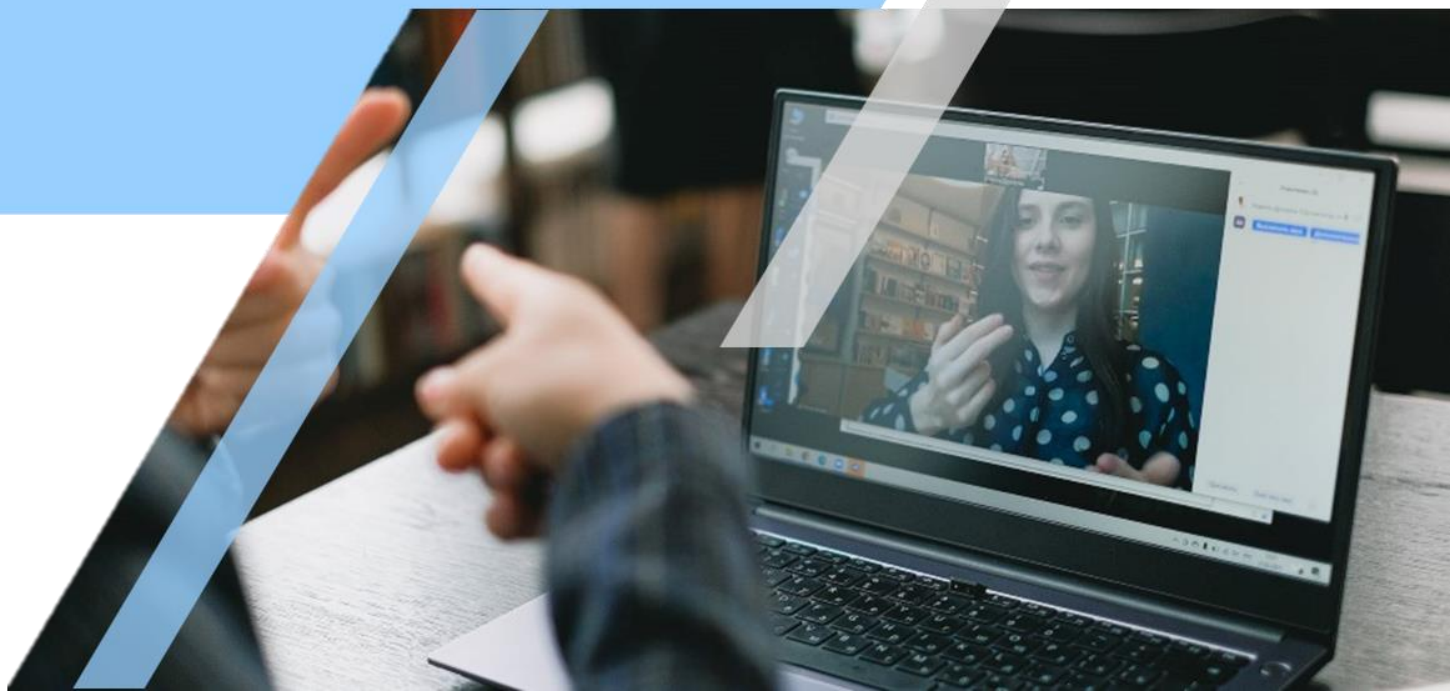
CS211: SYSTEM DESIGN AND PRACTICAL PROJECT

A Machine Learning-Powered ASL to Text and Voice Conversion Tool using Python for Deaf Communication

PROJECT REPORT



Dr Abdallah Sherief & Dr Hend ElMohandes

Presented by Group 10:

Hiba Hassan — Sara Mashaal

Anas Kanafani — Harsha Basavaraj Beth

Alireza Zabet — Zean Khazraji

The British University in Dubai

ACKNOWLEDGEMENT

We would like to thank everyone who helped make our project, "A Machine Learning-Powered ASL to Text and Voice Conversion Tool Using Python for Deaf Communication," a success. This endeavour would not have been feasible without their amazing assistance, counsel, and knowledge.

First and foremost, we would like to express our gratitude to Dr Abdallah Sherief and Dr Hend ElMohandes, two of our esteemed professors, for their unflagging support and encouragement during this endeavour. Their in-depth expertise, insightful criticism, and dedication have greatly influenced our understanding of system design and allowed us to explore machine learning.

We really appreciate everything that our teammates have done to help us conceptualise, create, and execute this project. This journey has been gratifying and pleasurable because of their cooperation, effort, and dedication to greatness. The success of our project has been strongly influenced by the special talents and viewpoints of each team member.

We thank The British University in Dubai for helping us by giving us access to the resources and facilities we needed for our research and development.

Last but not least, we would want to express our gratitude to our family and friends for their constant support, endurance, and comprehension during this effort. Their support and confidence in our skills have served as an ongoing source of inspiration.

Table of Contents

1. Introduction

1.1. Background and Motivation

In our everyday life, we use our hearing and speaking skills to communicate with each other. We, as humans, have developed numerous languages and communication styles for that cause. However, according to an encyclopedic entry written by the National Geographic Society (2022), there are still more than 72 million people on this earth who struggle with hearing and speaking, some are diagnosed as deaf and they are not able to hear at all, and some other are facing hard-of-hearing, which can be identified as partially deaf. It is also important to note that even though some people who were identified as deaf do actually speak, they do find it difficult to communicate using spoken language, due to them being unable to hear what is being said clearly or at all in many situations; furthermore, they have designed a couple of ways of communication that do not require any hearing or speaking skills.

For example, they have designed a fingerspelling technique, which can be used to spell letters, numbers and characters using finger gestures. They have also developed sign language for daily use, which uses hands, faces and arms to denote words and phrases from daily life, that one is very efficient in communicating between people rather than fingerspelling. This is called the WLASL or the Word Level ASL. Additionally, they have also developed sign language and fingerspelling techniques for every language, country and place. Hence, American sign language (ASL) is totally different from British sign language (BSL) and the fingerspelling techniques.

Because of this huge variety of signs and methods and languages, it is hard for the rest of the people to learn sign language and use it as a second language to communicate with, also the fact that it might be complicated to do some signs and remember them and be able to hold a conversation with sign language.

1.2. Problem Definition

ASL is a means of vital communication for individuals who are deaf or hard of hearing. The communication barrier between ASL users and those who aren’t fluent in ASL remains a major challenge for ASL users. For this project, we aimed to convert ASL to text and voice. This is crucial as deaf people are often times unable to speak as well. This project aims to bridge the gap between spoken language and sign language users. We hope to improve communication between people and make it easier for sign

language users to communicate with others. This automated system of converting fingerspelling ASL can benefit the deaf community.

For this project, we implemented various models to test and train the most suitable one to implement on our website, which can be accessed by everyone. We implemented a model for fingerspelling and another one for WLASL.

1.3. Literature Review

In the Sign Language Translation field, many people from around the world with different backgrounds tried to build this project with many different languages; they have also published research papers that support their findings and tests. These research papers paved the way for us as this research contributed insight on how to proceed with this project. We have collected the following research papers about the project:

1. *Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning.* conducted by Halder & Tayade (pages. 9-17, 2021):

   In this project, multiple datasets were collected for different sign languages, like American, Indian, and Italian for alphabets, and American and Turkey for numbers. Then they used mediapipe to pre-process and get the multi-hand landmarks out of the dataset. After that, they used Support Vector Machine (SVM) to make predictions. In the end, they got an accuracy of 99.15% for the ASL alphabets, 99.29% for the Indian alphabets and 98.19% for the Italian alphabets. They also got an accuracy of 99.18% for the ASL numbers and an accuracy of 96.22% for the Turkish numbers. They have found that even though MediaPipe is a very powerful tool to extract key points, it requires high computational power and a lot of training; depending on the dataset size.

2. *Sign Language to Text Conversion for Dumb and Deaf.* conducted by Luv, Singh, Chandra, Tauro and Gautam (2018):

   For their project, they have developed an ASL fingerspelling detection model using Convolutional Neural Network (CNN). At first, they collected a dataset of 26, 000 images for 26 classes, then they pre-processed and normalized it, then trained it using their CNN model that consisted of 2 convolutional layers, 2 pooling layers, 2 dense layers and at the end they had an output layer. In the end, they have reached an accuracy of 98.0%, however, their limitation was that the surrounding environment

might affect the testing process; like poorly lit areas and different kinds of backgrounds.

3. *An Integrated CNN-LSTM Model for Bangla Lexical Sign Language Recognition.* conducted by Hossain, Basin and Nahar (2020):

   In this project the team has developed a Bangala Sign Language (BSL) CNN-LSTM model; for a dataset they have developed that consists of 13,400 images for the 36 classes of Bangala fingerspelling signs. Then they have pre-processed the dataset, and after that they have integrated the CNN-LSTM model. The model was made of 2 convolutional layers, 2 pooling layers, 4 dropout layers, one flattens (time-distributed layer), one LSTM layer and at the end an output layer. The training accuracy of the model was 90% and the testing accuracy was 88.7%. The limitation of this model is that it cannot detect multiple lexical signs in a sequence which is to form words.

4. *Sign language recognition system for communicating to people with disabilities.* conducted by Obi, Claudio, Budiman, Achmad and Kurniawan (pages. 13-20, 2023):

   For this project, they have developed a dataset of 12,014 images divided into two sets, training and testing, each one is consistent of 27 classes for each sign, they have also used a Kaggle dataset, that is already pre-processed and normalized. After that, they have trained the model using a Convolutional Neural Network (CNN) model. They have used 2 convolutional layers, 3 pooling layers, 2 densely connected layers and at the end an output layer. The training accuracy for this model was 89.1%, and the validation accuracy was 98.6%. For this model they had plenty of errors causing reasons like low-resolution cameras, the Kaggle dataset had a problem with identifying between the letters H and G, so they would get mixed up sometimes and there might be some output error when have too many objects in the background or with a poorly lit area, also to form the letters into words it requires the hand to be in the same position for a couple of seconds which might affect the accuracy of the output word.

5. *Sign Language Recognition using OpenCV.* conducted by Juluru, Empati, Kallepalli and Anitha (pages. 606-610, 2022):

In this research, they have collected a dataset of word signs, and they have trained it using a Multi-class Support Vector Machine (MSVM). This model only trains 6 signs so it is very limited.

6. *Sign Language Recognition System using TensorFlow in Python.* conducted by Ashiksaibabu (2022):

   In this model, they used a dataset from Kaggle and pre-processed it, then it was trained CNN, the model consists of 2 convolutional layers, 2 max-pooling layers, 1 flatten layer, 2 batch Normalization layer, 2 dense layers and one dropout layer. The training accuracy for this model was 99.9% and the validation accuracy was 82.25%.

2. Project Methodology

2.1. Project Requirements

The ASL-to-text converter project aimed to develop a system capable of recognizing and converting American Sign Language (ASL) gestures into corresponding text. The project leveraged computer vision techniques, machine learning algorithms, and the Mediapipe library to detect and analyse hand gestures in real-time video input. We needed an adequate set of data on sign language to train the model and receive better results. We also required a well understanding of machine learning techniques, deep learning models and neural networks. In order to deploy the model into a website, we also needed an understanding of deploying techniques and should be well-versed in web development including frontend and backend development.

2.1.1. Software Development Environment

The ASL gesture recognition project was implemented using the Python programming language, which provided a rich set of libraries and frameworks to facilitate various tasks. Python libraries such as OpenCV, media pipe, pandas, seaborn, and Matplotlib played a crucial role in enabling computer vision functionalities, data pre-processing, and data visualization. OpenCV was utilized for capturing and processing live video input from the device's camera, while media pipe offered convenient solutions for hand detection and landmark tracking, essential for accurate ASL gesture recognition. Pandas aided in managing and manipulating the dataset, while Seaborn and Matplotlib were employed for creating informative and visually appealing data visualizations.

For the machine learning aspect of the project, the sci-kit-learn library was utilized, specifically the support vector machine (SVM) implementation, which is known for its effectiveness in classification tasks. Additionally, the pickle library was used for serializing and deserializing the trained SVM model, allowing for model persistence and easy deployment.

In terms of deploying the project onto the web, React and Tailwind CSS were employed to build the user interface, ensuring a responsive and visually appealing web application. The TensorFlow.js library played a significant role in hosting and serving the ASL recognition model on the server side, enabling real-time prediction of ASL gestures. Finally, Vercel was chosen as the deployment platform, providing a seamless and scalable hosting solution for the web application.

2.1.2. Requirement Specification Analysis

1. **Collecting the dataset of ASL gestures:** This requirement involves capturing a dataset of ASL gestures using a webcam. As a readily available dataset wasn't available, we had to make our own dataset. For the fingerspelling model, we collected images of the 26 English alphabets and for the WLASL model, we collected videos of certain words which we planned to incorporate into the model. The collected dataset included a diverse range of gestures to better train the model.
2. **Pre-processing and transforming the data:** Before training the machine learning model, the collected dataset needed to be pre-processed. This involves converting colour space, normalizing the data and extracting relevant information. The data should be prepared in a format suitable for training.
3. **Implementing a machine learning model for gesture recognition:** A machine learning model needed to be implemented in order to classify ASL gestures. This model should be able to recognize the pre-processed data as input and should be able to output the corresponding ASL gesture. The model should be trained using the collected dataset and validated to ensure its accuracy and reliability.
4. **Developing the text and voice conversion functionality:** Once the ASL gestures are recognized, they need to be converted into text and voice output. The system should have the capability to convert the recognized gestures into written text, providing a visual representation of ASL communication. For, audible communication, the recognized gestures should be converted to spoken words.
5. **Integrating all the components into a cohesive system:** The trained model was tested and validated then deployed into a website. This website had a user-friendly

interface that allows users to capture ASL gestures through a webcam and text and voice being displayed on the screen.

2.1.3. Use Case Diagram

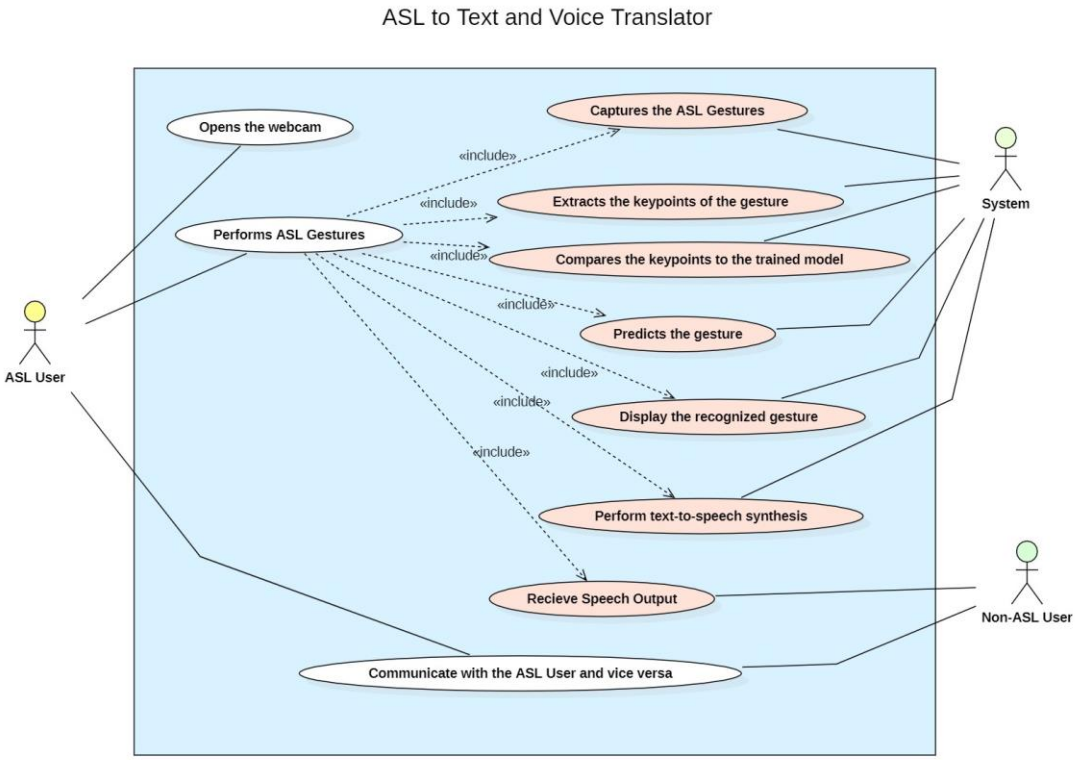


Figure 1 Use Case Diagram of the project

For this use case diagram, we have the primary actor who is the ASL User and the secondary actors are the System and the Non-ASL User. We have the three base use cases and have seven included base cases as every included use case is linked with "Performs ASL Gestures" using the webcam. Our system captures the ASL gestures, extracts the key points and compares them and outputs the predicted gesture. Our system should also be able to output the text and the spoken word along with the predicted gesture. In this way, we are able to establish solid communication between the ASL User and the Non-ASL User. Our system basically bridges the gap between the users and enhances communication within the deaf community and the spoken community.

2.2. Project Design

Our project aims to create and develop a gesture recognition system that enables effective communication between hearing-impaired individuals and others. The project's architectural design, user interface design, and system flow are all covered in-depth in this part.

- ***Architectural Design:*** The goal of the architectural design phase was to develop a modular, scalable system that smoothly connects different parts. In order to enable real-time gesture recognition, we used a client-server design. The server component held the ASL recognition model, while the client component was a website that users could access. Our goal was to create a user-friendly interface compatible with different screen sizes and devices.
- ***User Interface Design:*** An interesting and user-friendly experience was greatly influenced by the user interface design. The effectiveness of gesture visualisation was a key component of our design strategy.

  Users may record their hand motions using the camera on their device utilising the website's video input capability. The user interface included a live video stream that allowed users to see their movements in real-time. Users received rapid feedback as the system recognised and showed the relevant ASL gestures on the interface in response to their motions.

- ***System Flow:*** In order to provide a fluid and understandable user experience throughout the gesture detection process, the system flow was meticulously built. Our system's typical flow is shown in the following steps:
  - *User access the website*
  - *User grants permission for camera access and hand gestures are captured*
  - *Real-time gesture recognition*
  - *Display of recognized gestures*
  - *Optional text-to-speech conversion*
  - *Continuous gesture recognition*

2.2.1. Design Perspective

Design perspective is us taking into consideration the designing of a product, and adopting a certain mindset or strategy, it is applicable to a tangible good or an intangible service such as our software. It involves understanding the needs and requirements of our target users, which in our case, are people with hearing impairments, and developing a solution that meets those needs effectively. So, what design perspective did we adopt within the framework of our own project, then? Our project was created and put into action with the following guiding principles in mind: "user-centred, accessible, open source, and data-privacy free."

These are the main tenets on which we built our design perspective.

User-Centred Design: - Essentially, we kept in mind the user's needs at every step of the design process. Whether it was through a user-friendly UI or accepting user feedback regarding any and all issues, whether technical or for ease of use and appeal, to not only correct problems, but to also make it easy to use and nice for the user, at every point in the design process we thought of how it can affect users and how to make It a better experience for them, and if possible, have surveys and feedback to have a wide array of diverse opinions flow in a steady stream to us and allow us to make the best possible choices

Accessibility: - Being accessible and easily used by our, well, users. It is a standard albeit important principle all the same, unfortunately, our accessibility is limited due to the initial fact that it is only useful for American sign language, and not any user, but despite that, we have made our product in mind with the needs of customers, whether it be a high-contrast mode for those visually impaired, or a variety of settings a user can choose from, switch on or off, disable, enable to his liking, but it is a field that could be greatly expanded on as we have made mention near the end of this paper, but as it stands, it is in a satisfactory condition

Open Source: As a team, we accomplished our goal, but we are only a team still, there are thousands if not tens of thousands of people who might want to use our software, and we cannot possibly tailor it to every single need, or have in mind every single preference available as a setting for a person. Open source essentially allows us to finish what we started with accessibility, it is the ultimate accessibility option as any user with adequate knowledge may personally modify our tool however they wish to, one can list examples but there are so many of them, in short, perhaps some would call it a "cheap out" option of allowing users to make whatever changes we did not think of, but, we think otherwise as it is a positive choice that allows user creativity and freedom to do as they wish with our software!

Ethical Considerations*:* In accordance with our fourth tenet, our app does not track or collect any data about you, nor do we store any data you share with us in our database, such as pictures or videos of you. It is a practical as well as an ethical one, there are many reasons for doing so. Firstly, our respect of our consumers and users, it is merely a good public relations decision to respect a person's privacy and would certainly earn respect from any users. Additionally, it is a legal hassle, such as within the EU there is the "GDPR" General Data Protection Rights, it is a whole host of laws, articles,

exceptions, and rules, such as deleting data upon it being no longer useful, having our database be available to inspection to EU authorities, and so on. There are similar laws all over the world, and it is simply a hassle to deal with and not worth it, as there is no good reason for us to do so, so we have taken the easy option and entirely cut it out, as a problem both legal, ethical, and public perception of our product.

2.2.2. Process

Our project started with a meeting within ourselves to organize the process. Roles and responsibilities were assigned in that meeting, moreover, we talked about our ideas and how we could relate them to machine learning. Eventually, it was decided that ***American Sign Language (ASL)*** will be the focus of our project. In other words, we created a model which detects American sign language gestures and translates them into English words while also providing audio output.

To begin with, we started again with a small meeting to discuss quickly how we can perform this task. We had to make a model and a website to deploy it on. Our first priority was making a model, as we were simultaneously working on two different models, we had to collect the dataset suitable for those two models. While we were mainly focused on the models initially, we also progressed each week on bringing the website to completion. We divided the tasks among ourselves. Each person was designated with a role as we had to work on the dataset and we had to research the various machine learning models which work the best for our project. Furthermore, we had to look for methods to deploy machine learning into the website for easy access. We decided to go for an Agile Software development methodology for ease.

Initially, some members of the team focused on getting the dataset ready while the other team members concentrated on researching the models. As we progressed, we all understood more about how AI works and we saw first-hand the spine of machine learning.

Next stage, the only thing left was creating the website and deploying the model on it. The website was built using React, Tailwind CSS, tensorflow.js, and lastly Vercel. This stage of the process allowed us to understand how our model can be presented to the whole world through a simple website.

2.2.3. Table of Weekly Activities

Each week we had a formal meeting in which we discussed the requirements of each member and kept track of our progress. We used Clickup to keep track of the progress and to assign each member their roles in the project development.

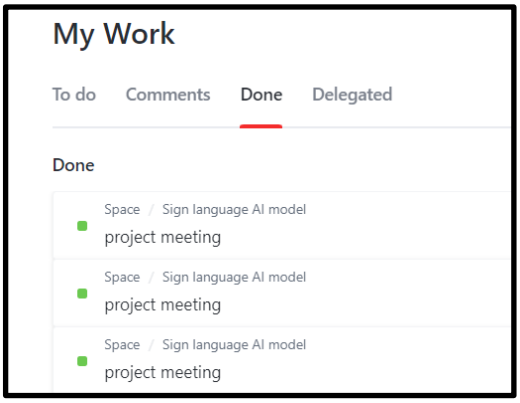


Figure 2 Week 1 progress

On the left you will see an image of the initial three meetings our team had specifically assigned for discussions for topic selection, research AI models and how to execute. The first meeting was on the 23rd of April, the beginning of this insightful journey. The next ones are a week apart from each other which as mentioned were for tracking progress.

Week 2: This week was dedicated to finding a dataset, as a trusted and well-known source we acquired a dataset from Kaggle which was later divided into testing and training sets. The dataset is made from labelled sets of videos of different words and gestures. However, one minor issue was that the dataset is huge and our computers don't have the space for it. For that reason, we took only some specific words to form multiple conversations as a demo for the ASL project. But soon realized that training videos may be much harder and time-consuming so we decided to go for an image dataset that we collected.

Week 3: The third week was for pre-processing the dataset, we achieved this by converting the images into 2D or 3D signals and matrices which allowed us to apply the desired hyperparameters. We tried various pre-processing techniques and then learned about the mediapipe library which already had computer vision models which can be used for extracting landmarks or key points.

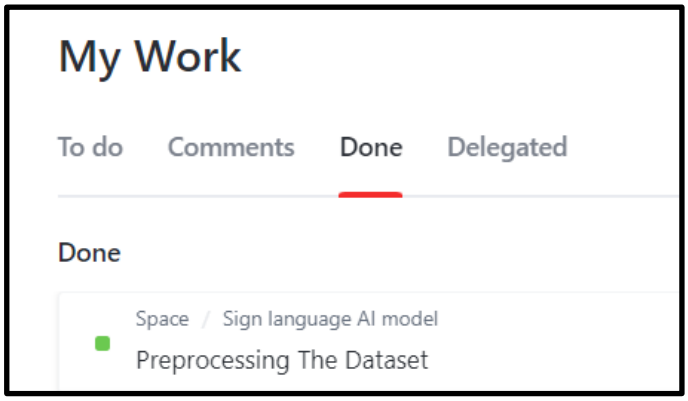


Figure 3 Week 3 progress

Week 4: We separated the dataset into testing and training and then the model was trained in the fourth week. According to their labels, we classified the images into their respective

classes or labels. After the training stage was done, our SVM model gave an accuracy of 99.8% which we were satisfied with.

Week 5: In the 5th week we worked on testing and validation. We used our testing dataset to observe how well our trained model performs with foreign data. Next, we made a comparison between the model and the result to see if that is what we required then validated our results using a chart. We also trained a WLASL model for research and we plan on expanding the functionalities of that model which we will be able to do with adequate computational power and time.

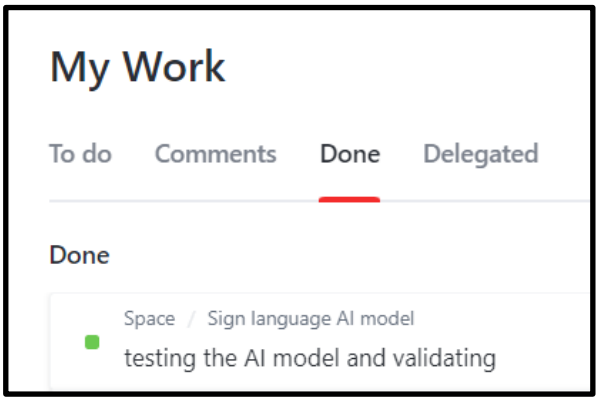


Figure 4 Week 5 progress

Week 6: We integrated the AI model into OpenCV, checking for the translation as a result of our training, and lastly making a website to deploy our model on. The website was built using React, Tailwind CSS, tensorflow.js, and Vercel for deployment.

Week 7: Lastly, all that was left was piecing the puzzle together, we connected all the components to a database server. Next and final stage, we did the final testing on the project to conclude it.

2.2.4. Roles and Responsibilities of Each Member

In this section, we would like to highlight the roles and responsibilities of each member who contributed to this project.

- **Hiba Hassan** – In charge of collecting and creating a suitable dataset for the model and along with pre-processing of the dataset suitable for the training process.
- **Sara Mashaal** – In charge of pre-processing the dataset suitable enough for the training process and was also responsible for training the model.
- **Harsha Beth** – In charge of testing the model and finding issues with it which contributed to enhancing the model.
- **Anas Kanafani** – In charge of the testing and creation of the website frontend and backend for model deployment.
- **Alireza Zabet** – In charge of evaluating and testing the model to get a sense of understanding to see where it can be improved.

- **Zean Khazraji** – In charge of further evaluation and comparison of the model to other existing models for deeper understanding.

2.2.5. Gantt Chart

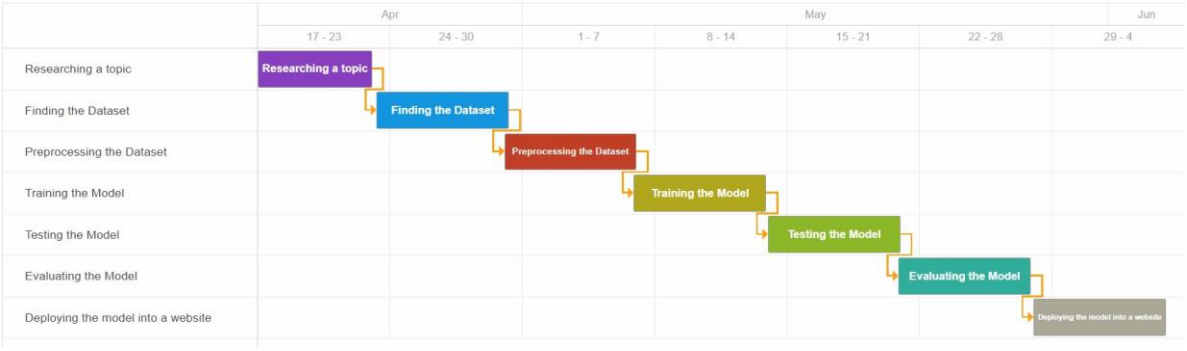


Figure 5 Gantt Chart representing our tasks in the project

Our project was expanded during the span of April 17, 2023, till the first week of June. We divided the project processes into snippets and assigned individual tasks for members to complete. Hence, we were able to finish this project within the deadline with ease. The project was well divided amongst the members and we combined everyone's work to make this project a success.

2.2.6. Project Diagram

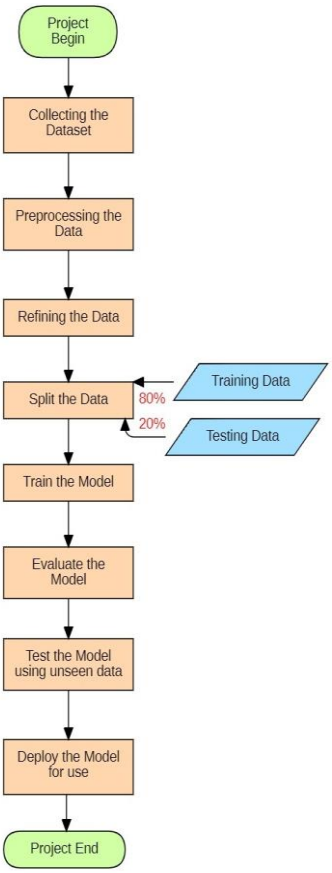


Figure 6 Project Diagram showing the various steps required for the project

The project diagram represents the workflow of this project as a flowchart indicating the processes involved while making this project. The success of each process led to the completion of the project. Here, each process is systematically arranged in order to clear and better understanding of our project.

2.2.7. Budget

In order to undertake this project, no budget was required, as all the technologies utilized were readily available without any associated costs. Furthermore, there was no necessity to utilize paid deployment or hosting services, given that the project could be exclusively executed on a local machine for the Python model. The website associated with the project would be deployed using a cost-free deployment service known as "Vercel."

2.3. Project Implementation

- **Fingerspelling American Sign Language Model**

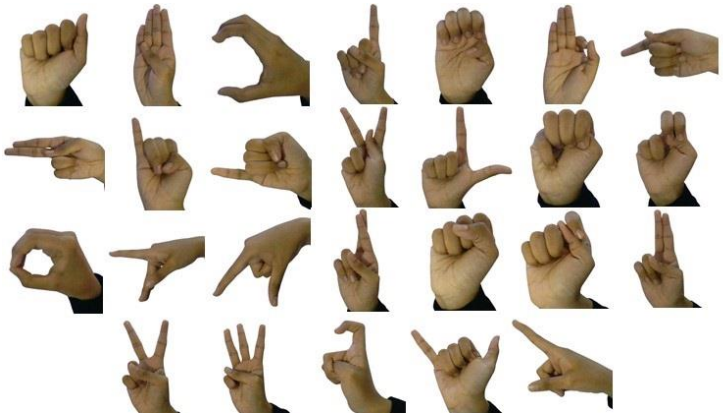


Figure 7 The dataset that was created for fingerspelling model of ASL

For the fingerspelling model, the first thing we did was collect our dataset by utilizing OpenCV to take 500 images for each of the alphabets, as a result, we had 13,000 images in our dataset. After that, we applied some pre-processing techniques such as changing the colour from BGR to RGB and flipping them along the Y-axis to align them correctly. The next step was extracting the important key points from the dataset, using a deep learning and computer vision library called ***Mediapipe***. Mediapipe is an open-source framework created by Google that offers a full range of tools and machine learning models for developing applications requiring perception tasks, such as object recognition, tracking, and hand tracking (Google Developers, n.d.). It provides a versatile pipeline-based architecture that enables programmers to build intricate processing graphs for examining and modifying multimedia data. From the mediapipe library, the ***MediaPipe Hand Landmarker*** function was utilized to detect the landmarks from the *right hand*. ASL is done mainly by using your dominant hand and since the majority of people are right-handed, we used the right hand for our model. Of course, it is possible to make the model for the left-hand users as

well, however, that will require some changes in the dataset as we would need to include images of left-handed gestures into the dataset as well. The hand landmark model bundle locates 21 hand-knuckle coordinates as key points inside the identified hand regions. The landmarks also known as key points were then extracted from all three axes, x, y and z and then converted to a NumPy array. If no hand gestures were recognized the arrays had the value of zeros.

The concatenated NumPy array of each image was saved in a ***CSV file***. To sum it up, all the key points that were extracted using *mediapipe hands* were converted into a CSV file. After that, the dataset is prepared by separating the features and labels which are essential for training a machine-learning model. This separation allows for further analysis and model training using the scikit-learn library. Following the dataset preparation, the training begins. The dataset is split into testing and training and 20% of the data is used for testing while the rest 80% is used for training.

Figure 8 Hand Land-marks detected by MediaPipe

Following this stage comes training. We trained it using the ***SVC model***. SVC stands for "Support Vector Classifier", which is a type of ***Support Vector Machine (SVM)*** algorithm used for classification tasks. The hyperparameters of the SVM model were set during training. According to the SVM documentation (scikit-learn n.d.), the C parameter is the penalty parameter for the error term in the SVM model while the gamma parameter is a parameter for the radial basis function as it defines the influence of each training example and affects the shape of the decision boundary. Spectating these parameters with *trial and error* we managed to get the best result possible. The performance of the model is evaluated using various metrics, including the F1 score, recall score, precision score, and accuracy. The F1 score is considered the mean of precision and recall and is used to measure when both precision and recall are equally important. Recall score is called the sensitivity of true positive rate and is a measure of how many actual positive instances are correctly predicted by the model while precision is a measure of how many of the predicted positive instances are actually true positives. Accuracy is the measure of how well a model predicts the overall instances in a dataset.

```
In [8]:  cf_matrix = confusion_matrix(y_test, y_pred)
         f1 = f1_score(y_test, y_pred, average='micro')
         recall = recall_score(y_test, y_pred, average='micro')
         precision = precision_score(y_test, y_pred, average='micro')
         f1, recall, precision

Out[8]: (0.9984615384615385, 0.9984615384615385, 0.9984615384615385)

In [9]:  from sklearn.metrics import accuracy_score

         accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy:", accuracy)

         Accuracy: 0.9984615384615385
```

Figure 9 F1 score, recall score, precision and accuracy of the SVM model

As the figure states, we got exceptionally high results as our F1 score, recall, precision and accuracy score are **99.84%**.

Additionally, the confusion matrix's heatmap visualisation is created using the seaborn library, providing a graphical representation of the model's performance across several classes.

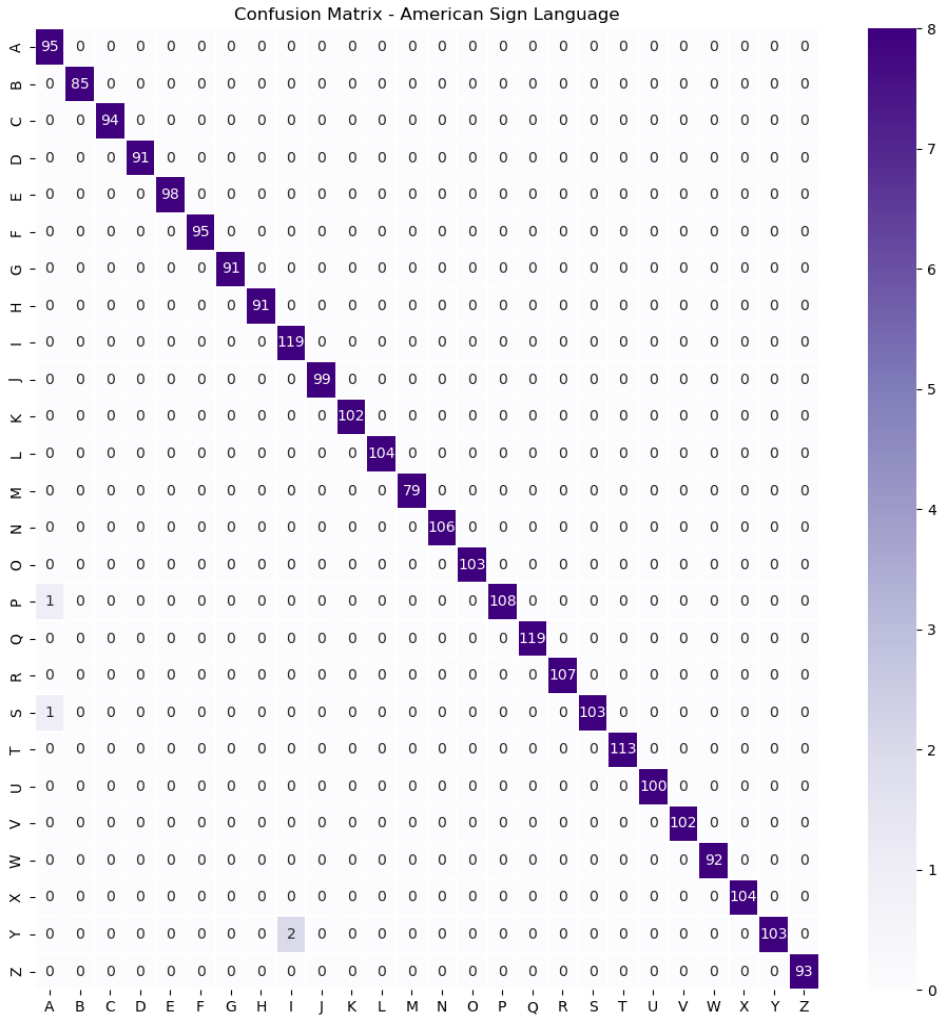


Figure 10 Confusion Matrix of the fingerspelling model of ASL

The trained SVM model was saved using the ***pickle module*** allowing it to be loaded and used for testing and analysis.

We can compare our model with the existing ASL fingerspelling models to get a better understanding of the capabilities of our model.

Algorithm / Method	**Accuracy**
Deep CNN (Das, Ahmed & Ali 2020)	94.3%
KNN (Saquib & Rahman 2020)	96.14%
Random Forest (Saquib & Rahman 2020)	96.13%
ANN (Saquib & Rahman 2020)	95.87%
SVM (Our Model)	99.8%

Figure 11 Comparison with other existing models

As can see, our trained SVM model has the highest accuracy among the existing models for ASL fingerspelling.

Moving forward, as per our plan we had to deploy the machine learning model onto a website to showcase it. This way the model can be accessed by anyone on the web. We built a website for the model using a popular JavaScript library called React, a utility-first CSS framework called Tailwind CSS, a JavaScript library for deploying machine learning models called tensorflow.js, and Vercel for deployment. We tried operating IBM Cloud to deploy the model however it requires a COMPANY account meaning they ask for the VAT number which we don't possess. We also tried selecting the PERSONAL account, unfortunately, we reached a dead end there too because it needs a credit card that's valid in the United States.

As an alternative, we looked at Google Cloud, which offers a service which is approximately $300. Hence, we discussed and decided to use an already deployed model as a proof of concept to show that if we had the needed resources, we could deploy our model into the website and it would look like this.

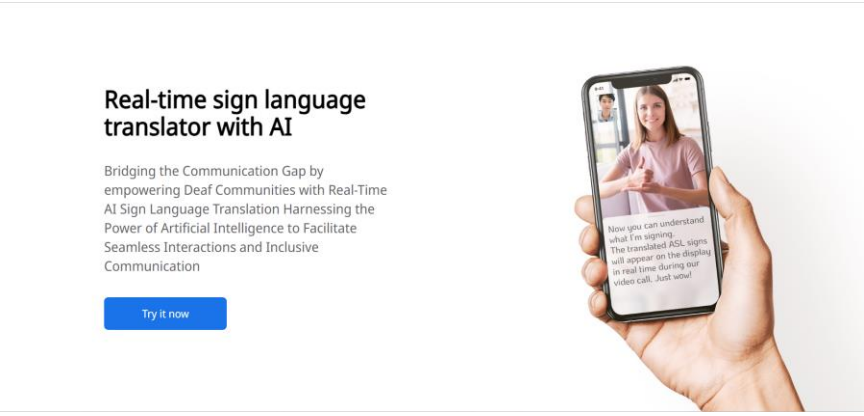


Figure 12 Display of our Website

- **Word Level American Sign Language Model**

As previously mentioned, the sign language that is more commonly used is the WLASL and it uses words for each gesture. The thing about WLASL is that each gesture is a video. Pre-processing videos requires a lot of computational power so we made a WLASL recognition system with only a few words.

As for the dataset of this model, we recorded ourselves doing some signs and recorded 30 videos per sign. Unfortunately, this data wasn't enough to give us good accuracy so we decided to record 150 videos per sign. The reason was there wasn’t a good dataset online that we could use for this project. The Mediapipe Holistic model was used for this. This model lets you combine, pose, face and hand landmarks but for simplicity, we used only the hands from the holistic model. Image processing techniques such as colour conversion from BGR to RGB and vice versa were done. Then we use the draw landmarks functions to visualize the key points in our left hand and right hand. Then a function returns this concatenated array of key points, which represents the coordinates of the detected hand landmarks. If any hand landmarks were not detected, the function returns an array of zeros in their place. Then we decided to collect the dataset to train this model. We created a folder to save actions or gestures of WLASL as frames of concatenated NumPy arrays. This allows for collecting training data for action recognition models. After that, we split the dataset into testing and training for model training.

For training, we use a sequential model for action recognition from TensorFlow. The model architecture consists of three LSTM layers, followed by three dense layers, culminating in a SoftMax output layer. The LSTM layers capture temporal information, while the dense layers learn to classify actions. The model is designed to recognize various actions based on input data.

An EarlyStopping call-back from the TensorFlow library is used to enhance the training process of the action recognition model. EarlyStopping is a valuable technique that monitors a specified metric, in this case, the validation loss, during training. Its purpose is to prevent overfitting and optimize the model's performance. The parameters for monitoring and terminating the training process are set by importing the EarlyStopping call-back and creating an instance of it. The "monitor" option is configured to watch the validation loss, and the "patience" parameter specifies the number of epochs to wait before quitting if the loss does not decrease. As expected, the epoch stopped at 102 because of EarlyStopping and prevented overfitting.

```
Epoch 102/2000
23/23 [==============================] - 3s 117ms/step - loss: 0.0875 - categorical_accuracy: 0.9649 - val_loss: 0.2215 -
val_categorical_accuracy: 0.9474
```

Figure 13 Epoch stopping at 102 due to EarlyStopping

A model summary was obtained from the training

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 64)            38144

 lstm_1 (LSTM)               (None, 30, 128)           98816

 lstm_2 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 5)                 165

=================================================================
Total params: 192,773
Trainable params: 192,773
Non-trainable params: 0
_________________________________________________________________
```

Figure 14 Sequential Model summary

After training, the trained model is saved using the Keras save method. This method allows you to serialize the model architecture, optimizer configuration and learned weights into a single file. The accuracy score of the model was calculated to be **92.1%** for the selected words.

```
In [51]: accuracy_score(ytrue, yhat)
Out[51]: 0.9210526315789473
```

Figure 15 Accuracy Score

- **Fingerspelling British Sign Language Model**

Figure 16 Dataset before pre-processing

For this model we have collected a British fingerspelling lexical signs dataset of 7,800 images divided into 26 class and each class contains 300 images. The dataset was collected using OpenCV and then it was saved and labelled in distinct classes. The next step was pre-processing, we had to apply background extraction, rescaling and resizing, Gray scaling and data augmentation.

Figure 17 Dataset after Pre-processing

After we pre-processed the dataset; we converted the image data into NumPy arrays that we normalized so the values were between 0 and 1. Then we mapped each class to its corresponding label.

When building the Convolutional Neural Network (CNN) model, the model consisted of two convolutional layers, two max pooling layers, one flatten layer and two dense layers one of them is an output layer as well.

We have used Adam optimizer and sparse categorical cross entropy for testing and validation.

For training data augmentation, we used the ImageGeneration class, with batch size 32 and 10 epochs.

Eventually, we received the accuracy as the following:

```
Epoch 10/10
245/245 [==============================] - 20s 82ms/step - loss: 0.0442 - accuracy: 0.9858
```

Figure 18 Accuracy of the BSL model

2.4 Project Testing and Evaluation

- **Fingerspelling American Sign Language Model**

After the implementation of the model, we had to test to see if our model correctly predicts on a live webcam when hand gestures are used. Using OpenCV the frames are collected from our webcam and then the image processing techniques of colour correction and flipping along the Y-axis are done. Mediapipe is used to detect the hand landmarks within each frame. The SVM model is loaded from the previously saved pickle module and the detected landmarks are taken as input to predict the ASL gesture, in our case an alphabet. To make the system accessible to individuals with hearing impairments, the recognised gestures are converted to spoken words using text-to-speech conversion. The pyttsx3 library is employed to initialize the text-to-speech engine and the speech rate is set to 150 words per minute providing a natural and understandable auditory output. The model is tested by checking the last 10 consecutive frames. If the last 10 frames are corresponding to a character in our trained model it prints it out along with the spoken character. The characters are concatenated as words on screen and the words can also be spoken aloud for seamless communication. The words can also be reset by using a key and more words can be added for testing. Here is a demonstration of our project.

Figure 19 Testing of our fingerspelling ASL model

Here, the word “HELLO” is printed out and spoken out during this testing.

- **Word Level American Sign Language Model**

For testing and evaluating the sequential model, the real-time prediction is done using OpenCV. The key points are extracted from the frames and detected on the basis of the last 30 frames shown. If the last 30 frames are shown to match a gesture of WLASL, the predicted word is displayed on the screen.

Overall, this enables real-time gesture detection, prediction, and the formation of sentences based on the recognized actions. It provides an interactive and dynamic visualization of the detected gestures, enhancing the usability and interpretability of the system.

- **Fingerspelling British Sign Language Model**

For testing this model, we used real-time evaluation and testing using OpenCV. We have imported the model as a pickle model into the testing code and ran the video frame.

However, the training accuracy was high, the testing accuracy was pretty low, which didn't allow us to get the desired output from the model while testing it, and that is probably due to the limited number of images per class, which resulted in some underfitting.

3. Discussion, Conclusion and Future Work

3.1. Reflection on what you learned from the project

This project was a chance to learn not only innovative technology but also a team-building exercise and a way to explore the development of software in what would one consider “niche” spaces, that still came out useful.

Practical knowledge of computers was of great value, as we gained real-time experience with the OpenCV library to process and analyse video input, which was used for detecting and tracking hand gestures in real-time, which would then would, understandably be used got sign language, machine learning played a part too as it

Additionally, to advertise and make our project easily accessible to as many people as possible digitally, we have made use of web development, it is a mix of design and practical coding skills, to both write the code and make a site easy on the eyes, that is both interesting, informative and importantly concise, allowing the users to easily read, understand, and then use our product with no long hassle

For one, effective communication and team management is vital, and this lesson has only been reinforced with this project. As a project of course, naturally, we acquire practical knowledge, but you can have the world's best programmers working together and if they don't communicate with one another, they will likely fail, despite great individual skill and knowledge. Therefore, effective team management, duty allocation, and constant and efficient communication played a key role in this project, and will always play a role in any future project. It is perhaps not a hard lesson to learn, but what makes it hard is the continued willingness of a person to set up and continue to engage with one another when working, being able to solve problems faster, and ultimately produce an efficient, bug-free, problem-free, product/software an acceptable time, lest it gets stuck in the infamous "development hell" where due constant issues meaningfully collectively contribute and finally finish said software/product.

3.2. Your comments on the testing and evaluation and feedback from users

Actively seeking feedback from users our first and biggest priority was to fix complaints about "core" issues, such as the software incorrectly translating or misinterpreting ASL. A rather straightforward affair that does not need much in-depth explanation, basic troubleshooting, and the type.

Some of the positive feedback from users typically included the intuitive and user-friendly nature of the project, including a simple and workable user interface. Of course, it could always be optimized, but in its current state, it is good to leave it as it is.

Interestingly, ASL has slang, much like normal language. So, the aid of users mentioning a lack of slang and further inclusion of it, even though it was a basic feature, relatively speaking, merely added more to the existing interpretation and vocabulary. It can be used as a form of accessibility and user-friendly design, perhaps not solving a problem, but definitely making this a slightly better software than it previously was.

Perhaps we could theoretically receive some complaints of this not being not "proper" sign language, but much like with real language, even if we don't consider slang to be proper language, does not exclude its use and thus preferable integration into our software to increase the use and enjoyment of it. To allow it to be useful in both professional and casual environments.

All in all, the feedback was crucial as without ASL specialists on our team, users who regularly encountered, needed, and worked with translations were critical to ironing out any existing flaws.

3.3. The limitations of the project

The project could potentially suffer from a variety of drawbacks in some cases that would be challenging to fix, for example, not everyone has access to the best cameras or a good internet connection. Or there is some particular nuance missing that cannot be interpreted, much like emotional nuance in text-to-speech models that simulates human speech, as for software to emulate such emotion, it can be just as hard for it to interpret it.

What do we mean by this? Simply put, our initial primary objective was about simple, direct translation, from a total lack of emotion. Imagine it akin to saying “Apple”, with no inflexion, no tone, no emotion, or anything else of the kind. That is what our project does, except the other way around of course, in interpreting sign language as simply so and nothing more. This could perhaps be seen as an issue, as American Sign Language or ASL is a complex language like any other, even though it may lack the inflexion that comes with speaking, there is still some certain emotion to it, from predefined gestures and signs that software would simply fail to catch, and we are not entirely sure whether the extra effort would be worth it as it seems like a monumental task, for only a little additional gain in the aforementioned tones and emotions that would come associated with ASL.

Of course, this was one of the more “far out” and advanced issues, if we can say that, some of the more immediate and perhaps more useful solutions needed to certain problems would be a real-time system that interprets ASL, and reads it out in a text-to-speech format, the simple way would be the standard Alexa or Bot voice you commonly hear online or from your phone, a simple 1:1 reading of whatever the person signed at a reasonable pace. This would require fine-tuning and plenty of troubleshooting with surveys and feedback, but ultimately perhaps one of our more attainable goals had we had more time to work on it. As with live-streamed ASL communication and translation could be a great boon to many people, and greatly ease communication between sign language users and non-sign language users.

The other problem at hand is general ease of use, as we have mentioned not everyone has access to an ultra-high definition camera and the best internet connections. If theoretically, we could implement real-time ASL interpretation such as during video calls, the next issue at hand would be, making the system more robust, being able to capture and understand

motion not only during real-time, or perhaps even estimate and translate those hand gestures during imperfect conditions, such as perhaps, low quality, varying lighting conditions, imperfect internet connection, and so on, or having a wider arc from which upon it can capture said footage. It would be a somewhat slow development admittedly and is rather small and simple in terms of results, and direct improvement.

According to the British Sign Language (BSL) model, we have encountered numerous limitations of the model. One of the underfitting issues was caused by having a CNN model that requires a bigger dataset than ours, so for further works we can double the number of images per class so it can be trained easily and have a bigger variety of data. Another factor that limited the model was the simplicity of the CNN model, so for future engagements we might choose to add a LSTM layer and add another Convolutional layer to the model, to have something like the Bangla sign language project that was mentioned earlier.

Lastly, as a minor development, ASL is not the only sign language out there, it’s a very simple yet practical development to include inter-sign language translations as well, thoroughly interconnected, from American to Spanish SL, or from French SL to Italian, of course, it would be a great undertaking in terms of language understanding, but in we already have the existing foundations since we already are translating ASL. Its biggest benefits would be accessibility and ease of use, something that many software applications constantly strive for to achieve and so would ours under more favourable circumstances.

In short, while the core objective has been achieved, there are still plenty of rich additions that could be made and further expanded upon to make this software truly top-of-the-line in quality and usefulness.

4. References

Basnin, N., Nahar, L. and Hossain, M. (2020) *An Integrated CNN-LSTM Model for Bangla Lexical Sign Language Recognition*. rep. Available at: https://www.researchgate.net/publication/347443925_An_Integrated_CNN-LSTM_Model_for_Bangla_Lexical_Sign_Language_Recognition (Accessed: 12 June 2023).

Das, P., Ahmed, T. & Ali, M. F. (2020). Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network. *2020 IEEE Region 10 Symposium, TENSYMP 2020*. Institute of Electrical and Electronics Engineers Inc., pp. 1762–1765.

Google. (n.d.). *Hand Landmark Detection.* Retrieved from https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

Haldera, A. and Tayadeb, A. (2021) *Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning, ijrpr.com*. Available at: https://ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdf (Accessed: 12 June 2023).

National Geographic Education. (2022). *Sign Language*. Retrieved from https://education.nationalgeographic.org/resource/sign-language/

Saquib, N. & Rahman, A. (2020). Application of machine learning techniques for real-time sign language detection using wearable sensors. *MMSys 2020 - Proceedings of the 2020 Multimedia Systems Conference*. Association for Computing Machinery, Inc, pp. 178–189.

scikit-learn. (n.d.). *Support Vector Machines (SVM)*. Retrieved from https://scikit-learn.org/stable/modules/svm.html

Singh, N. *et al.* (2018) *Sign Language to Text Conversion for Dumb and Deaf*. rep. Available at: https://github.com/luvk1412/Sign-Language-to-Text/blob/master/Project%20Report.pdf (Accessed: 12 June 2023).