection : in this step, Database administrators (DBA) and programmers work together to extract data from various sources and then merge.

- Data understanding : this step is to understand the distribution of the variables

- Data preparation : this step is to prepare data using many operations such as addressing missing or invalid values and removing duplicates.

- Modeling : Models are constructed in this manner to finally produce the desired outcomes.

- Evaluation : It's the step in which we check if the model we have already generated answer the initial request or not.

- Deployment : Once we develop the model we deployed it in the production environment or in a test environment

- Feedback : to re-evaluate the model from the customers' point of view.

1.3.3 IBM Master Plan vs CRISP-DM Methodology

- Both methodologies emphasize the importance of understanding business objectives and aligning data science efforts with those goals.

- They share common stages such as data understanding, data preparation, modeling, evaluation, and deployment.

- CRISP-DM is more generic and can be applied to a wider range of industries and projects, while IBM's methodology is tailored to IBM's tools and services.

- The IBM methodology places a stronger focus on data collection and is closely associated with IBM's cloud-based data science offerings.

After a thorough comparison between the CRISP-DM methodology and IBM's approach, we chose CRISP-DM due to its adaptability across diverse industries and projects. Its generic framework provided a robust structure to align data science efforts with our project's goals, ensuring transparency and effectiveness in achieving our objectives.

1.3.4 Gantt Chart

Our project's Gantt chart encompasses various key stages that we plan efficiently, as depicted in Figure 1.3. Initially, we allocate a 6-week boot camp to ensure comprehensive training. Following that, we dedicate 3 weeks to documentation to establish a solid foundation and a shared understanding of the project. Requirement specification and design are critical stages, requiring 3 weeks to clarify requirements, create a robust architecture, and provide a clear vision for our project. We allocate 3 weeks for data collection as it involves research, selection, and data preparation. Data processing is a critical step that requires meticulous attention, and we allocate 2 weeks for this task. Then, we dedicate 4 weeks to modeling, during which we develop accurate and reliable models. The implementation of the mobile application is a crucial step, taking 3 weeks to ensure smooth development and meet project requirements. Finally, we allocate 3 weeks for testing and integration to ensure that all features are properly integrated, and the application is ready for delivery. This Gantt chart is thoughtfully designed to optimize our scheduling and ensure a consistent workflow throughout the project

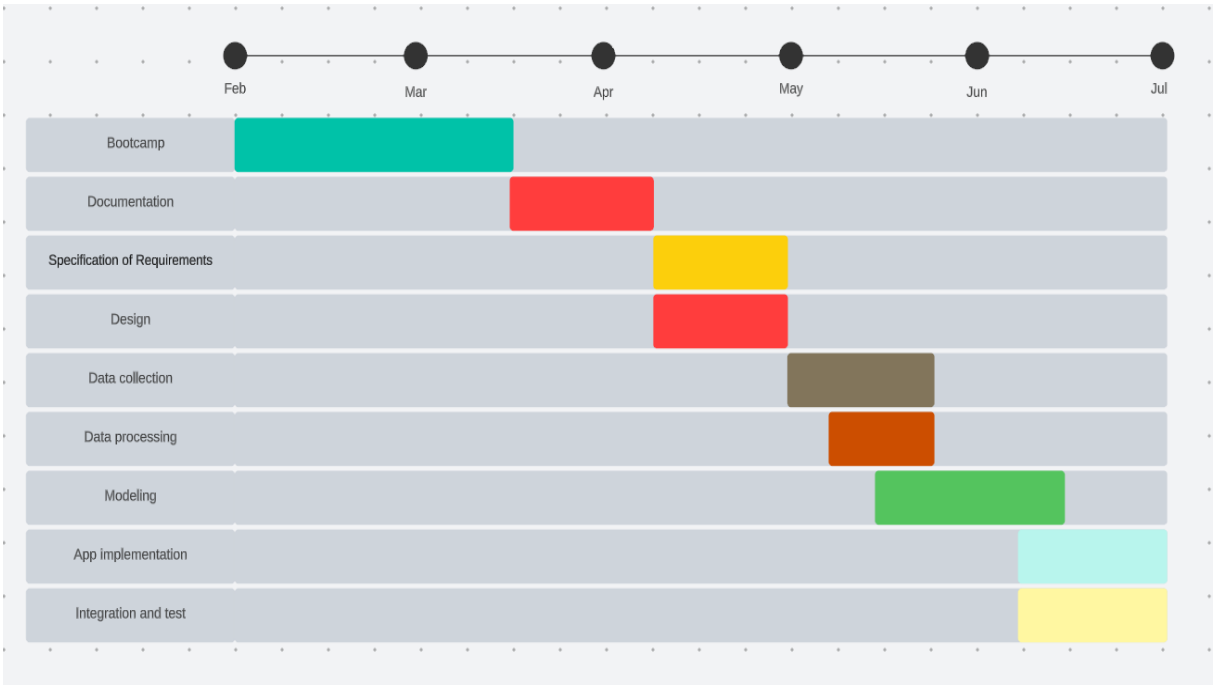


Figure 1.3: Gantt Chart

Conclusion

Throughout this chapter, we introduced Talan Tunisia, our distinguished hosting company, and proceeded to outline the issues we aim to address. Our goals were then established, and we have decided upon an appropriate project management methodology to follow.

Chapter 2

Key concepts of AI and Computer Vision

Contents

Introduction

The purpose of this chapter is to provide a comprehensive understanding of the fundamental aspects of AI techniques and Computer Vision used in our project. We'll cover topics like convolutional neural networks (CNNs), transfer learning, and Performance evaluation to give you a clear picture.

2.1 Computer vision

2.1.1 What is computer vision?

"Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand."[6].

2.1.2 How does computer vision work?

Computer vision operates by imitating how our brains understand visuals. It requires plenty of data and repeatedly studies the data until it can spot differences and finally identify images.

These accomplishment rely on two key tools: a special kind of machine learning learning called deep learning, and a convolutional neural network (CNN).

- Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will "look" at the data and teach itself to tell one image from another.

- A CNN helps a machine learning or deep learning model "look" by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions and makes predictions about what it is "seeing."

2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models designed for processing structured grid data, like images and videos. They're particularly powerful for tasks involving image analysis and recognition due to their ability to capture spatial hierarchies and local patterns.

CNNs have revolutionized computer vision tasks, achieving state-of-the-art performance in image and video classification, segmentation, object detection, and more.

At their core, CNNs employ a hierarchical structure comprising convolutional layers, pooling layers, and fully connected layers.

- Convolutional layers are responsible for detecting features like edges, textures, and shapes through the application of filters.
- Pooling layers subsequently downsample the detected features, reducing computational complexity while preserving essential information.
- Fully connected layers then interpret these extracted features for final classification or prediction.

Figure 2.1 presents an illustration of the CNN architecture.[7]

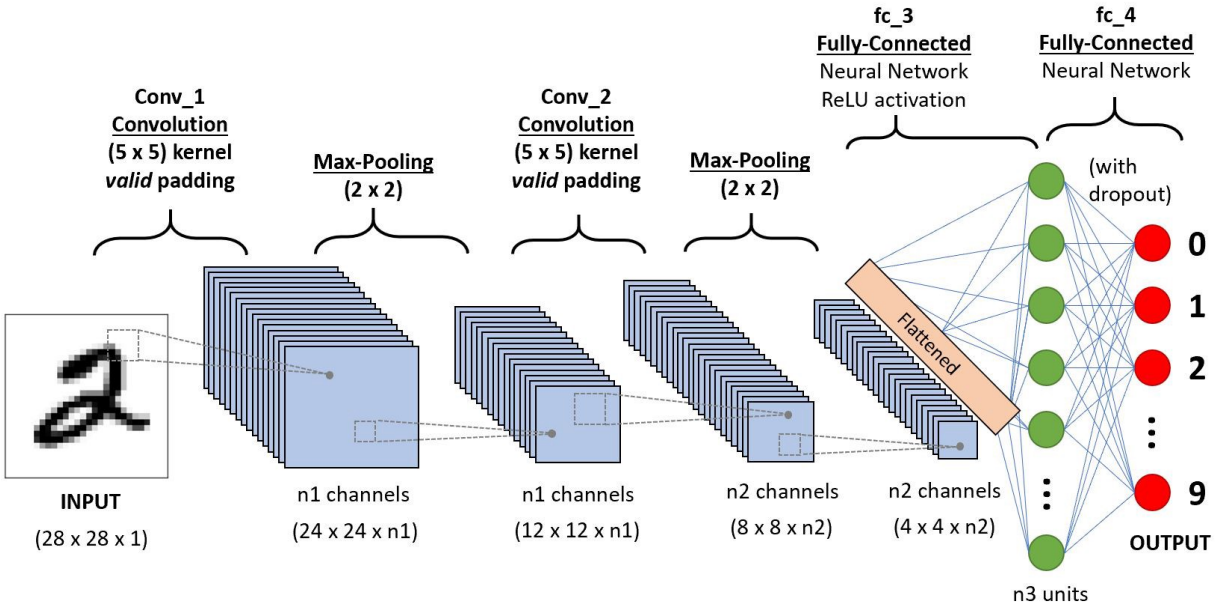


Figure 2.1: CNN architecture

2.3 Image Classification

The concept of image classification revolves around discerning the appropriate class for a given input image. This is accomplished by evaluating the probability or likelihood of the image aligning with a particular predefined category. In simpler terms, it entails assigning a single label to an input image from a predefined collection of categories (refer to the example provided in the above Figure 2.1).

Furthermore, in the specific framework of CNNs, a convolutional layer is employed to handle data characterized by two dimensions (Figure 2.2) [8], which inherently possess distinct height and width attributes. This layer's primary function is to effectively capture relevant features within images for subsequent classification tasks.

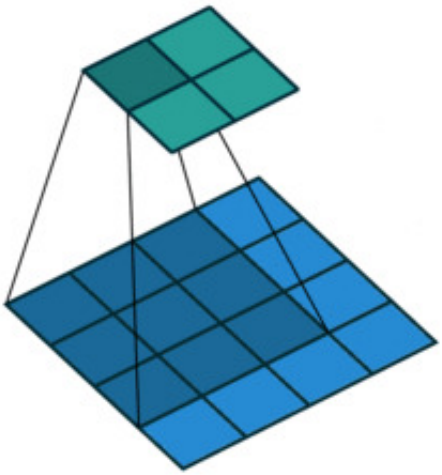


Figure 2.2: Convolution

2.4 Video Classification

Video classification stands as the formidable task of deciphering the essence conveyed by a video. It involves training a specialized model on a video dataset enriched with diverse categories, each signifying distinct actions or motions. The model's role is to analyze the individual frames of the video and produce the likelihood of each category being depicted within the video.

While both video classification and image classification models employ images as their input, tasked with forecasting the likelihood of these images falling into predefined categories, video classification goes the extra mile. It not only scrutinizes individual frames but also navigates the intricate web of spatio-temporal relationships between successive frames, thereby discerning the dynamic actions encapsulated within the video.

2.4.1 3D Convolution

In the preceding era, a majority of methodologies leaned upon 2D convolutional neural networks (CNNs) for the purpose of video classification. However, this approach encountered notable constraints. It struggled to encapsulate the intricate temporal interdependencies between frames, and additionally fell short in effectively grasping threedimensional attributes like motion.

In response to these challenges, the realm of research introduced a promising solution in the form of 3D convolutional neural networks (3D CNNs). While sharing similarities with their 2D counterparts, 3D CNNs were strategically engineered to surmount the temporal intricacies by processing sequences of frames rather than isolating individual frames. Furthermore, they possess the intrinsic capability to decipher threedimensional characteristics inherent in video sequences, notably encompassing motion attributes, an aspect that 2D CNNs struggled to accomplish.

In 3D Convolution it takes time * height * width * channels inputs and produces channels outputs (assuming the number of input and output channels are the same. So a 3D convolution layer with a kernel size of (3 x 3 x 3) would need a weight-matrix with 27 * channels ** 2 entries.

Figure 2.3 presents an illustration of the 3D Convolution [9].

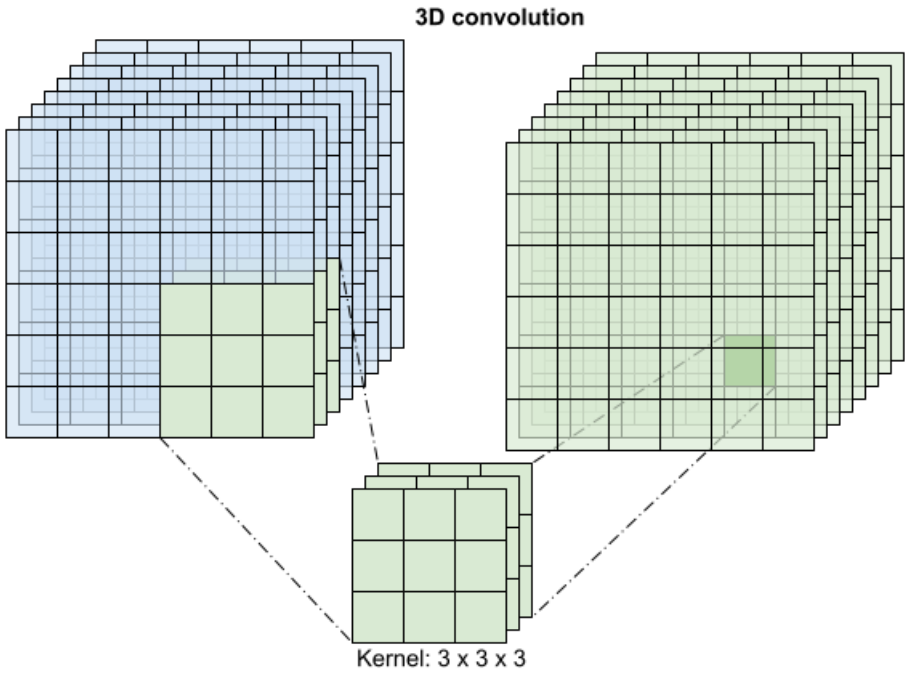


Figure 2.3: 3D Convolution

2.4.2 (2+1)D Convolution

Rather than relying on a single 3D convolution to process both time and space dimensions, an alternative approach was proposed, known as '(2+1)D' convolution. This novel technique processes spatial and temporal dimensions separately, as illustrated in the figure below depicting the factored spatial and temporal convolutions of a (2 + 1)D convolution.

The primary advantage of this approach lies in its parameter reduction. In the (2 + 1)D convolution, the spatial convolution operates on data with a shape of (1, width, height), while the temporal convolution operates on data with a shape of (time, 1, 1). For instance, a (2 + 1)D convolution with a kernel size of (3 x 3 x 3) would require weight matrices of size (9 * channels2) + (3 * channels2) – less than half the size of weight matrices needed for a complete 3D convolution.

Figure 2.4 presents an illustration of the (2 + 1) D Convolution [9].

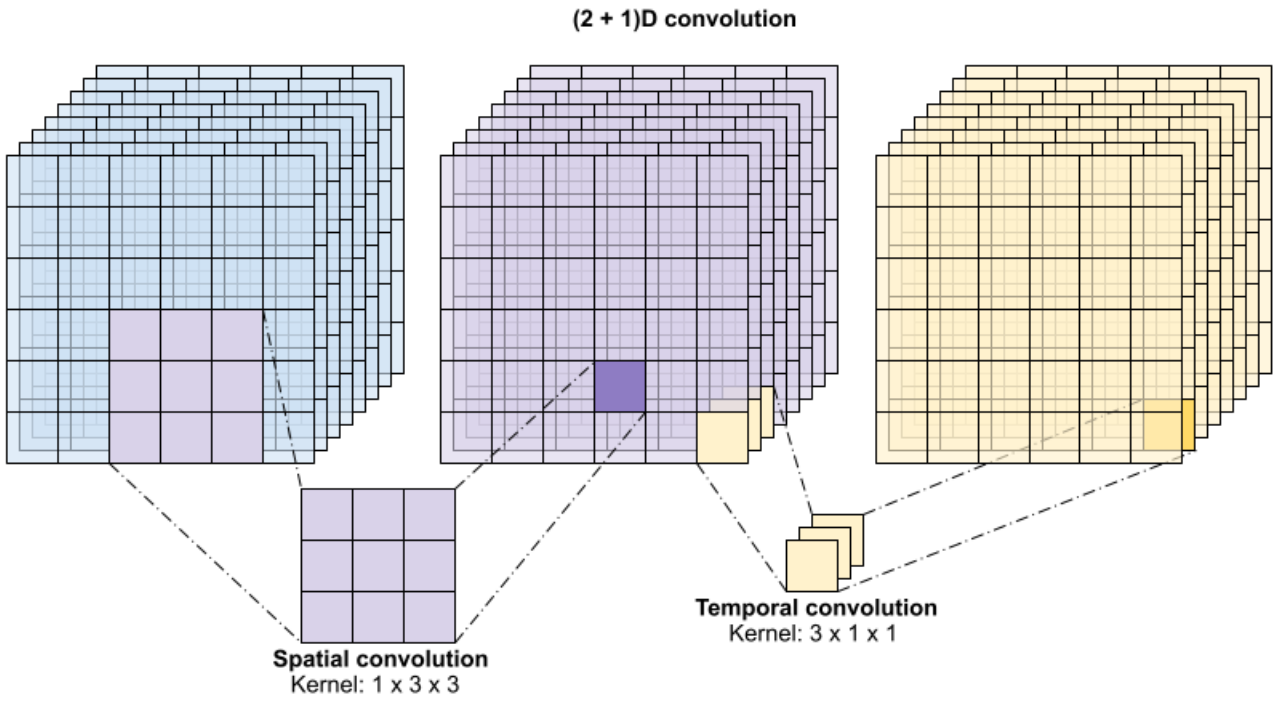


Figure 2.4: (2 + 1)D Convolution

2.5 Transfer Learning

The training process necessitates substantial computational capacity, time, and resources. In numerous instances, previously trained models can offer significant utility in unrelated scenarios, particularly when these tasks share commonalities or resemblances. This is where the concept of transfer learning becomes relevant. Transfer learning is a machine learning technique which is the method of starting with a pre-trained model and training it for a new, related, problem domain. The pre-trained network serves as transferred knowledge to be applied in another domain. But there are numerous options that can be used, including feature transfer and fine-tuning (which depend upon the similarity of the problems at hand), in addition to freezing certain layers of the network and retraining others. Transfer learning offers several advantages. It diminishes the necessity for extensive labeled data since the pre-trained model has already acquired valuable representations from a vast dataset. Additionally, it conserves substantial computational resources and time, as initiating model training from scratch can be highly computationally demanding. This technique proves especially advantageous in scenarios where there's a scarcity of labeled data for a particular task or when training resources are constrained. Transfer learning capitalizes on the notion that deep learning models can acquire generic features applicable to diverse tasks. Instead of commencing the training process anew, transfer learning enables the reuse of these acquired features, subsequently fine-tuning them for the new task. By harnessing the pre-trained model's grasp of common patterns and concepts, it expedites the training process and yields superior performance, even in situations with limited data. In summary, transfer learning is a methodology that empowers us to apply knowledge gleaned from one task to another closely related task, offering substantial advantages in various applications. Through the utilization of a pre-trained model's acquired features and their adjustment to a fresh task, we can attain enhanced performance, even when confronted with restricted data availability. This strategy has consistently demonstrated its substantial value across diverse domains, facilitating the creation of resilient and precise machine learning models.

In Figure 2.5 [10], you'll find a visual representation illustrating the distinctions between Traditional machine learning and transfer learning.

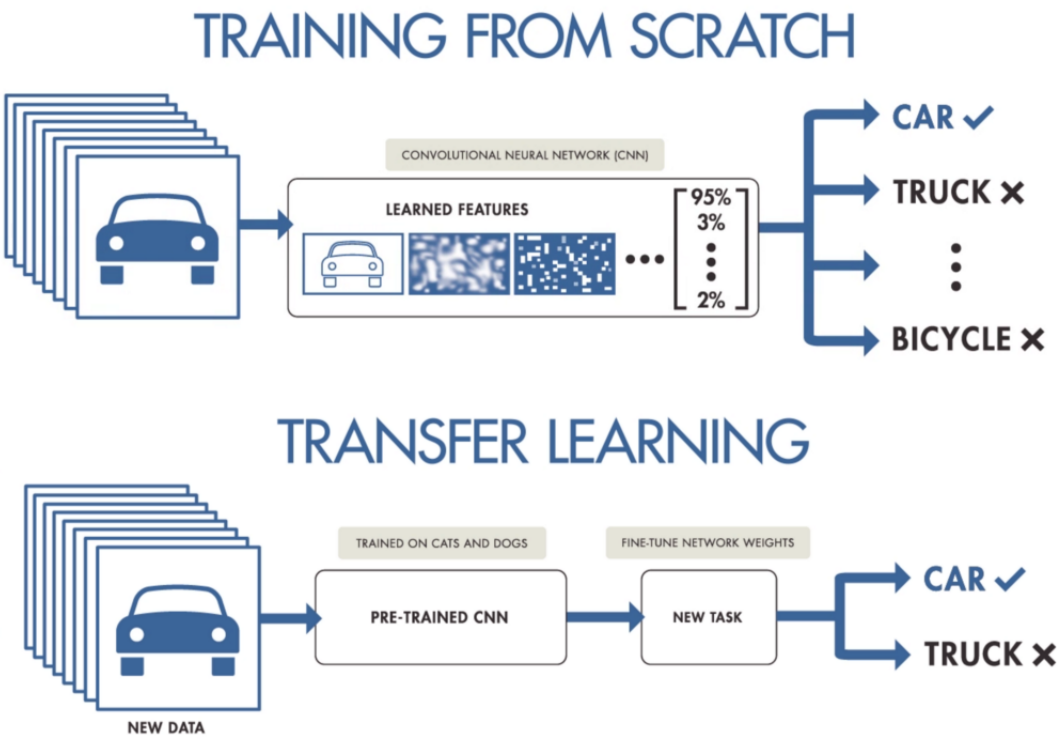


Figure 2.5: Traditional machine learning vs Transfer learning

2.6 ResNet-50

ResNet-50, short for Residual Network with 50 layers, is a deep convolutional neural network architecture that has had a significant impact on the field of computer vision and deep learning. It was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their seminal 2015 paper titled "Deep Residual Learning for Image Recognition."

At its core, ResNet-50 is a neural architecture that transcends conventional convolutional networks. Comprising precisely 50 layers, as the name suggests, each layer typically encompasses a blend of two to three convolutional layers. However, the distinguishing hallmark of ResNet designs is the ingenious incorporation of residual connections, often referred to as "skip connections." These connections orchestrate a direct and unimpeded flow of information from prior layers to their successors, offering an elegant solution to the perplexing vanishing gradient quandary that plagues deep neural networks. This predicament typically manifests when gradients dwindle to infinitesimal values during their passage through extensive neural layers, detrimentally hampering the learning process. ResNet-50's exceptional prowess lies in its aptitude for extracting high-level features through these residual connections, positioning it as an icon of innovation in the ever-evolving landscape of deep learning and computer vision.

Figure 2.6 shows an illustration of the Residual Connection [11].

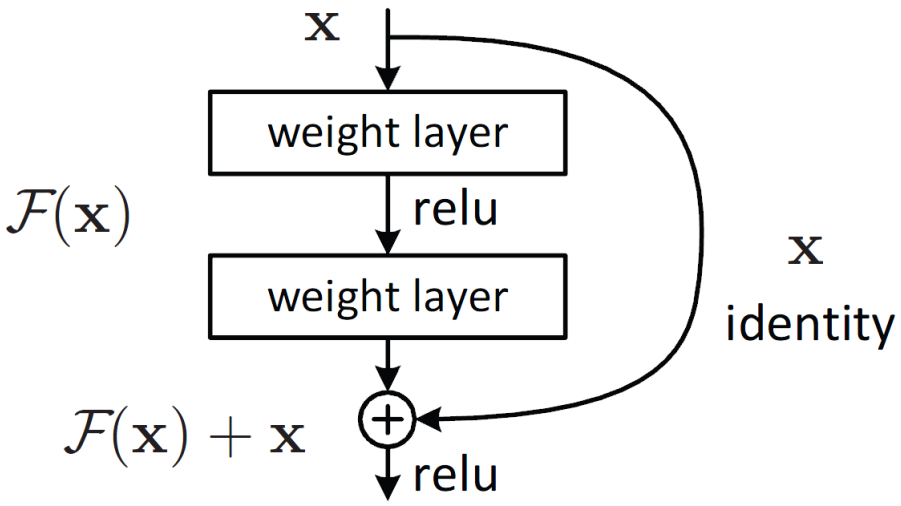
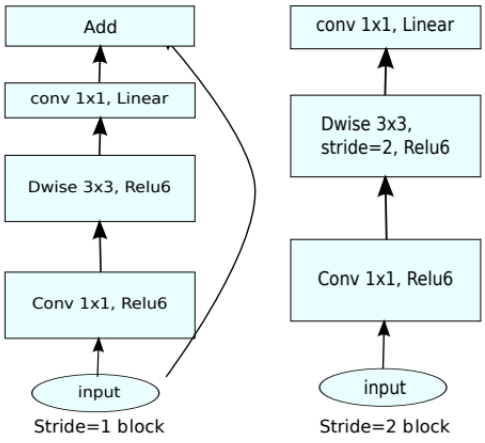


Figure 2.6: Residual Connection

2.7 MobileNet V2

MobileNetV2, a convolutional neural network design tailored for optimal performance on mobile devices, adopts an innovative approach. Its architecture centers around an inverted residual structure, where residual connections link the bottleneck layers. Within this design, the intermediate expansion layer employs lightweight depthwise convolutions to impart non-linearity by filtering features. In its entirety, MobileNetV2 comprises an initial layer featuring 32 filters followed by a sequence of 19 residual bottleneck layers. Figure 2.7 shows an illustration of the MobileNet V2 [12].



(d) Mobilenet V2

Figure 2.7: MobileNet V2

2.8 EfficientNet

EfficientNet is a groundbreaking family of convolutional neural network architectures that has made significant contributions to the field of deep learning, particularly in image recognition and computer vision tasks. Introduced by Tan and Le in their 2019 paper titled "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," EfficientNet aims to strike a balance between model size, accuracy, and computational efficiency.

The foundational structure of the EfficientNet-B0 network draws inspiration from MobileNetV2's inverted bottleneck residual blocks and incorporates squeeze-and-excitation blocks for enhanced performance.

Furthermore, EfficientNets exhibit remarkable transferability, consistently achieving state-of-the-art accuracy across various datasets, including CIFAR-100 (91.7%), Flowers (98.8%), and three additional transfer learning benchmarks. Remarkably, this superior performance is achieved with significantly fewer parameters, underscoring their efficiency.

Figure 2.8 shows an illustration of the EfficientNet B0 architecture [13].

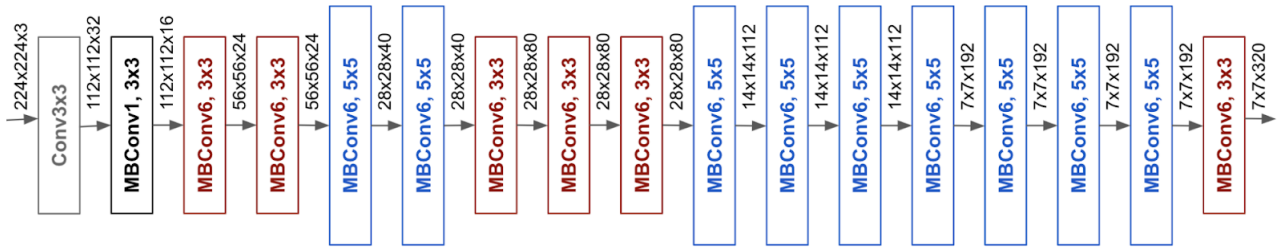


Figure 2.8: EfficientNet B0 architecture

2.9 DenseNet-201

DenseNet-201, short for Densely Connected Convolutional Networks with 201 layers, represents a pivotal advancement in the realm of computer vision and deep learning. This architectural masterpiece was introduced by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger in their influential paper titled "Densely Connected Convolutional Networks," published in 2017.

At its essence, DenseNet-201 is a neural network architecture that pushes the boundaries of conventional convolutional networks. True to its name, it boasts an impressive 201 layers, a substantial depth that far exceeds its predecessors. Each layer within this network is densely connected, facilitating a unique and innovative approach to feature extraction.

2.10 VGG16

VGG16, short for Visual Geometry Group 16, is a widely recognized and influential convolutional neural network architecture in the field of computer vision. It was developed by the Visual Geometry Group at the University of Oxford and presented in the paper titled "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Karen Simonyan and Andrew Zisserman, published in 2014.

VGG16 is characterized by its remarkable simplicity and uniformity in architecture. It is composed of 16 layers, which consist primarily of convolutional and pooling layers, followed by fully connected layers. The most common configuration of VGG16 uses 13 convolutional layers and 3 fully connected layers.

2.11 Performance evaluation

2.11.1 Overfitting/Underfitting

To diagnose the reasons behind poor model performance, it's essential to delve into how well the model aligns with the data. This understanding serves as a compass for guiding us in implementing corrective measures. By closely analyzing the prediction errors on both the training and evaluation datasets, we can discern whether the model is afflicted by underfitting or overfitting.

- **Underfitting**: Underfitting occurs when a model is too basic and can't understand the data's main patterns. It doesn't learn well from training data, so it's bad at both the

training data and new data. In pictures, it makes a lot of mistakes during training and with new data.

- **Overfitting**: Overrfitting occurs when a model is too complicated and learns all the little details, even the random ones, from the training data. This makes the model perform great on the training data but not so well on new data it hasn't seen before.

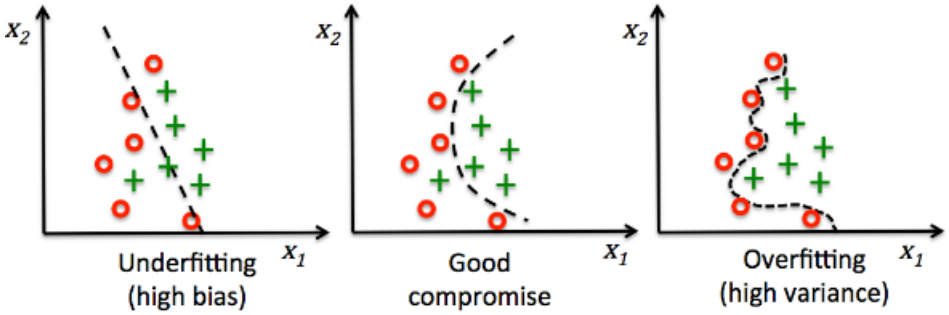


Figure 2.9: Schematic illustration of three models for classification: underfitting, appropriate fitting, and overfitting

Preventing underfitting and overfitting are essential aspects of training machine learning models. Here are techniques to address both underfitting and overfitting:

- **To Address Underfitting:**
  - Increase Model Complexity: Use a more complex model with a higher capacity, such as increasing the number of hidden layers and neurons in a neural network, or using more complex algorithms.
  - Feature Engineering: Improve the quality of your features by selecting more relevant ones or creating new features that better represent the underlying data patterns.
  - Hyperparameter Tuning: Experiment with different hyperparameters like learning rate, batch size, and regularization strength to fine-tune your model's performance. Grid search or random search can help you efficiently explore the hyperparameter space.
  - Reduce Regularization: If you're using regularization techniques like L1 or L2 regularization, consider reducing the strength of regularization to allow the model to fit the training data better.
  - More Training Data: Collect more data if possible, as having a larger dataset can help the model learn the underlying patterns better.
  - Early Stopping: Monitor your model's performance on a validation set during training and stop training when the performance plateaus or starts to degrade.
- **To Address Overfitting:**
  - Regularization: Introduce regularization techniques to constrain the model's complexity, such as L1 and L2 regularization. These penalties discourage extreme parameter values.
  - Dropout: Use dropout layers in neural networks to randomly deactivate neurons during training, which prevents over-reliance on specific neurons and encourages the network to generalize better.
  - Cross-Validation: Employ techniques like k-fold cross-validation to get a better estimate of your model's performance on unseen data. This helps identify if your model is overfitting to a particular dataset split.

- Feature Selection: Remove irrelevant or redundant features that might be causing overfitting. Feature selection can be done using techniques like feature importance scores or recursive feature elimination.
- Data Augmentation: Increase the effective size of your training dataset by applying data augmentation techniques, especially in computer vision tasks. This can help the model learn more robust features. Ensemble Learning: Combine predictions from multiple models (e.g., bagging, boosting, stacking) to reduce overfitting. Ensemble methods often lead to better generalization.
- Simpler Models: Consider using simpler model architectures, which are less prone to overfitting. For example, if you're using a deep neural network, try a shallower architecture.
- Pruning: In decision trees or random forests, pruning can remove branches of the tree that do not contribute significantly to improving accuracy on the validation set.
- Validation Set: Ensure that you have a separate validation set for monitoring model performance during training and making decisions about model complexity.
- Early Stopping: Implement early stopping based on the performance on the validation set to prevent the model from overfitting the training data.

2.11.2 Metrics and hyperparameters

Having explored the concepts of overfitting and underfitting, we now shift our focus to the crucial aspect of metrics and hyperparameters, which provide the means to objectively measure and fine-tune our models' performance.

Loss:

Loss, often referred to as the loss function or cost function, is a mathematical measure that quantifies how well a machine learning model is performing by calculating the difference between its predictions and the actual target values in the training data.

- Mean Squared Error (MSE):

  Formula: MSE $= 1\frac{}{n\sum_i (y_i - \hat{y}_i)^2}$

  Used in regression tasks to measure the average squared difference between the predicted values (ŷi) and the true target values (yi).

- Binary Cross-Entropy Loss (Log Loss):

Formula for a single example:BCE = -$[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$

Used in binary classification tasks, where yi is the true label (0 or 1) and ŷi is the predicted probability of belonging to class 1.

- Categorical Cross-Entropy Loss (Softmax Loss):

Formula for a single example: CCE = -$\sum_i y_i \log(\hat{y}_i)$

Used in multi-class classification tasks with more than two classes. It calculates the negative log-likelihood of the true class labels.

Accuracy:

Accuracy is a performance metric used mainly in classification tasks. It measures the proportion of correctly predicted instances out of the total number of instances in the dataset.

Formula for Accuracy:
Accuracy = (TP + TN) / (TP + TN + FP + FN)

- TP (True Positives) represents the number of positive instances correctly classified as positive.
- TN (True Negatives) represents the number of negative instances correctly classified as negative.
- FP (False Positives) represents the number of negative instances incorrectly classified as positive.
- FN (False Negatives) represents the number of positive instances incorrectly classified as negative.

Learning rate:

The learning rate is a hyperparameter (In machine learning, a hyperparameter is a predetermined parameter whose value is set prior to the start of the learning process, such as the number of network layers or the learning rate. In contrast, other parameters are determined and adjusted during the training process.) used in machine learning algorithms, and it plays a crucial role in shaping the size of the steps the algorithm takes when it fine-tunes the model's parameters throughout the training process. As the value decreases, our progress down the slope becomes more gradual, while conversely, with a higher value, we move down the slope more

rapidly.The connection between the learning rate and the model's weights can be expressed as follows:

$$\mathbf{w}_{i,j} = \mathbf{w}_{i,j} \ - \alpha \frac{dL}{\mathbf{w}_{i,j}}$$

α: represents the learning rate.

$\mathbf{w}_{i,j}$: corresponds to the model's weight.

L : stands for the model's loss function.

Conclusion

In this chapter, we have delved into the foundational concepts of AI techniques and Computer Vision, essential for our project's context. By exploring topics such as convolutional neural networks (CNNs), transfer learning, and performance evaluation, we have aimed to equip you with a solid foundation and a clear perspective on these crucial aspects. With this knowledge in hand, we are well-prepared to proceed to the practical application and implementation phases of our project.

Chapter 3

Analysis and Conceptual Study

Contents

Introduction

This chapter is dedicated to the conceptual examination of the project. We delve into a comprehensive analysis of our project's features in Section 3.1. Subsequently, we delve into the design phase, a pivotal step in the process, within the following section.

3.1 Specification of Requirements

In this section, we delve into two crucial components of our final project report. Firstly, we initiate by providing a definition of the primary actor within our application. Subsequently, we conduct an analysis of the functionalities and services offered by our system, taking into consideration the associated constraints.

3.1.1 Actors identification

In our project, we target individuals who are deaf or hard of hearing, as well as anyone interested in facilitating communication with this community. The only actor of our project is someone who may be a deaf individual, a person wishing to engage in conversations with the deaf, or someone simply curious about learning sign language. The main objective is to assist people in comprehending sign language conversations and fostering inclusive interactions between the deaf and hearing individuals.

3.1.2 Functional Requirements

The project encompasses the following core functionalities:

- User Registration and Authentication: Users can easily register for an account and log in securely.
- Real-Time Translation of American Sign Language Conversations: The application offers real-time translation capabilities for American Sign Language (ASL) conversations, bridging communication gaps efficiently.
- Real-Time Translation of Tunisian Sign Language Conversations: Additionally, the system extends its real-time translation features to encompass Tunisian Sign Language (TSL), broadening its accessibility and inclusivity.
- Arabic Sign Language Letter Instruction: Users can engage in an educational experience that involves learning Arabic Sign Language letters, contributing to their sign language proficiency.

- American Sign Language Letter Instruction: In a parallel educational feature, the application facilitates the learning of American Sign Language (ASL) letters, empowering users to communicate more effectively.

- American Sign Language Letter Classification: The application also offers a classification feature for American Sign Language (ASL) letters, aiding users in recognizing and using ASL letter signs accurately.

- Community-Driven Contributions: Users have the opportunity to actively participate in improving our application by contributing new words and signs in various sign languages, fostering a collaborative and continually evolving environment.

3.1.3 None functional Requirements

Our application must meet the following requirements:

- Accuracy: The solution must provide models that ensure a minimum accuracy of 70

- Availability: The solution must be available 24/7 to provide real-time assistance to customers.

- User-Friendliness: The user interface of the solution should be user-friendly, intuitive, and easy to use, allowing users to become proficient with it within approximately 5 minutes of training.

3.1.4 Use Case Diagram

The use case diagram is a visual representation that captures the dynamic aspect of a system. It is used to gather system requirements, both internal and external, in order to design the system. Use cases represent the system's functionalities, while actors are external entities that interact with the system.

Figure 3.1 presents the use case diagram of our application which illustrates the interactions between actors and the system's use cases. This diagram provides a clear visual representation of user interactions and functionalities. It showcases how users can create accounts, log in, select languages for real-time translation, initiate translations, learn sign language letters, assess their progress, and actively contribute to the application's improvement. This diagram serves as a comprehensive guide to the user experience, illustrating the versatility and richness of our application's features.

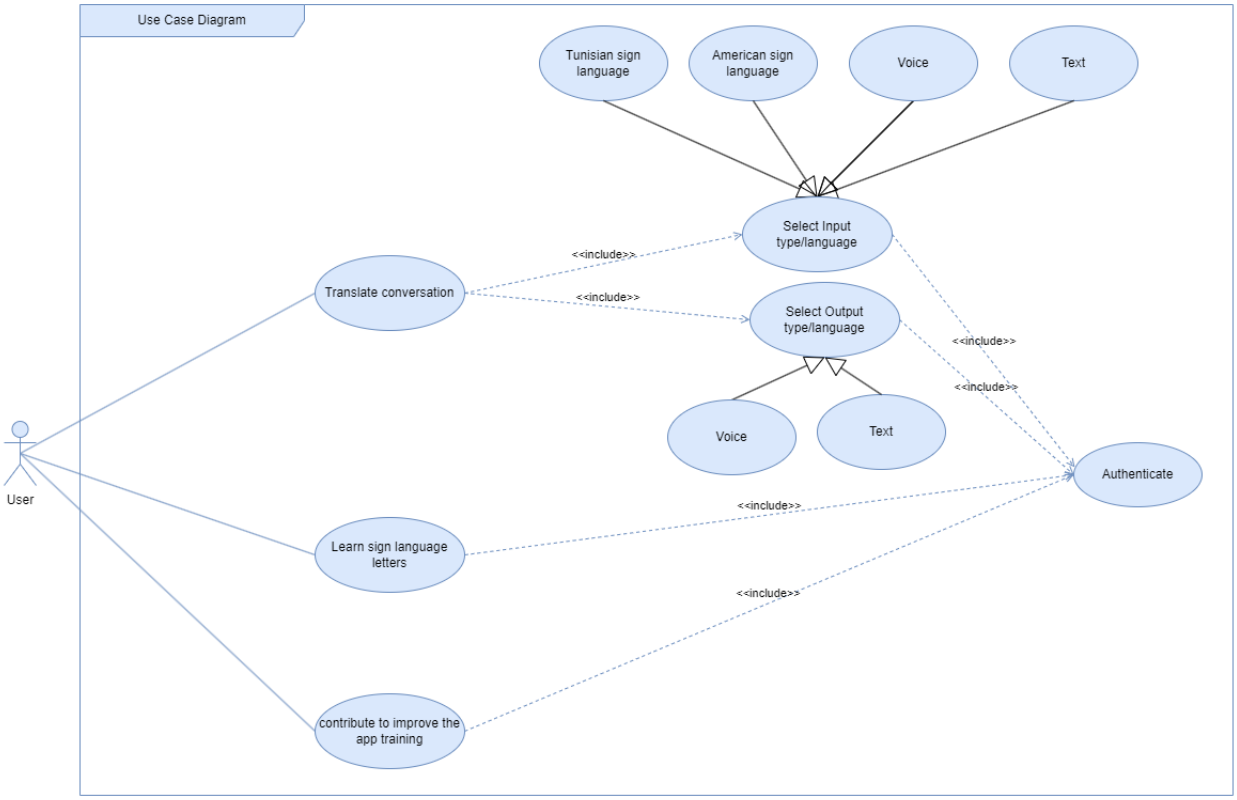


Figure 3.1: Use Case Diagram

3.1.5 System Sequence Diagram

A system sequence diagram is a visual representation summarizing the flow of interactions and messages between an external actor and a system for a particular use case or scenario.

To enhance clarity, we initiate by elucidating the sequence diagram for authentication, then proceed to delve into the diagram for sign language conversation translation in Section, followed by the diagram for Teaching sign language letters. Finally, we explore the sequence diagram for contributing to our app.

3.1.5.1 Sequence Diagram to Use Case "Authentication"

Figure 3.2 displays the sequence diagram detailing the authentication process. When the user visits the authentication page, they input their email address and password as a means to access our application. The system proceeds to verify this data, granting authentication when it proves accurate. Should the input data be found invalid, the system promptly displays an error message, affording the user the opportunity to make another attempt.

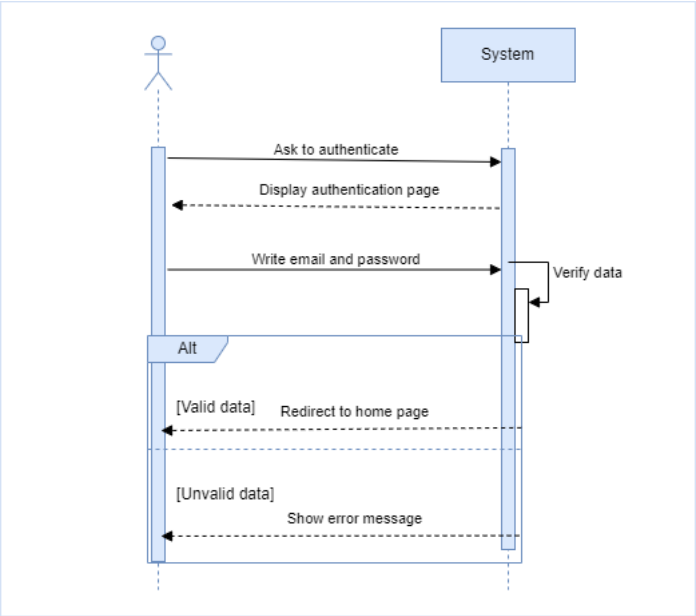


Figure 3.2: Authentication sequence diagram

3.1.5.2 Sequence Diagram to Use Case "Sign language translation"

Figure 3.3 presents the sequence diagram that outlining the conversation translation process. It begins with the user initiating the process by selecting their input type, whether it is Tunisian or American Sign Language, text, or voice. Next, they choose the output type, which can be text or voice. Following these selections, the user presses the 'Start' button, initiating the translation process, which prompts the system to translate the conversation.

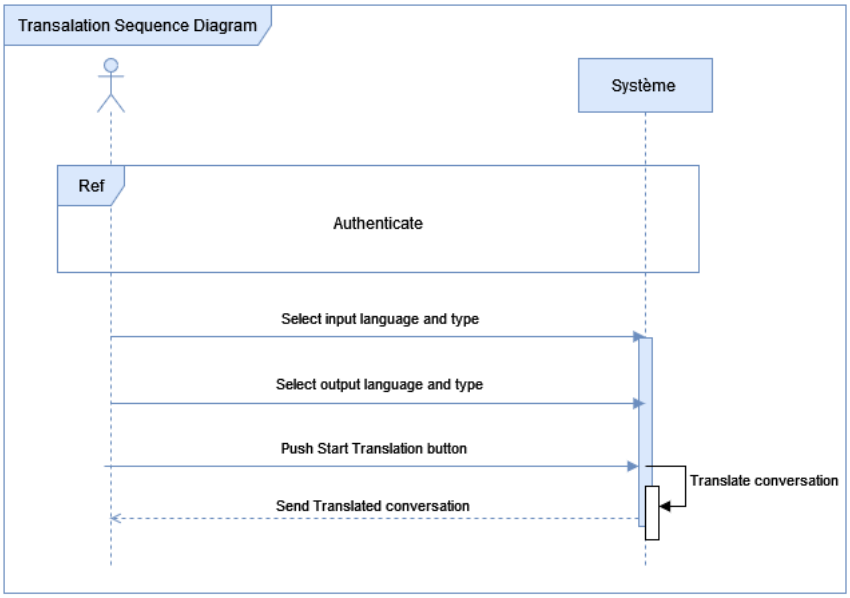


Figure 3.3: Translation Sequence Diagram

3.1.5.3 Sequence Diagram to Use Case "Learn sign language letters"

Figure 3.4 portrays the sequence diagram delineating the teaching process. It commences with the user choosing whether to begin learning sign language letters, test their knowledge of sign letters, or access the relevant system pages corresponding to their choice

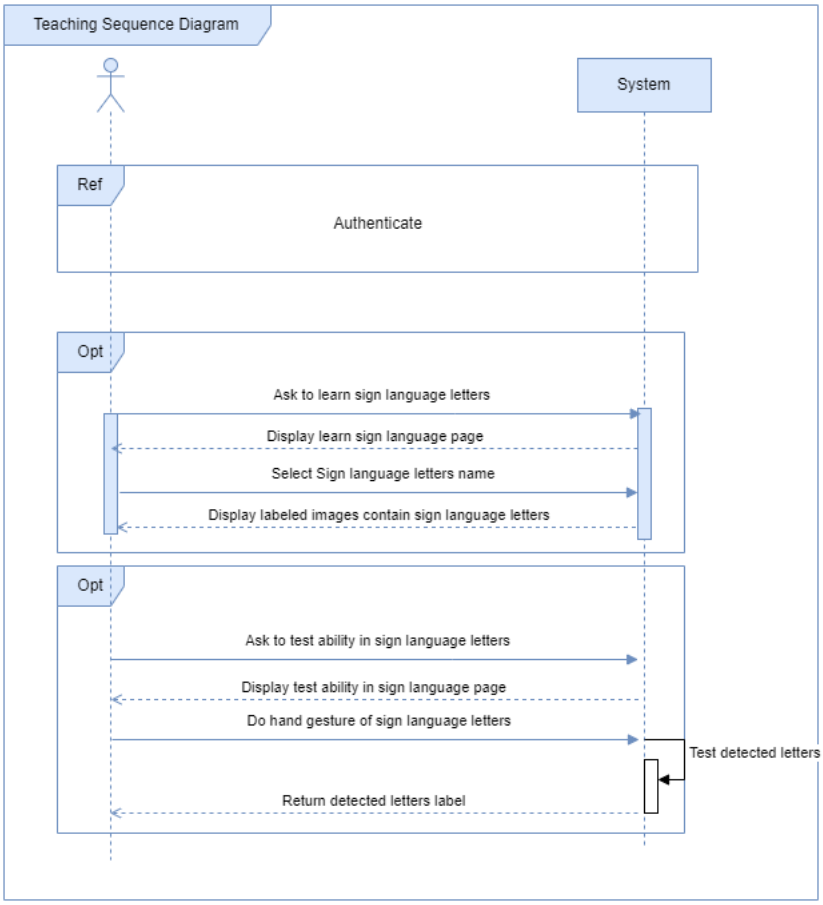


Figure 3.4: Teach Sign Language letters Sequence Diagram

3.1.5.4 Sequence Diagram to Use Case "Contribute to improve App learning"

Figure 3.5 depicts the sequence diagram that outlines the contribution process. It commences with the user entering the sign language name, followed by inputting the word they wish to add. Afterward, the user uploads a video or image related to the word and proceeds to submit their contribution. The system then verifies the input and, upon successful verification, submits the contribution, providing a success message. In cases where the input is not valid, the system sends an error message.

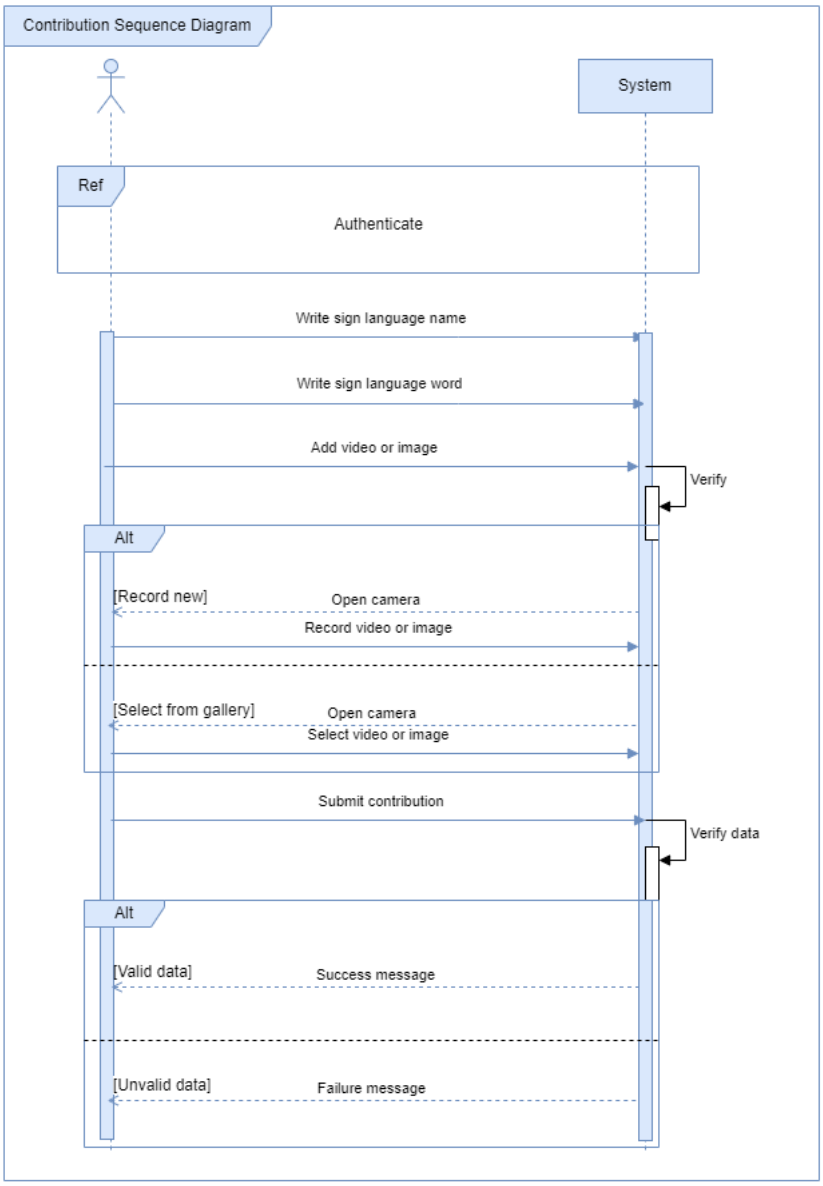


Figure 3.5: Contribution Sequence Diagram

3.2 Design

3.2.1 Architectural Design

In this section, we propose a two-fold approach. First, we present a logical view through the MVC (Model-View-Controller) model. Then, we provide the physical view of the system.

3.2.1.1 MVC

The MVC (Model-View-Controller) architecture is a widely adopted framework in the realm of web and mobile application development. This architectural approach effectively partitions an application into three interconnected layers: the model, the view, and the controller, as exemplified in Figure 3.6. The model layer serves as the repository for both data and the application's business logic, ensuring efficient management of data storage, retrieval, and manipulation. On the other hand, the view layer assumes the crucial role of presenting data within the user interface, handling the visual representation and facilitating user interaction. Additionally, the controller layer serves as an intermediary that manages the flow of data and events between the model and the view.

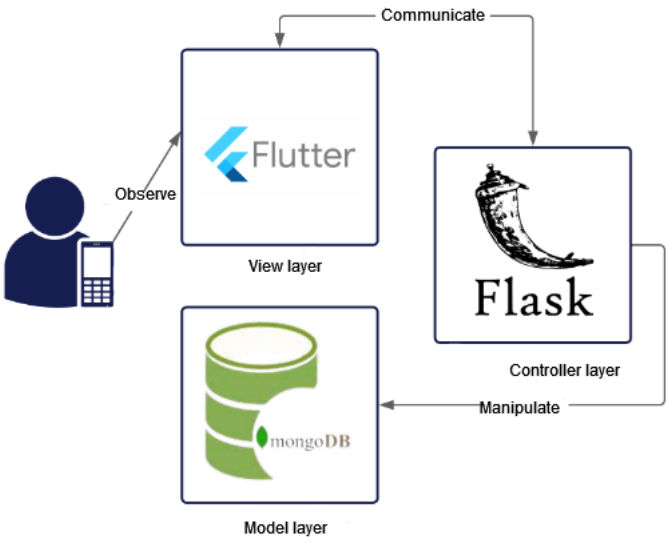


Figure 3.6: MVC Architecture

3.2.1.2 Physical architecture

Physical architecture is the arrangement and organization of physical components or elements within a system, structure, or entity. In our scenario, we selected a physical architecture featuring centralized data management, resulting in improved project management efficiency and ensuring overall project coherence, specifically employing a 3-tier architectural framework.

A 3-tier architectural framework, an integral component of our study, embodies a triad of distinct layers — presentation, application logic, and data storage — synergistically harmonizing to underpin the operational fabric of the software application under scrutiny. Figure 3.2 illustrates the three levels:

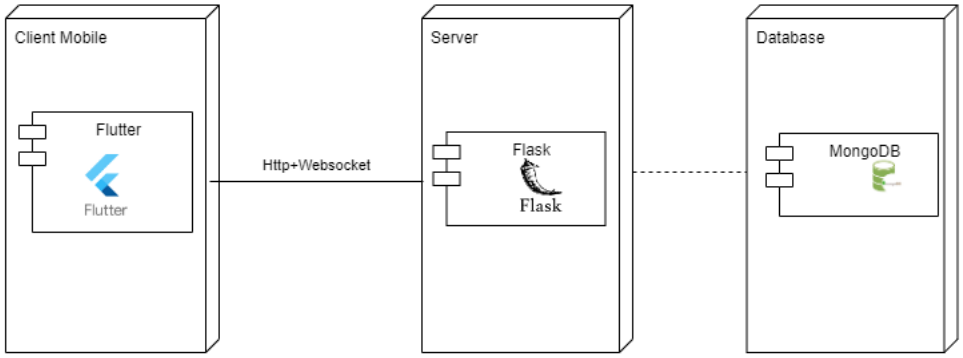


Figure 3.7: System Deployment Diagram

Conclusion

In this chapter, we have established a solid and reliable foundation for the realization of Sign AI. Furthermore, the next chapter focuses on the implementation phase.

Chapter 4

Sign AI approch and experimental study

Contents

Introduction

This chapter, dedicated to experimental investigation, comprises several key sections that showcase our approach and findings. We initiate by outlining the solution's methodology in Section 4.1, elaborating on the overarching approach adopted. Sections 4.2 and 4.3 are dedicated, respectively, to data collection and preprocessing. We continue in Section 4.4 by elucidating the model training process, encompassing architectures, algorithms, and specific techniques employed for training. The subsequent section focuses on assessing model performance using specific metrics. The final part is dedicated to the deployment of our solution.

4.1 Solution approach

The strategy of our solution, as illustrated in Figure 4.1, covers multiple phases, starting from data collection and extending to the presentation of results within a mobile application. To enhance the organization of this procedure, we segment it into three primary sections:

- Machine Learning Pipeline: The machine learning process depicted in the Figure 4.1 is utilized for several tasks, including classification of both American Sign Language and Tunisian Sign Language conversations, classification of Arabic and American Sign Language letters.

- Inference Phase: This stage involves putting the artificial intelligence models into production and deploying them using sockets, which are developed in the backend portion of the project.

- Mobile Application: In this phase, the mobile application connects to the backend via sockets and displays the prediction results.

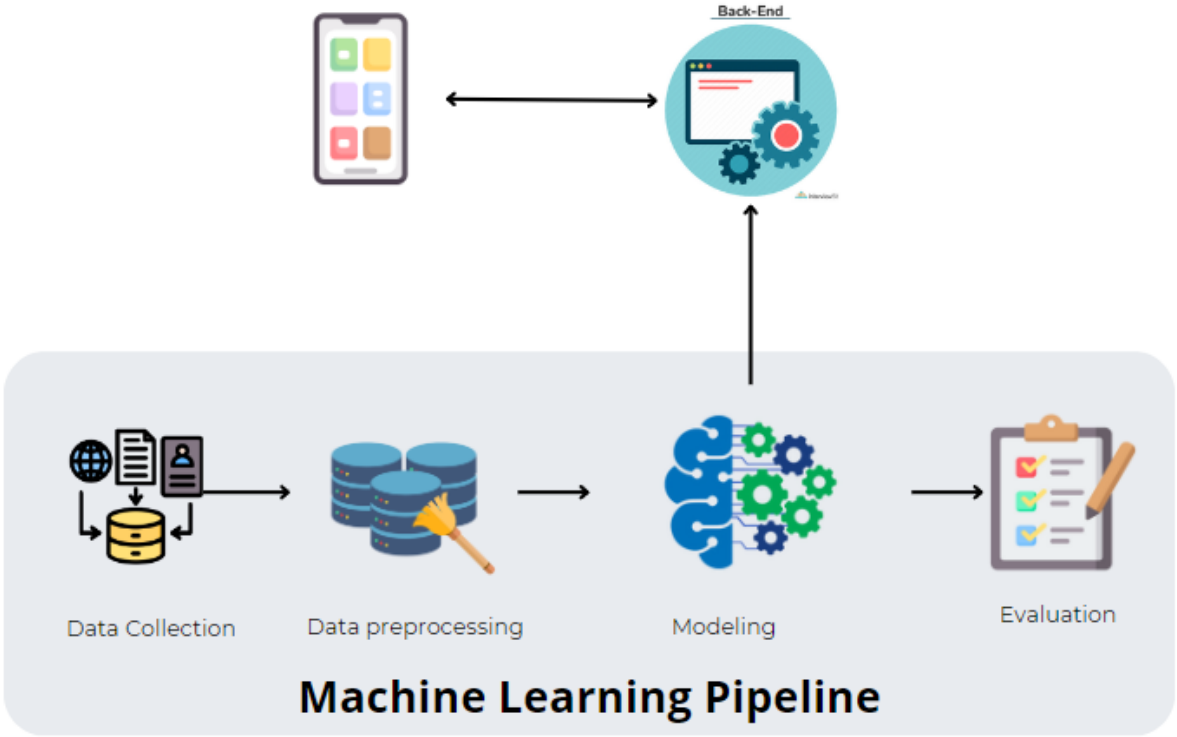


Figure 4.1: Solution approach

4.2 Data Collection

The data collection process holds pivotal significance in the development of our application. To ensure the creation of a comprehensive dataset and the establishment of a strong foundation for various functionalities, we strategically leverage a variety of data sources. By combining these sources, we are able to collect data for the classification of both American Sign Language and Tunisian Sign Language conversations, as well as the classification of Arabic and American Sign Language letters.

Regarding the classification of American Sign Language, we have identified a relevant dataset on Kaggle known as "WLASL". "WLASL" is the largest video dataset designed for Word-Level American Sign Language (ASL) recognition. It encompasses 2,000 common and distinct ASL words. It's important to note that all WLASL data is exclusively intended for academic and computational purposes, and commercial usage is prohibited. This dataset was created by Dongxu Li and Hongdong Li, and further information can be found in their official paper titled "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison."

Here is a brief description of the WLASL dataset:

- Purpose: The primary purpose of the WLASL dataset is to facilitate research in sign language recognition, translation, and related computer vision tasks.
- Content: The dataset contains video clips of individuals signing various ASL words or signs. Each video clip corresponds to a specific word in American Sign Language.
- Size: The dataset consists of a large number of video samples, covering a wide range of ASL signs. The exact number of signs and video clips may vary depending on the specific version or release of the dataset in our case we have 2,000 word(classes).
- Annotations: Each video clip is typically annotated with the corresponding ASL word or sign it represents. This annotation is essential for training and evaluating machine learning models.

In the context of classifying Tunisian Sign Language, we employ a custom video dataset that we meticulously curated. This dataset was meticulously assembled through an extensive research effort, involving outreach to members of the deaf community who generously contributed to its creation. It encompasses video recordings featuring my facial expressions.

At the outset, this dataset consisted of 19 unique words (classes), with each word comprising between 5 to 15 videos, each of which captured a specific gesture.

1 to 19 of 19 entries Filter

index	Subdirectory	File Count
0	abyedh	13
1	ahad	15
2	ahmer	8
3	akhdher	9
4	akhel	7
5	asfer	15
6	azrak	10
7	irbaa	11
8	jemaa	11
9	khmis	10
10	labes	10
11	mahlek	10
12	nhebk	11
13	omi	5
14	salem alaykom	12
15	sebet	15
16	thnin	11
17	tleth	15
18	tunis	10

Show 25 per page

Figure 4.2: Tunisian sign language intial dataset

For American Sign Language Letters, we leveraged a dataset comprising 26 distinct classes. This dataset was meticulously curated by David Lee, a dedicated data scientist specializing in accessibility. It serves as a valuable object detection resource, encompassing individual ASL letters, each meticulously annotated with its corresponding bounding box. David Lee's diligent efforts in compiling this dataset have made it publicly accessible, thereby significantly contributing to the advancement of research and development in the field of sign language recognition.

4.3 Data preprocessing

4.3.1 Data Resizing

In the context of our project, our initial step involves resizing the videos within our Tunisian Sign language and American Sign language datasets to a standardized dimension of 224x224x3. This choice of dimensions is driven by the objective of achieving maximum compatibility with the various models employed in our research. This specific size is well-regarded within the field for its efficiency and superior performance in video processing tasks. Furthermore, this standardized dimension not only facilitates quicker computations but also ensures the retention of crucial information within each video. Similarly, for our ASL classification image dataset, we adhere to this same resizing process, configuring the images to 244x244x3 dimensions to maintain consistency and compatibility throughout our research endeavors.

4.3.2 Data augmentation

During the implementation of machine learning, the shortage of data often poses a significant challenge. Indeed, the performance and accuracy of a deep learning model are greatly enhanced when it is trained on a large and diverse dataset. However, collecting a substantial amount of data can be costly in terms of time and financial resources. This is why we resort to artificial techniques to increase the size of our datasets. We primarily employ these techniques to overcome this data limitation.

- Rotations° : This technique involves rotating images by a certain angle (e.g., 90 degrees, 180 degrees, etc.). It helps the model become invariant to the orientation of objects in the images.

- Gray scaling : Converting color images to grayscale reduces the dimensionality of the data and can help the model focus on important features, especially when color is not

critical to the task.

```
# Define the augmentations to apply
def Grayscalev(video_path,filename,i):


  augmentationsgrayscale = iaa.Sequential([
     iaa.Grayscale(alpha=( 0.5*i)),
  ])

# Open the video66039.mp4'
  cap = cv2.VideoCapture(video_path)

# Create the output video
  fourcc = cv2.VideoWriter_fourcc(*"mp4v")
  new_video_filename = "{}_{}{}.mp4".format(filename, 'gray',i)
  print(new_video_filename)
  out = cv2.VideoWriter(new_video_filename, fourcc, cap.get(cv2.CAP_PROP_FPS),
   (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

# Loop through each frame of the video
  while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        # Apply augmentations to the frame
        frame = augmentationsgrayscale.augment_image(frame)

        # Write the augmented frame to the output video
        out.write(frame)
    else:
        break

# Release the video capture and output writer
  cap.release()
  out.release()
```

Figure 4.3: Video Grayscale

- Saturation : Adjusting the saturation of an image can make it more or less colorful. This can help the model learn to recognize objects under different lighting conditions or color variations.

```python
# Define the augmentations to apply
def Saturationv(video_path,filename,i):


  augmentationsSaturation = iaa.Sequential([
    iaa.AddToHueAndSaturation(( 50*i)), # Add random values to the hue and saturation channels
  ])

# Open the video66039.mp4'
  cap = cv2.VideoCapture(video_path)

# Create the output video
  fourcc = cv2.VideoWriter_fourcc(*"mp4v")
  new_video_filename = "{}_{}_{}.mp4".format(filename, 'sat',i)
  print(new_video_filename)
  out = cv2.VideoWriter(new_video_filename, fourcc, cap.get(cv2.CAP_PROP_FPS),
   (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

# Loop through each frame of the video
  while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        # Apply augmentations to the frame
        frame = augmentationsSaturation.augment_image(frame)

        # Write the augmented frame to the output video
        out.write(frame)
    else:
        break

# Release the video capture and output writer
  cap.release()
  out.release()
```

Figure 4.4: Video saturation

- Horizontal & vertical flips : Flipping images horizontally or vertically creates mirror images. This is particularly useful for tasks where the orientation of objects doesn't matter, such as image classification.

```
# Define the augmentations to apply
def flipv(video_path,filename,i):


  augmentationsverticalflip = iaa.Sequential([

    iaa.Flipud(0.5*i),  # Flip vertically with a probability of 0.5
  ])

# Open the video66039.mp4'
  cap = cv2.VideoCapture(video_path)

# Create the output video
  fourcc = cv2.VideoWriter_fourcc(*"mp4v")
  new_video_filename = "{}_{}_{}.mp4".format(filename, 'flipv',i)
  print(new_video_filename)
  out = cv2.VideoWriter(new_video_filename, fourcc, cap.get(cv2.CAP_PROP_FPS),
   (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

# Loop through each frame of the video
  while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        # Apply augmentations to the frame
        frame = augmentationsverticalflip.augment_image(frame)

        # Write the augmented frame to the output video
        out.write(frame)
    else:
        break

# Release the video capture and output writer
  cap.release()
  out.release()
```

Figure 4.5: Vertical flip

```python
def fliph(video_path,filename,i):


  augmentationsHflip = iaa.Sequential([

    iaa.Fliplr(0.5*i),  # Flip horizontally with a probability of 0.5
  ])

# Open the video66039.mp4'
  cap = cv2.VideoCapture(video_path)

# Create the output video
  fourcc = cv2.VideoWriter_fourcc(*"mp4v")
  new_video_filename = "{}_{}_{}.mp4".format(filename, 'fliph',i)
  print(new_video_filename)
  out = cv2.VideoWriter(new_video_filename, fourcc, cap.get(cv2.CAP_PROP_FPS),
    (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

# Loop through each frame of the video
  while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        # Apply augmentations to the frame
        frame = augmentationsHflip.augment_image(frame)

        # Write the augmented frame to the output video
        out.write(frame)
    else:
        break

# Release the video capture and output writer
  cap.release()
  out.release()
```

Figure 4.6: Horizontal flip

- Scaling : Scaling involves resizing images to different dimensions. Scaling can simulate objects at different distances or sizes in the dataset and helps the model generalize better.

- Noise : Adding noise to images can mimic real-world variations in data. Gaussian noise, for example, introduces random pixel-level variations. Noise augmentation helps the model become more robust to imperfections in the data.

```python
# Define the augmentations to apply
def noisev(video_path,filename,i):
  augmentationsnoise = iaa.Sequential([
    iaa.AdditiveGaussianNoise(scale=0.1*255*i), # Add Gaussian noise with a scale of 0.1

  ])



# Open the video66039.mp4'
  cap = cv2.VideoCapture(video_path)

# Create the output video
  fourcc = cv2.VideoWriter_fourcc(*"mp4v")
  new_video_filename = "{}_{}_{}.mp4".format(filename, 'noise',i)
  print(new_video_filename)
  out = cv2.VideoWriter(new_video_filename, fourcc, cap.get(cv2.CAP_PROP_FPS),
   (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))

# Loop through each frame of the video
  while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        # Apply augmentations to the frame
        frame = augmentationsnoise.augment_image(frame)

        # Write the augmented frame to the output video
        out.write(frame)
    else:
        break

# Release the video capture and output writer
  cap.release()
  out.release()
```

Figure 4.7: Noise

- Data augmentationn for American Sign language Words:

Figure illustrates the various techniques used to augment video dataset of American sign language words.

Figure 4.8: Data augmentation

We encountered a unique challenge. Our dataset initially consisted of a staggering 2000 classes, each representing different categories or labels. However, the distribution of samples across these classes was far from proportional. Some classes contained as few as 5 samples, while others boasted a much larger count, with up to 16 samples per class. This initial imbalance posed a significant obstacle to our subsequent modeling efforts.

To rectify this situation, we recognized the need for data augmentation, a pivotal step in the process. Our aim was to rebalance the dataset by generating additional data points, particularly for those underrepresented classes. This augmentation process played a crucial role in ensuring that each class had a more equitable representation in our dataset, ultimately improving the fairness and accuracy of our subsequent machine learning models.

Our progress can be visually traced in Figure 4.3, which illustrates the before-and-after effects of the data augmentation process. This transformation allowed us to achieve a more balanced dataset, setting the stage for more effective model training and evaluation.

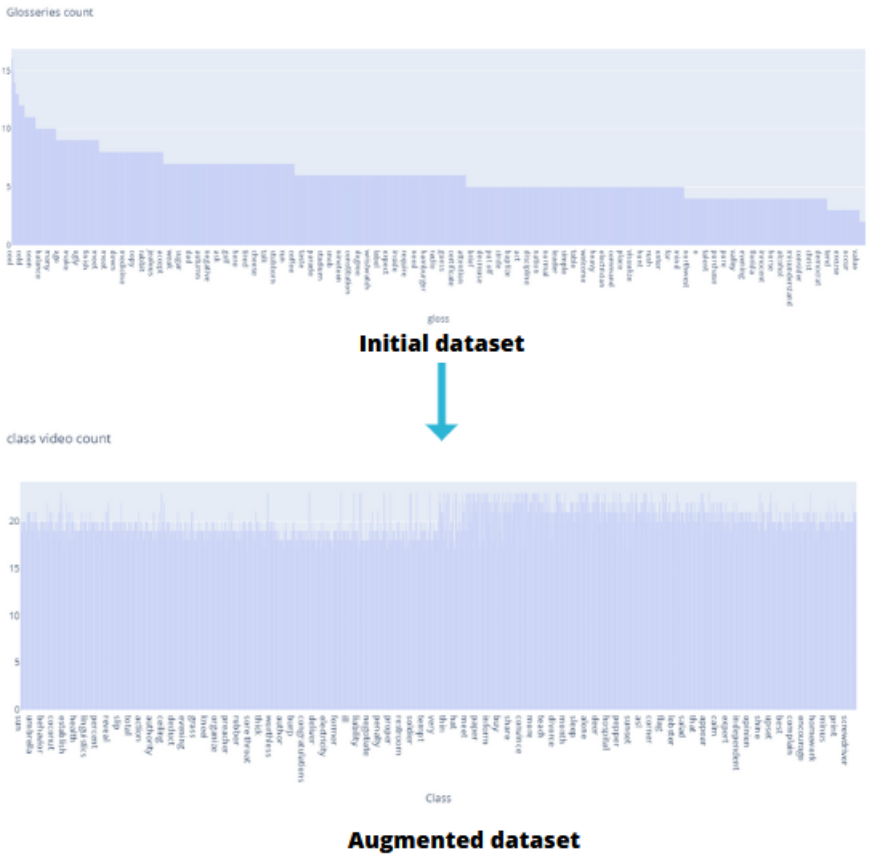


Figure 4.9: Augmented dataset

Building on this foundation, we made the strategic decision to narrow our focus. From the now-augmented dataset, we meticulously selected a subset of 35 words, as shown in figure 4.4 that were most relevant to our research objectives. By concentrating our efforts on this smaller, more manageable subset, we aimed to optimize our model's performance while minimizing computational complexity.

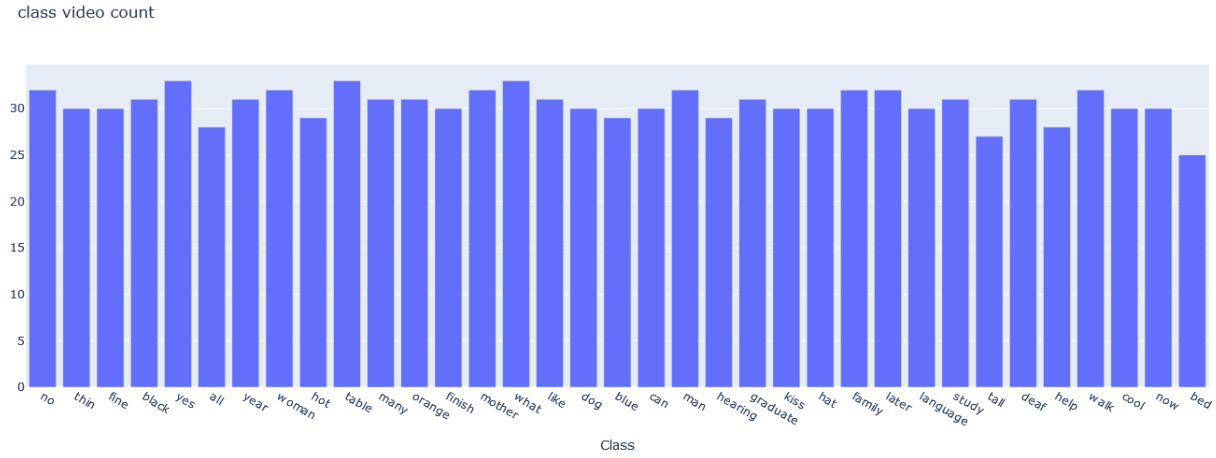


Figure 4.10: American sign language words final dataset distribution

- Data augmentationn for Tunisan Sign language Words

In our investigation of Tunisian Sign Language, our focus shifted to a dataset comprising 19 distinct classes, each representing specific sign language gestures and expressions. Just as in our previous data preprocessing endeavors, we encountered an initial data imbalance within these classes, where certain classes had more data samples than others. We applied a similar approach when dealing with American Sign Language data to address this issue. The graphical representation in Figure 4.5 visually encapsulates the ultimate dataset distribution

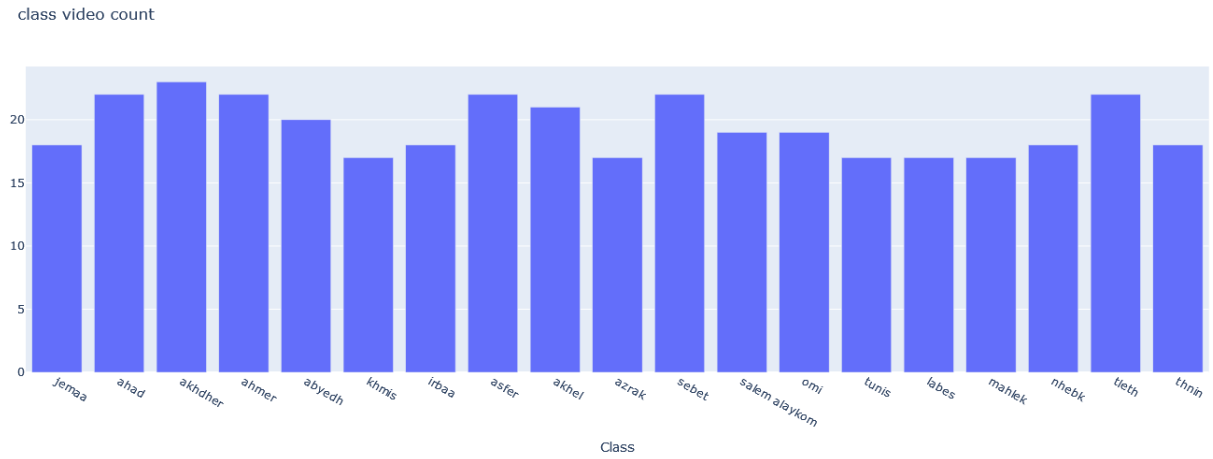


Figure 4.11: Tunisian sign language words final dataset distribution

- Data augmentationn for American Sign language letters

In our exploration of American Sign language letters, our attention turned toward a dataset that encompasses 26 unique classes, each corresponding to a distinct sign language gesture or expression, mirroring the number of alphabets. However, we initially confronted issues of data imbalance within these classes, compounded by the dataset's relatively small size. To address these challenges, we implemented several data augmentation techniques, including rotation, random flip, and random brightness adjustment. Figure 4.6, depicted graphically, serves as a visual representation that succinctly conveys the final distribution of our dataset after these augmentation measures were applied.

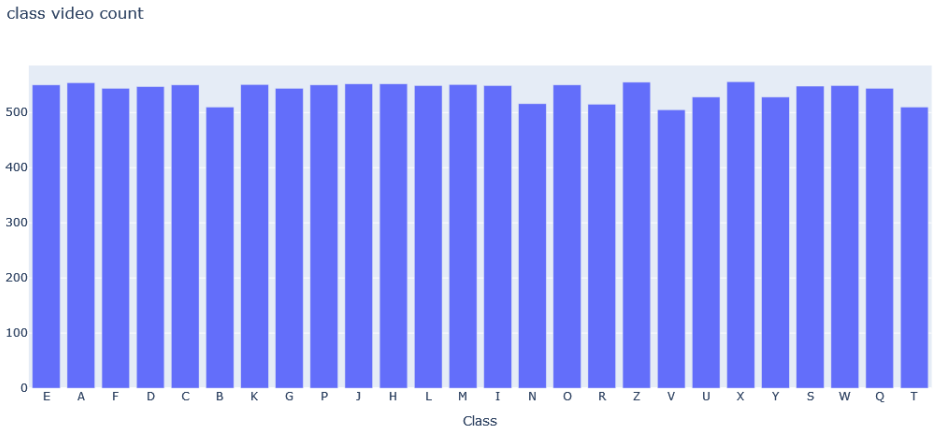


Figure 4.12: American sign language letters final dataset distribution

4.3.3 Video framing

In the context of our project, it is crucial to prepare and format video frames extracted from a dataset to ensure they are compatible with our machine learning models. The following process outlines how we format frames from video files:

- Frame Formatting Function (format frames):
  - This function takes as input an individual frame from a video and the desired output size for the frame.
  - Initially, it converts the frame to a floating-point format for numerical compatibility.
  - Next, it resizes the frame while maintaining its aspect ratio to fit the specified output size. The resizing process includes padding to ensure that the output frame retains its original aspect ratio.
- Frames Extraction Function (frames from video file):
  - This function is responsible for extracting frames from a video file.
  - It takes several parameters, including the path to the video file, the number of frames to be created per video, the desired output size for the frames, and the frame step (which determines how many frames are skipped between each extracted frame).
  - The function first opens the video file for reading frame by frame.
  - It calculates the total number of frames in the video file (video length) and determines the number of frames needed to fulfill the specified number of frames.
  - To ensure randomness in frame selection, it calculates a random starting point (start) within the video, considering the required number of frames and frame step.
  - The function then seeks to the calculated starting frame in the video.
  - It reads and formats the first frame, which is appended to the result list.
  - Subsequently, the function iteratively reads and formats the subsequent frames, skipping frames as determined by the frame step.
  - If the video ends before extracting all required frames, it pads the result with black frames to maintain the desired frame count.
  - Finally, the function returns an array containing the formatted frames in the shape of (number frames, height, width, channels).

- Color Channel Reordering: Before returning the formatted frames, the function reorders the color channels to match the standard RGB format (Red, Green, Blue) commonly used in image processing and machine learning.

This frame formatting process is a critical step in our project, ensuring that our input data is standardized and ready for subsequent machine learning tasks such as model training and evaluation. It ensures that all frames are of consistent dimensions and pixel values, facilitating seamless integration into our computer vision pipeline.

4.3.4 Final Data Structure

Before beginning the modeling process, it is crucial to prepare our data by dividing it into training and testing sets. This step, known as the 'train-test split,' involves separating our initial dataset into two distinct parts, as shown in Figure 4.6. Indeed, we allocate 80% of the data for training and 20% for testing.

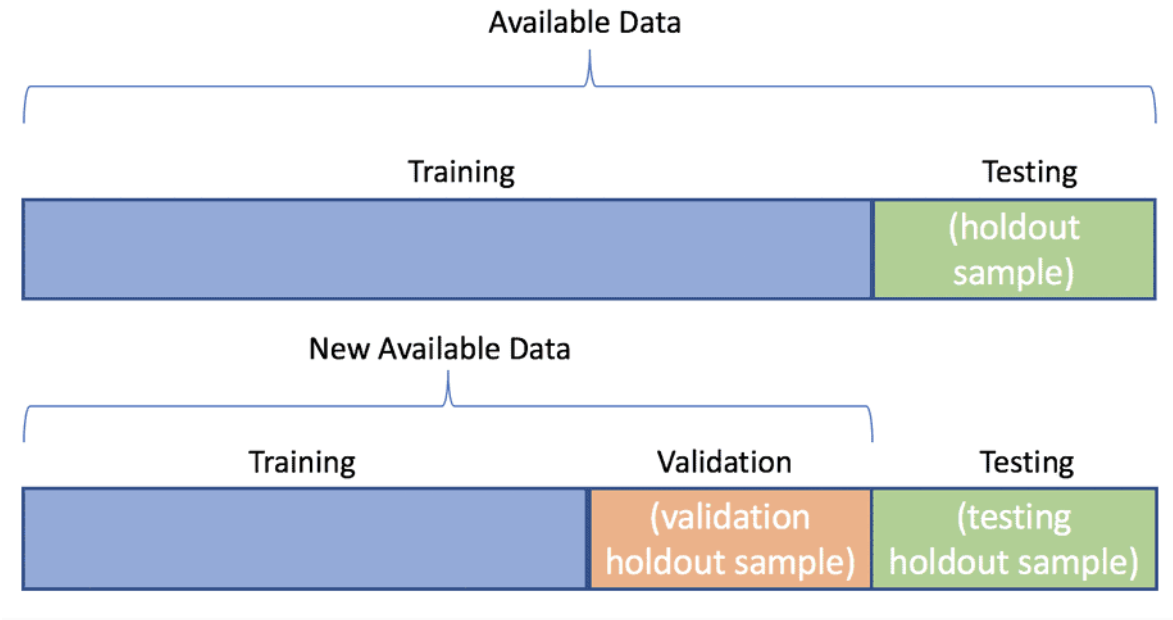


Figure 4.13: Data Partitioning

4.4 Learning

American Sign Language Classification:

For the classification of American Sign Language (ASL), we have adopted a transfer learning approach, leveraging the power of pre-trained convolutional neural networks (CNNs). Specifically, we have employed architectures like EfficientNet B0 and MovinetA0, which have consistently demonstrated exceptional performance across a wide range of computer vision tasks.

This choice of pre-trained models empowers our system to accurately classify and interpret sign language gestures in the ASL domain. By building upon the knowledge and representations learned from large-scale datasets in these models, we are able to expedite the development of an ASL recognition system with a strong foundation.

Tunisian Sign Language Classification:

In the context of Tunisian Sign Language (TSL) classification, we have similarly adopted a transfer learning approach. Our arsenal includes architectures such as EfficientNet B0 and MovinetA0, known for their prowess in computer vision applications.

By applying transfer learning to the TSL domain, we leverage the generalizable features learned by these pre-trained models. This approach significantly enhances our capacity to accurately classify and interpret gestures in TSL, ultimately leading to a robust and effective TSL recognition system.

Additionally, our exploration extends to the use of a Sequential classifier, offering a versatile approach to tackling the complexities of TSL recognition.

Figure 4.14 shows an Example of my model for Tunisian sign language words learning

```
23]
    net = tf.keras.applications.EfficientNetB0(include_top = False)
    net.trainable = False

    model = tf.keras.Sequential([
        tf.keras.layers.Rescaling(scale=255),
        tf.keras.layers.TimeDistributed(net),
        tf.keras.layers.Dense(19,activation='relu'),
        tf.keras.layers.GlobalAveragePooling3D()
    ])
    model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])
```

Figure 4.14: Tunisian sign language words learning

	Parameter	value	Output Shape	Param #
0	Input Shape	(2, 10, 224, 224, 3)	NaN	NaN
1	Activation Function	Relu	NaN	NaN
2	Optimizer	Adam	NaN	NaN
3	Loss Function	SparseCategoricalCrossentropy	NaN	NaN
4	Metrics	Accuracy	NaN	NaN
5	Rescaling	NaN	(2, 10, 224, 224, 3)	0
6	TimeDistributed	NaN	(2, 10, 7, 7, 1280)	4,049,571
7	Dense	NaN	(2, 10, 7, 7, 19)	24,339
8	GlobalAveragePooling3D	NaN	(2, 19)	0

Figure 4.15: Table of Parameters of Tunisian sign language words learning

Sign Language Letter Classification:

For the task of sign language letter classification, we have chosen to employ a diverse set of architectures, namely VGG16, ResNet50, and a Simple CNN Classifier. These architectures are well-regarded in the computer vision community and are widely recognized for their effectiveness in image classification tasks.

Our iterative optimization process involves tuning hyperparameters, such as the number of training epochs and batch size, to fine-tune these models. This meticulous fine-tuning effort is aimed at boosting the accuracy and overall performance of our models in the critical task of classifying sign language letters.

By harnessing the strengths of these architectures and continuously refining our models, we are committed to achieving state-of-the-art accuracy and delivering a robust solution for sign language letter classification.

Figure 4.15 shows ou model for Sign Language Letter Classification learning

```
# Load the pre-trained ResNet50 model
resnet = ResNet50(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

# Add a fully connected layer on top of the ResNet50 model
model_res = Sequential()
model_res.add(resnet)
model_res.add(Flatten())
model_res.add(Dense(256, activation='relu'))
model_res.add(BatchNormalization())
model_res.add(Dropout(0.5))
model_res.add(Dense(26, activation='softmax'))
```

Figure 4.16: Sign Language Letter Classification learning

	Layer	Output Shape	Param #
0	ResNet50 (Functional)	(None, 7, 7, 2048)	23,587,712
1	Flatten	(None, 100352)	0
2	Dense	(None, 256)	25,690,368
3	BatchNormalization	(None, 256)	1,024
4	Dropout	(None, 256)	0
5	Dense	(None, 26)	6,682

Figure 4.17: Table of parameters of Sign Language Letter Classification learning

4.5 Evaluation

Section 4.5.1 focuses on the selection of evaluation metrics, while the other sections are dedicated to the results of the models.

4.5.1 Selection of Evaluation Metrics

When making classification predictions, there are four types of outcomes:

- False Positives
- False Negatives
- True Positives
- True Negatives

In our approach, our primary objective is to minimize false positive predictions. Therefore, we adopt precision as the primary evaluation metric for classification tasks. Additionally, for localization tasks, we incorporate both precision and IOU (Intersection over Union) as evaluation measures to conduct a more comprehensive analysis of the results. This dual assessment considers not only the accuracy of predictions but also their alignment with reference regions, offering a more holistic evaluation of the model's performance

4.5.2 Results of American Sign Language Words Classification Models

Table 4.1 presents the performance results of our classification models for recognizing American Sign Language (ASL) words. The models included in our evaluation are EfficientNetB0 and MobileNetA0. Each model underwent training with a specific learning rate, a fixed batch size, and over a set number of training iterations.

Upon a careful analysis of the model performances, we observe that EfficientNetB0 achieved the highest accuracy, attaining an impressive score of 87.60%. Furthermore, the MobileNetA0 model exhibited commendable performance, achieving an accuracy rate of 2.57%.

These results lead us to conclude that the EfficientNetB0 model is the most suitable choice for accurately classifying ASL words. Consequently, we have selected the EfficientNetB0 model with 50 epochs to fulfill this crucial functionality in our system.

Model	Learning rate	Batch size	Number of iterations	Accuracy
Movinet A0	0.001	10	2	0.0257
Efficentnet B0	0.001	10	30	0.7696
Efficentnet B0	0.001	10	50	0.8760

Table 4.1: Results of the classification models for Tunisian Sign Language words

The loss curve of the selected model is depicted in Figure 4.7.

Figure 4.18: EfficientNetB0 Loss Curve

Interpretation:

- The x-axis represents the number of training epochs, which is the number of times the model has gone through the entire training dataset.
- The y-axis represents the loss, which is a measure of how well the model is performing. Lower values indicate better performance.
- The blue line represents the training loss, which should generally decrease over time as the model learns from the data.
- The orange line represents the validation loss, which is a measure of how well the model generalizes to unseen data. It's expected to decrease initially but may start to increase if the model overfits.

Observations:

- In the initial epochs, both training and validation loss decrease, indicating that the model is learning from the data.
- As the training progresses, there is a point where the training loss continues to decrease while the validation loss starts to level off or increase. This is a sign of overfitting, where the model is fitting the training data too closely and may not generalize well to new data.
- Monitoring the loss curves can help you decide when to stop training or apply techniques like early stopping to prevent overfitting and achieve the best model performance.

4.5.3 Results of Tunisian Sign Language Words Classification Models

Table 4.2 presents the results of the classification models for Tunisian Sign Language words. Indeed, the included models are EfficientNetB0 and MobileNetA0. Each model was trained with a specific learning rate, a fixed batch size, and over a duration of iterations.

Analyzing the model performances, we can observe that EfficientNetB0 achieved the highest accuracy with a score of 93,08%. Additionally, the MobileNetA0 model showed good performance with an accuracy of 81,19%.

These results suggest that the EfficientNetB0 model is the most effective for classifying the state of the car, followed by MobileNetA0. Therefore, we choose the EfficientNetB0 model for this functionality.

Model	Learning rate	Batch size	Number of iterations	Accuracy
Movinet A0	0.001	10	2	0.8119
Efficentnet B0	0.001	10	30	0.9308

Table 4.2: Results of the classification models for Tunisian Sign Language words

The loss curve of the selected model is depicted in Figure 4.7.

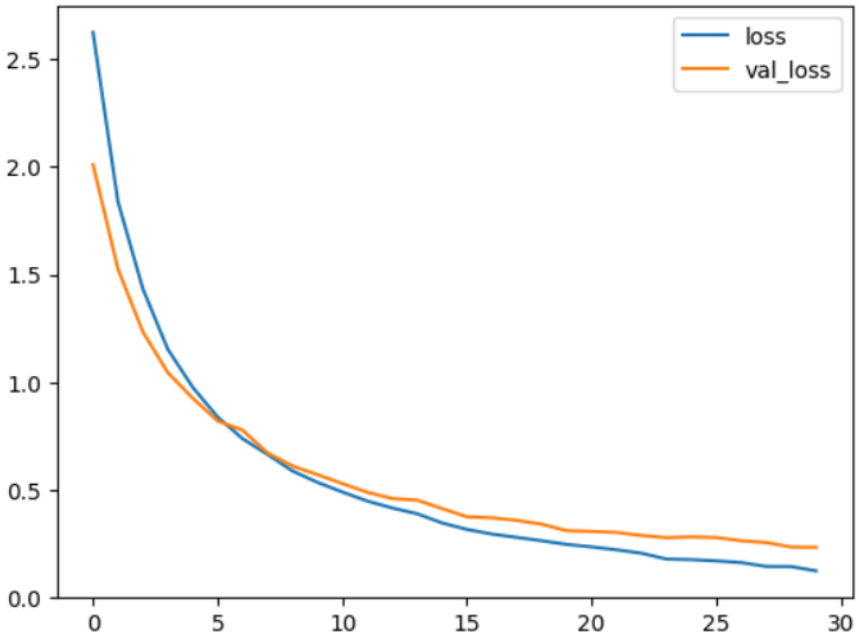


Figure 4.19: EfficientNetB0 Loss Curve

Interpretation:

- The x-axis represents the number of training epochs, and the y-axis represents the loss.
- The blue line represents the training loss, which should generally decrease as the model learns.
- The orange line represents the validation loss, which shows how well the model generalizes to unseen data.

Observations:

- In the initial epochs, both training and validation loss decrease, indicating that the model is learning.
- As training progresses, the training and validation loss continue to decrease, suggesting that the model is improving its performance.
- It's important to monitor the validation loss, and if it starts to increase while the training loss keeps decreasing, it may indicate overfitting, and you may consider applying regularization techniques or early stopping to prevent it.

Regenerate

4.5.4 Results of American Sign Language Letters Classification Models

American Sign Language Alphabet Classification Model Results Table 4.1 presents the performance outcomes of our classification models designed for recognizing the American Sign Language (ASL) alphabet. The models subjected to our evaluation include ResNet-50 and VGG16, each undergoing training with specific parameters such as learning rate, fixed batch size, and a predetermined number of training iterations.

Upon meticulous examination of the model performances, it becomes evident that ResNet-50 achieved the highest accuracy, demonstrating an impressive score of 96,47%. Additionally, the VGG16 model displayed commendable performance with an accuracy rate of 5,66% and the DenseNet model 58.82%.

Based on these findings, we can confidently conclude that the ResNet-50 model is the optimal choice for accurately classifying ASL alphabet signs. Consequently, we have

selected the ResNet-50 model, trained over 50 epochs, to fulfill this essential function in our system.(

Model	Learning rate	Batch size	Number of iterations	Accuracy
DenseNet201	0.001	10	20	0.582
Resnet-50	0.001	10	20	0.9647
VGG-16	0.001	10	30	0.0566

Table 4.3: Results of the classification models for Tunisian Sign Language words

The loss curve of the selected model is depicted in Figure 4.7.

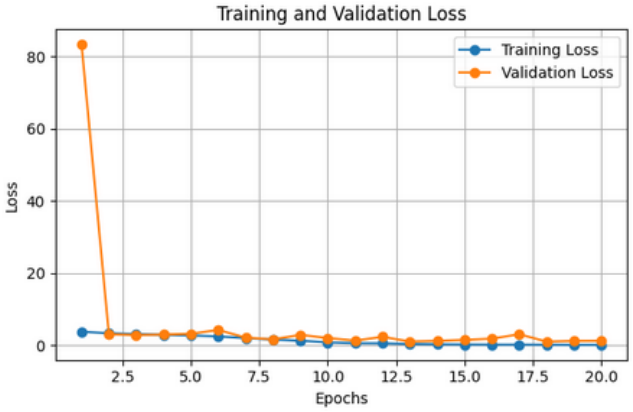


Figure 4.20: Resnet-50 Loss Curve

Interpretation:

- The x-axis represents the number of training epochs, while the y-axis represents the loss.
- The blue curve represents the training loss, which measures how well the model fits the training data.
- The orange curve represents the validation loss, indicating how well the model generalizes to unseen data.

Observations:

- In the initial training epochs, both the training and validation loss decrease. This behavior is typical and signifies that the model is learning and improving its performance.
- As training continues, both the training and validation loss continue to decrease. This trend indicates that the model is further refining its understanding of the data and enhancing its ability to make predictions.
- The convergence of the two loss curves (training and validation) suggests that the model is not overfitting. Overfitting would be a concern if the validation loss were to increase while the training loss continued to decrease. Fortunately, this is not the case here.
- The validation loss reaches a relatively stable level, which indicates that the model's performance on unseen data is consistent and reliable.

In summary, the provided training and validation loss curves demonstrate that the model is effectively learning from the training data and generalizing well to validation data. The absence of overfitting is a positive sign, and the model appears to be making steady progress in its training. Monitoring these loss curves is essential to ensure the model's quality and generalization capabilities throughout training.

4.6 Deployment

To ensure seamless integration and maximize the utility of the models and functionalities we're developing, we're establishing a robust WebSocket infrastructure using the

Flask framework. Specifically, we're creating dedicated WebSocket connections for each model and functionality within our system. For instance, to enable real-time translation of both Tunisian and American Sign Language conversations, we've implemented separate WebSocket connections that leverage our sign language word classification model. Additionally, for real-time detection of sign language letters, we've incorporated another WebSocket.

All of these WebSocket connections are seamlessly integrated into our front-end application, developed using Flutter. This design choice delivers a user-friendly interface, enhancing the overall user experience. By adopting this approach, we enable intuitive interactions with our system's functionalities, ultimately providing an enriched user experience.

```python
@socketio.on('video')
def TranslateConversation(video_data):

    print('Video received')  # Add this line to print a message

    # Save the video to disk or perform any desired processing
    video_bytes = video_data['videoBytes']
    text = video_data['text']
    print(text)
    with open('received_video.mp4', 'wb') as f:···
    sample_video = frames_from_video_file(os.path.join(BASE_DIR , 'received_video.mp4'), n_frames = 8)
    sample_video = np.expand_dims(sample_video, axis=0)  # Add batch dimension
    sample_video = np.expand_dims(sample_video, axis=-1)
    if text=='Tunisian Sign Language':
        result = model.predict(sample_video)
        classes=classesTSL
    else:
        result = model_.predict(sample_video)
        classes=classesWLSL

    predicted_labels = tf.argmax(result, axis=-1).numpy()
    # Send a response back to the Flutter app
    response =classes[predicted_labels[0]]
    print('reeesponse')
    print(response)
    socketio.emit('response', response)
```

Figure 4.21: Translate sign language conversation WebSocket

```python
@socketio.on('image')
def Classify_Signlanguageletters(image_data):
    # Save the image to disk or perform any desired processing
    image_bytes = image_data['imageBytes']
    # Open the image from bytes
    image = Image.open(BytesIO(image_bytes))
# Convert the image to RGB mode
    image = image.convert('RGB')
# Save the image as JPEG
    image.save('converted_image.jpg', 'JPEG')
    try:
      image = Image.open('converted_image.jpg')

      #image = format_image(image)
      image = image.resize(( 320, 320))
      image = np.array(image)  # Convert the image to a NumPy array
      image = image / 255.0  # Normalize the image

# Reshape the image to match the model's input shape (e.g., if the model expects 4D input)
      image = np.reshape(image, (1,  320, 320, 3))

      # Make predictions
      result = modelasl.predict(image)
      predicted = np.argmax(result[0])
      predicted_label=classesASL[predicted]

      # Send a response back to the Flutter app
      response = str(predicted_label)
      socketio.emit('response', response)
    except OSError as e:
      print('Error loading or resizing image:', e)

    except Exception as e:
      print('Error processing image:', e)
      socketio.emit('response', 'Error processing image', broadcast=True)
```

Figure 4.22: Classify sign language letters WebSocket

Conclusion

In this chapter, we have presented our approach, the results achieved, and the key steps of our project. The next chapter will be dedicated to the implementation of our solution.

Chapter 5

Implementation

Contents

Introduction

This chapter is dedicated to the implementation of our application. In section 5.1, we delve into the hardware environment utilized and the various technologies employed. Following that, in section 5.2, we introduce the different interfaces of "Sign AI".

5.1 Development Environment

In this section, we provide an overview of the hardware and software environment used in the implementation of "SignAI".

5.1.1 Hardware Environment

To bring our project to fruition, we select a hardware environment equipped with the following characteristics:

Google Colab:

It's a cloud-based Jupyter notebook platform that enables users to run and collaborate on Python code directly from their web browser, featuring the following specifications:

- CPU: Intel Xeon @ 2.20 GHz - 2 Cores
- GPU: Tesla T4 - 12 Go de VRAM GDDR5
- Memory: 13 GB
- Storage: 5 GB
- Session: 12 hours

Kaggle:

It's a Jupyter notebook platform hosted in the cloud, providing competitions and datasets tailored for Data Scientists and developers, featuring the following specifications:

- CPU: Intel Xeon @ 2.20 GHz - 4 Cores
- GPU: Tesla T4 ×2 | Tesla P100 - 16 GB GDDR5 VRAM
- Memory: 13 GB
- Storage: 5 GB
- Session: 12 hours

HP Pavilion Gaming Laptop:

For the development of the project, we utilized my personal Laptop with the following specifications:

- Operating System: Windows 11 Professional 64-bit
- Processor: AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz
- RAM: 16 GB
- Storage: 250GB SSD

5.1.2 Software Environment

In this section, we present the various technologies utilized in the realization of our project. Specifically, we have employed:

- Python: An interpreted programming language based on the object-oriented paradigm.
- Dart: a modern, object-oriented programming language developed by Google often used for building web and mobile applications with Flutter.
- Numpy: a fundamental Python library for numerical computing and data manipulation.
- Pandas: a powerful Python library widely used for data manipulation and analysis, offering versatile data structures like DataFrames and Series to simplify tasks related to data cleaning, exploration, and transformation.
- OpenCV: a computer vision and image processing software library, offering a comprehensive set of tools and functions for tasks like image and video analysis, object detection, and machine learning.
- Pillow: a Python Imaging Library (PIL) fork designed for image processing, manipulation, and format conversion in Python.
- sklearn:
- TensorFlow: An open-source machine learning library.
- PyTorch: A Python software library for deep learning.
- Flask: A Python web framework.
- Flutter: An open-source framework for building cross-platform applications.

- StarUML: UML modeling software.
- LaTeX: A system for composing academic documents.
- MongoDB: a versatile NoSQL database system known for its flexibility, scalability, and JSON-based data storage.

5.2 Sign AI description

In this segment, we showcase the user interface of our application, SignAI, to emphasize the diverse range of services it provides.

5.2.1 Home page

When users launch the application, they are initially presented with a splash screen, illustrated in Figure 5.1. To access their personal user space, they must then proceed through two introductory pages, as shown in Figures 5.2 and 5.3.

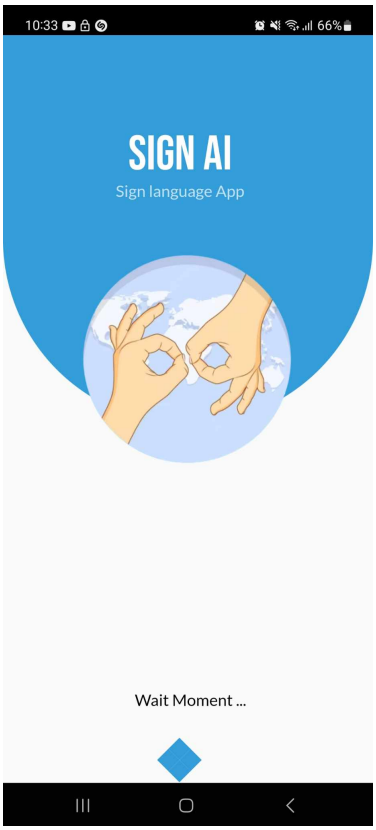


Figure 5.1: Splash Screen

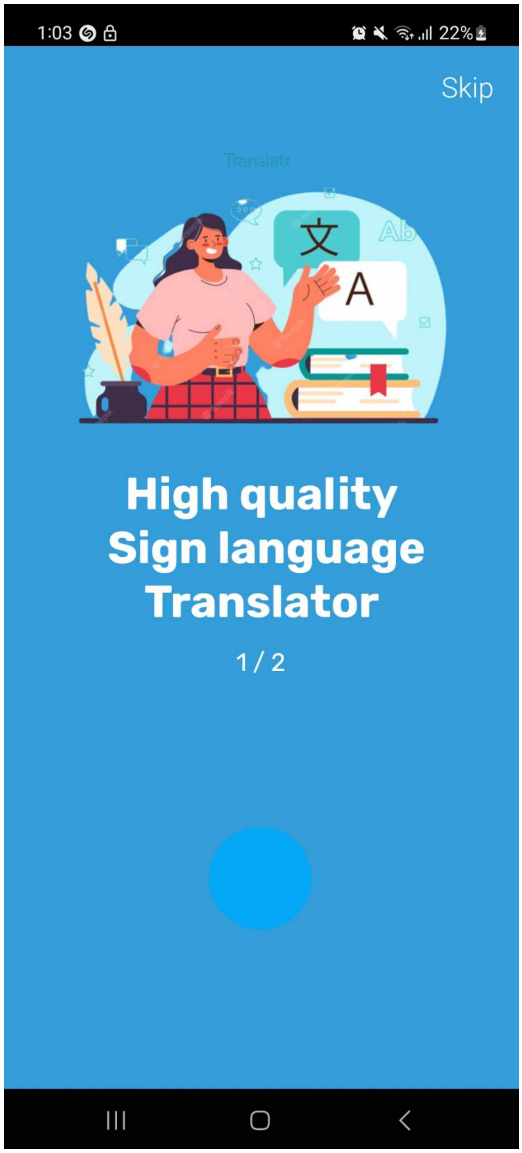


Figure 5.2: introductory page one.

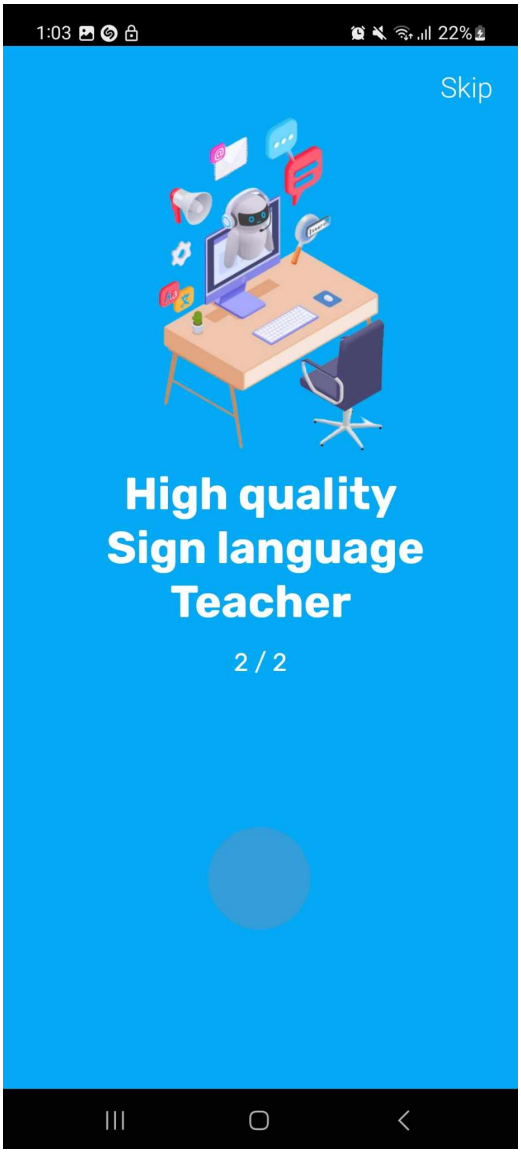


Figure 5.3: introductory page two.

5.2.2 Authentication pages

- User Registration Interface figure 5.4: The interface for user registration facilitates the sign-up process through the input of personal details. Users are prompted to input their complete name, email address, and to choose a strong and secure password.
- Login Interface figure 5.5: After registering, users can access the authentication interface to log in to their account. They are required to enter their email address and password to access the application.

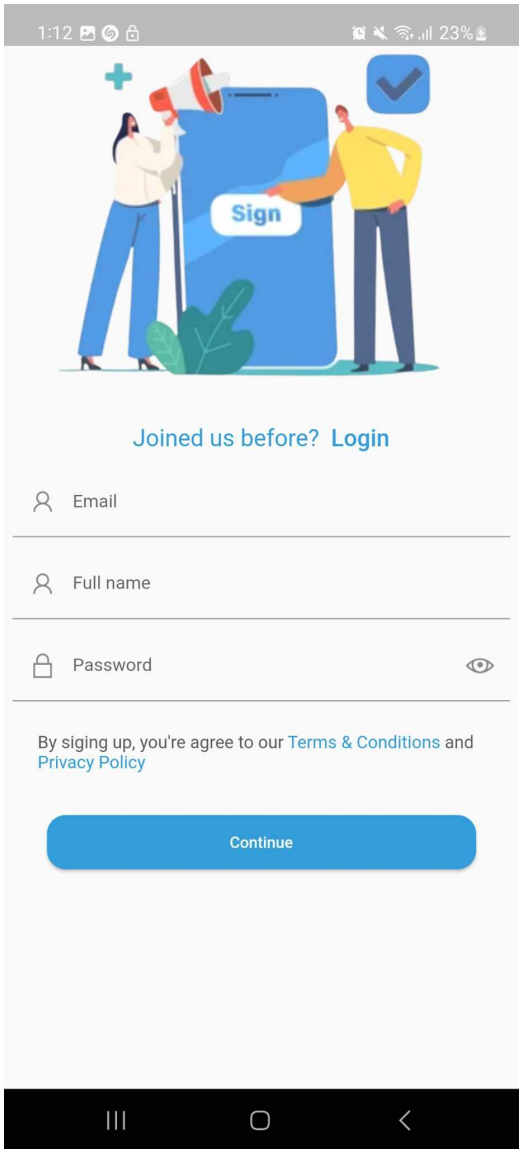


Figure 5.4: Sign up page

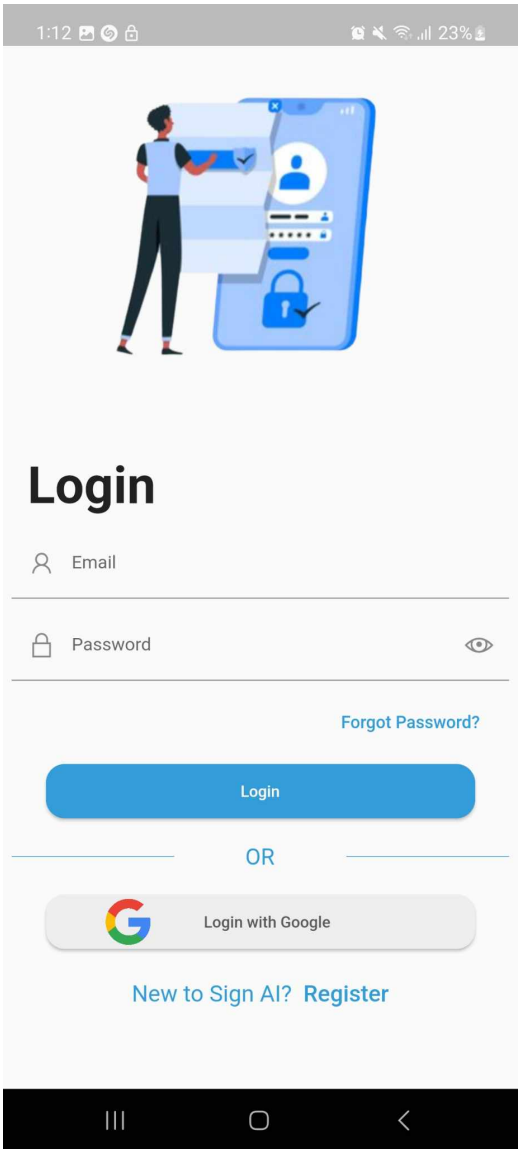


Figure 5.5: login page

5.2.3 Translation pages

After the user logs in, they will be directed to the main page for sign language translation "SignAI Translator". On this page, they can choose from various translation options, including:

- Tunisian Sign Language to text or voice
- American Sign Language to text or voice
- Text to voice
- Voice to text

Once they have made their selections, they can proceed to the translation room, by

clicking on the "Start" button, to initiate the conversation translation process.

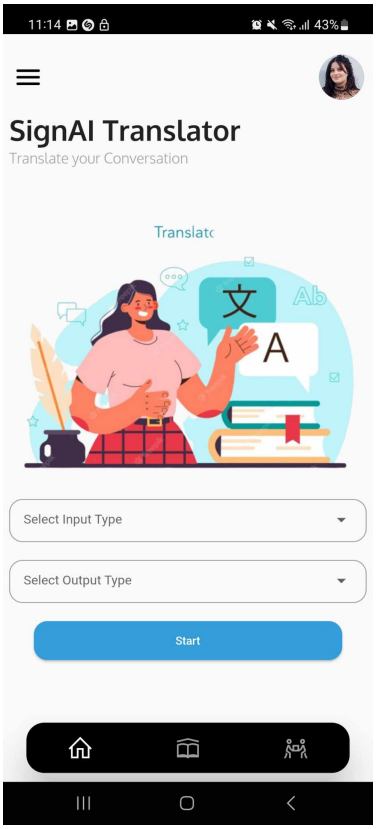


Figure 5.6: SignAI Translator main page

- Tunisian Sign Language to text or voice room Interface figure 5.7: In this interface, you will find a live camera view that captures real-time sign language gestures. These gestures are then translated into text, displayed below the camera view, or converted into voice output.

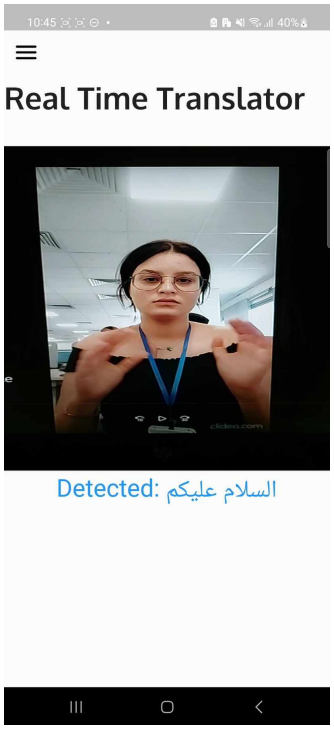


Figure 5.7: Tunisian Sign Language to text or voice room Interface

- American Sign Language to text or voice room Interface figure 5.8: In this interface, you will find a live camera view that captures real-time sign language gestures. These gestures are then translated into text, displayed below the camera view, or converted into voice output.

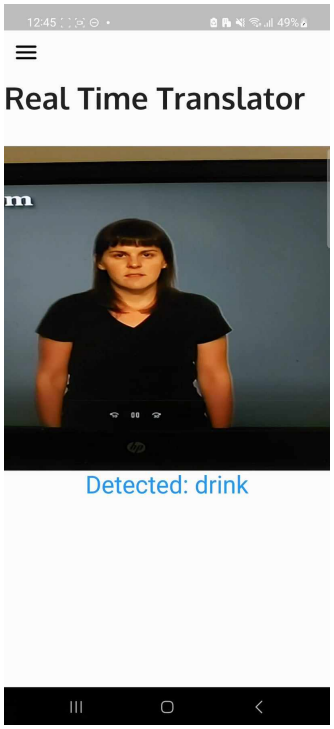


Figure 5.8: American Sign Language to text or voice room Interface

- Voice to text room Interface figure 5.9: Within this room, you will encounter a record button. By pressing and holding this record button, you can speak into it. Subsequently, the system will transform your spoken input into text.

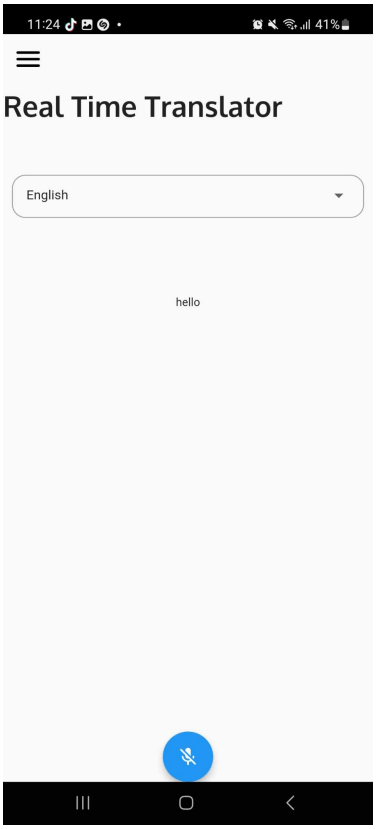


Figure 5.9: Voice to text room Interface

- Text to voice room Interface figure 5.10: In this room, you will discover an editable text field and a button. After composing your desired text, you can specify the language preference (French, English, Arabic), and then transform the text into voice output by pressing the "Send" button.

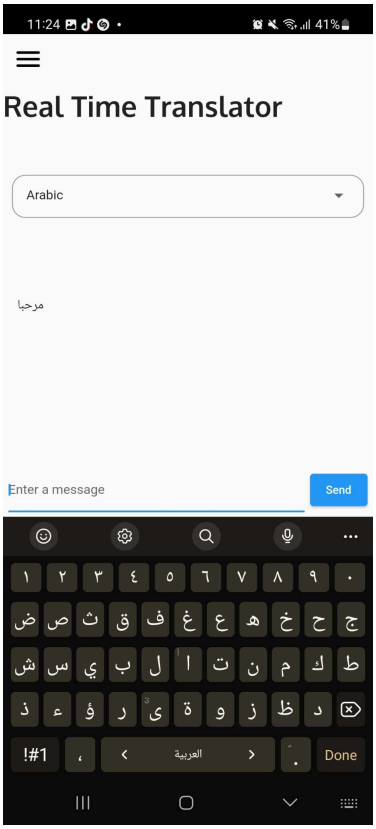


Figure 5.10: Text to voice room Interface

5.2.4 learn sign language letters pages

After the user chooses to learn sign language letters, they will be redirected to the main page for SignAI Teacher figure 5.11. On this page, the user can decide whether they want to engage in learning sign language letters, test their knowledge of sign language letters, or write their names with sign language letters.

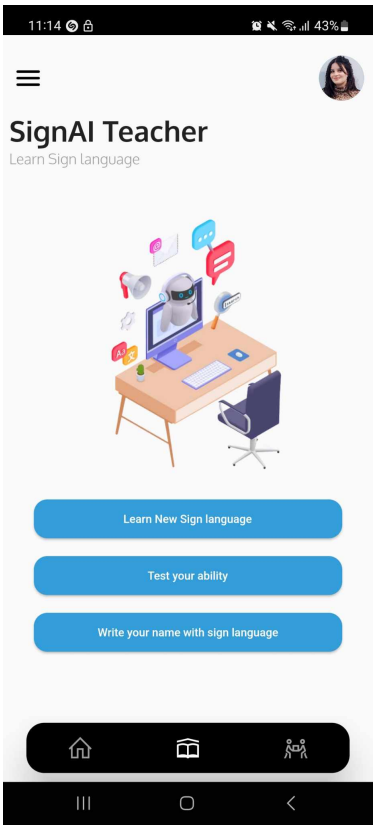


Figure 5.11: SignAI Teacher main page

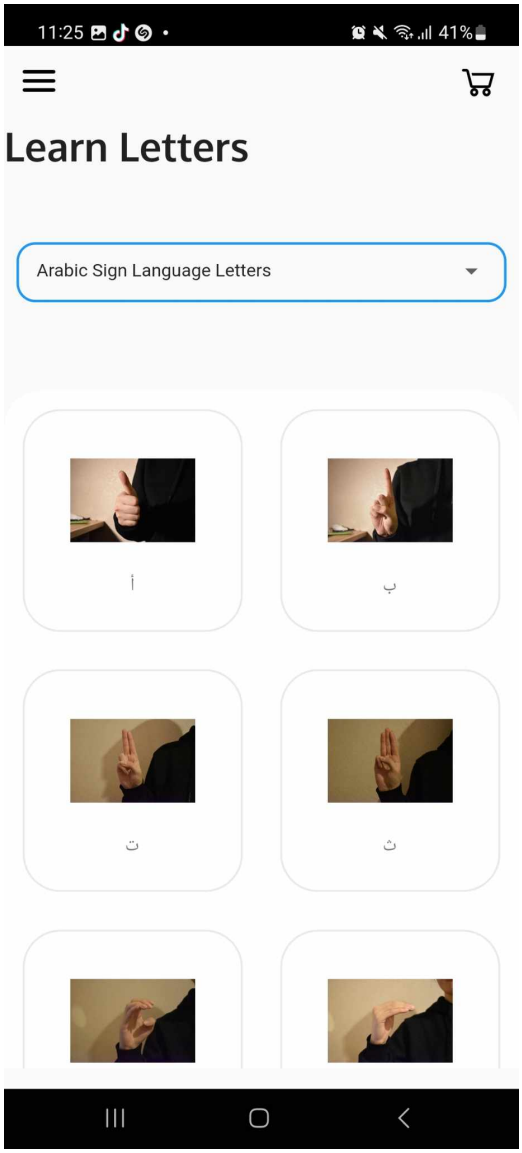


Figure 5.12: Arabic sign language letters page

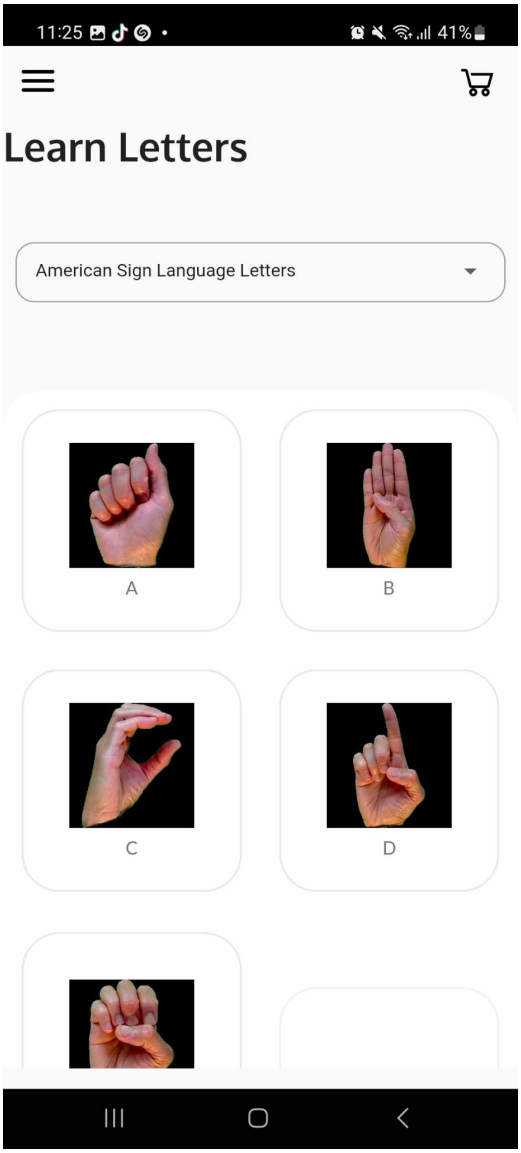


Figure 5.13: American sign language letters page

- Learn new sign language letters Interface figure 5.12 & figure 5.13: In this interface, you have the option to choose whether you want to learn Arabic or American Sign Language letters. The interface contains pictures that depict sign language letters, and each picture is labeled with its corresponding letter for easy reference and learning.

- Test your ability in sign language Interface figure 5.14:Within this interface, a camera view is displayed that detects hand gestures in real-time and classifies them into the correct sign language letter. The identified letter is then displayed below the camera view to assess the user's sign language proficiency.

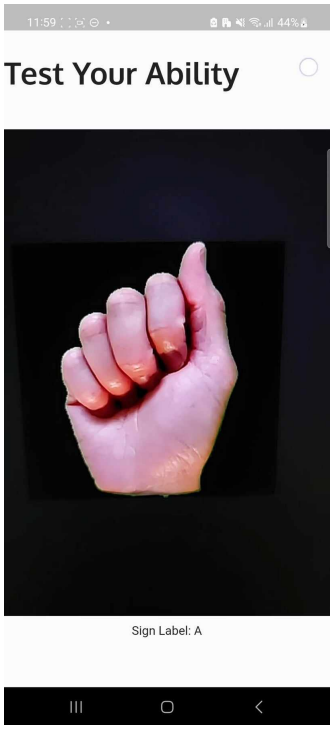


Figure 5.14: Test your ability in sign language Interface

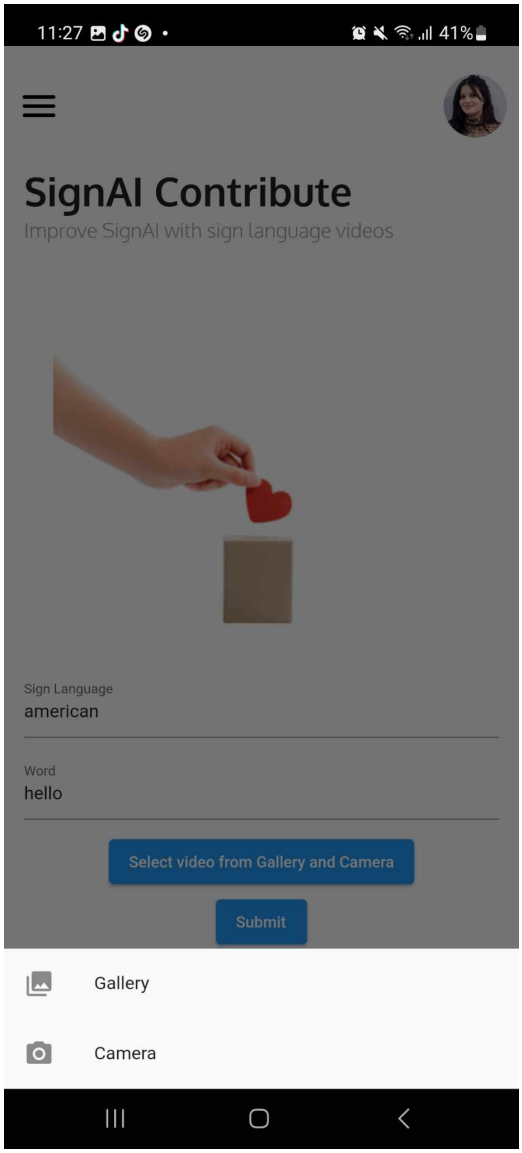


Figure 5.15: Contribution page one.

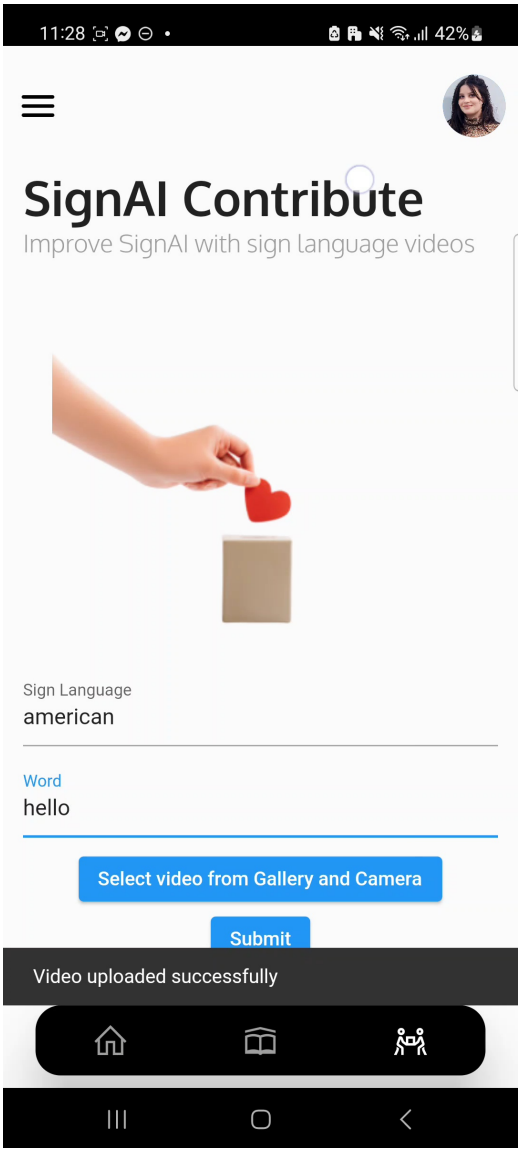


Figure 5.16: Contribution page two.

5.2.5 Contribution page

In this interface, users have two text input fields. The first field is for entering the name of the sign language, and the second field is for typing the word they wish to contribute. Below these input fields, there is a button that allows users to either record a video or select one from their gallery. This video should capture the sign language gesture corresponding to the word entered. Finally, there is a "Submit" button that users can click to submit their contribution.

Conclusion

In this final chapter, we have delved into the hardware and software environment of our project. Subsequently, we have highlighted the features of "SignAI" through illustrations representing the various interfaces of our application.

General Conclusion and Perspectives

In conclusion, this project represents a culmination of extensive research, rigorous analysis, and innovative development efforts. Through the exploration of the intricate world of sign languages, we have illuminated their cultural significance, linguistic complexity, and the challenges faced by deaf and hard-of-hearing individuals in mainstream society.

The core achievement of this project lies in the creation of three deep learning models, each tailored to address specific linguistic and communicative needs. These models offer real-time translation for American Sign Language, Tunisian Sign Language, and the classification of alphabets in American Sign Language. The development of an intuitive mobile application as an interface enhances the accessibility of these models, serving as a powerful tool for effective communication.

Furthermore, the inclusion of educational resources for American and Arabic Sign Languages within the application reflects our commitment to promoting language acquisition, cultural understanding, and inclusivity. This thesis underscores the importance of recognizing sign languages as valuable cultural assets and essential means of communication.

Looking ahead, our project presents several exciting perspectives. Firstly, we are committed to the ongoing refinement and enhancement of our models, integrating the latest advancements in machine learning to enhance accuracy and user-friendliness. Our future plans also include expanding the tool's language support to encompass a broader range of sign languages, thereby reinforcing our dedication to inclusivity.

Moreover, we are enthusiastic about collaborating with organizations and communities dedicated to empowering the deaf and hard-of-hearing individuals. Through these collaborations, we aim to ensure that our tool aligns with the specific needs and preferences of the community it serves.

In essence, our project signifies not only the culmination of our academic journey but also a significant stepping stone towards a more inclusive future. In this envisioned future, sign language will be universally recognized, celebrated, and readily accessible to all, reflecting the core of our mission and vision.

Bibliography

[1] United nation website. 2023.
Disponible on : https://www.un.org/en/observances/sign-languages-day [Access March-20-2023].

[2] Talan tunisie consulting. Talan Tunisie Consulting official website.
Disponible on : https://tn.talan.com [Access March-20-2023].

[3] Bootcamps.
Disponible on : https://www.ventraafrica.com/bootcamps [Access March-21-2023].

[4] Crisp-dm. IBM official website.
Disponible on : https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview [Access March-22-2023].

[5] Cnn architecture.
Disponible on : https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/ [Access April-20-2023].

[6] What is computer vision. IBM.
Disponible on : https://www.ibm.com/topics/computer-vision [Accès le 13-Mars-2019].

[7] Cnn architecture.
Disponible on : https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/ [Access April-20-2023].

[8] A guide to convolution arithmetic for deep learning.
Disponible on : https://arxiv.org/pdf/1603.07285.pdf [Published January 12, 2018].

[9] A closer look at spatiotemporal convolutions for action recognition.
Disponible on : https://arxiv.org/pdf/1711.11248v3.pdf [Published April 12, 2018].

[10] Traditional machine learning vs transfer learning. Transfer Learning en Deep Learning: Más allá de nuestros modelos.
Disponible on : https://medium.com/@josumsc/transfer-learning-en-deep-learning-mÃąs-allÃą-de-nuestros-modelos-c304c3a77d4 [Access April-20-2023].

[11] Residual connection. Deep Residual Learning for Image Recognition.
Disponible on : https://arxiv.org/pdf/1512.03385v1.pdf [Published December 15, 2015].

[12] Mobilenetv2n. MobileNetV2: Inverted Residuals and Linear Bottlenecks.
Disponible on : https://arxiv.org/pdf/1512.03385v1.pdf [Published March 21, 2019].

[13] Efficientnet b0 architecture. ResearchGate Logo.
Disponible on : https://www.researchgate.net/figure/The-architecture-for-baseline-network-EfficientNet-B0-is-simple-and-clean-making-it_fig4_347109918 [Access April-10-2023].