Translating Sign Language to Speech

Final Report

Group 1:
David Plotkin
Mervin McDougall
Francisco del Castillo

DECEMBER 8, 2024

Project Overview

Thanks to the increased investment and development of Artificial Intelligence and Machine Learning, we are now able to interact with computers in a new way. We can enunciate commands to compose an email or respond to a text message via our digital assistants using verbal prompts. We can also initiate and hold verbal conversations with our smart devices like Siri, Alexa, and Google Assistant when we have pressing questions about the weather, traffic, or the latest headline news. Moreover, we can use these digital assistants to translate real-time communication with someone who speaks a foreign language. These features free a user from the use of keyboards and bottomless menus allowing him or her to interact with the technology naturally. However, there are some groups for which the benefits of these technological advances are currently unreachable. Our project aims to bridge this technological divide by allowing smart devices to translate sign language in real-time and give a voice to those who are unable to speak.

Currently, none of the major manufacturers of personal assistants provide a means of translating from sign language to speech. The most well-known personal assistant, Siri, requires a user who has speech disabilities to type their response using an on-screen keyboard and have Siri convert it to a voice (Apple). The other well-known assistant Alexa, from Amazon, requires a similar means of interaction, requiring a user to make use of an on-screen keyboard which can be operated remotely or by touchscreen to engage with Alexa. It should be noted that Amazon limits its text-to-speech facility to Fire Tablets only (Amazon). Lastly, Google's Assistant, with Android-enabled devices, does not provide a means of directly translating from sign language to speech (Google). But it does provide a setup video for your Android device which is presented using ASL (American Sign Language) (Google)

Our goal is to empower individuals without speech to communicate more organically and intuitively, without being tethered to a keyboard. This enables them to engage in coherent, fluid conversations, whether with colleagues at work, a private doctor during a video chat, or even a treasured family member at Thanksgiving table. Through our application, we are proposing a new approach by leveraging Artificial Intelligence (AI) to translate sign language into text or audio. This provides a more familiar and natural communication experience for those who are speech impaired.

Our application - the ASL (American Sign Language) Translator named "Inner Voice" - allows a user, through sign language, to issue commands to an AI, communicate with family or friends, and participate in social settings and society in a second-nature way. We will consider the project successful if we can deliver a complete gesture-to-speech software solution that is both practical for daily communication and easily downloadable onto any computer.

Prediction Inference and Other Goals

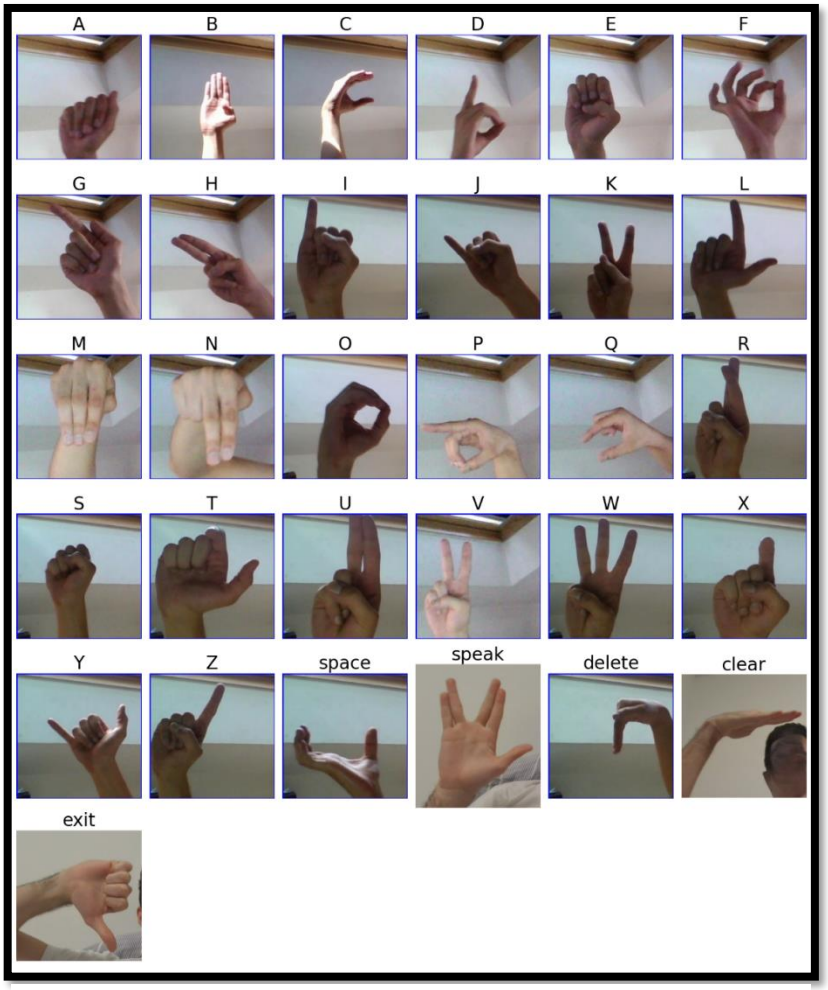

Figure 1

The goal of this project is to build an application that can interpret American Sign Language (ASL). Although the language consists of gestures representing complete words and grammar, we chose to limit the scope of the project to the ASL alphabet. A reference for the valid gestures used by the application is shown in Figure 1. The alphabet is accompanied by control gestures which allow the user to also control the application through gestures.

To ease the process of training our model, we developed a pipeline process of converting the gestures into coordinates by passing through Google MediaPipe Hand Landmark API (Google). A diagram illustrating the pipeline is shown below.

The images were funneled through MediaPipe. The returned results were compiled into a table, and these values were used to train our fully connected model.

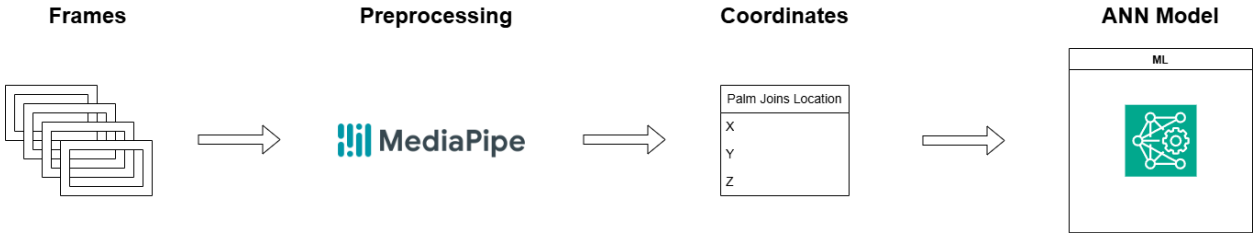

Figure 2

We have provided a special section covering the model selection below to discuss that process further.

Data Exploration

The dataset, retrieved from Kaggle, contains 87,000 images of American Sign Language (ASL) gestures categorized into 29 classes: 26 alphabetic letters (A–Z) and three additional classes (“space,” “delete,” and “nothing”) (Nagaraj). Each alphabetic class contains 3,000 images, each size 200x200 pixels in RGB format.

The image dataset was preprocessed by a third-party API, Google MediaPipe, to create a second dataset of coordinates from the gestures (Google). The coordinates are derived from 21 landmarks from the layout of the hand, with each landmark having its own x, y, and z coordinates. With 21 landmarks having at least an x and y coordinate. That leaves us with 21 x 2 = 42 data points. The data points exist for all 26 alphabetic classes 26 x 42 = 1,092. Further, we have 3000 images per class 3000 x 1092 = 3, 276, 000 data points. This calculation does not include the control gestures but gives a relative measure of the minimum amount of data that was expected back from MediaPipe.

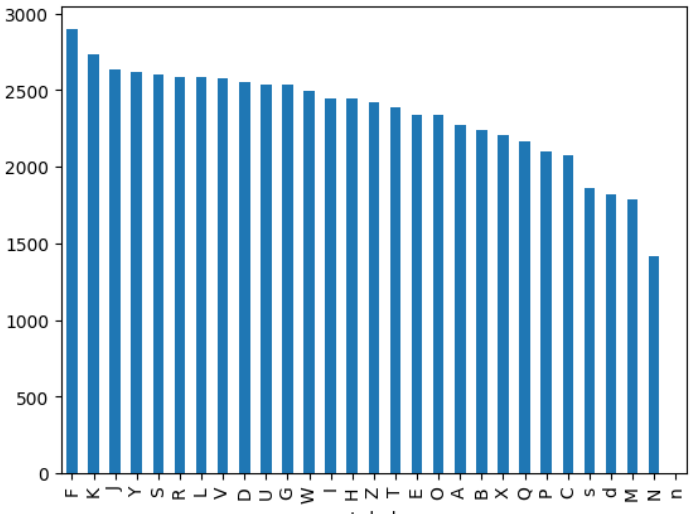

Figure 3

Figure 3 shows a distribution of the number of preprocessed images per class lowercase letters d, s, and n representing the delete, space, and nothing classes, respectively. Notice that although there were 3,000 observations per class, the returned data resulted in a skewed dataset. This prompted further investigation into which images were processed, and which images were not when sent through Google Media Pipe from batches of images that were processed and those that were unprocessed. It was determined that uneven lighting sometimes causes MediaPipe to fail when performing conversions as depicted in Figures 4 and 5.

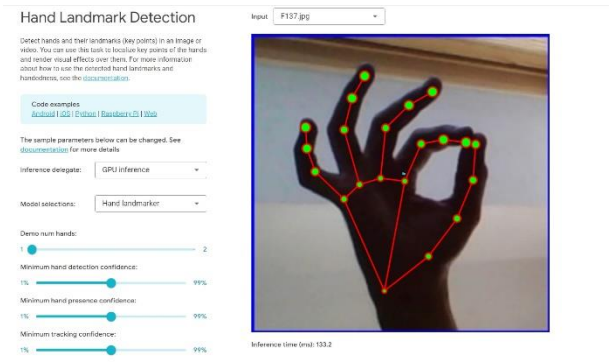

Figure 4

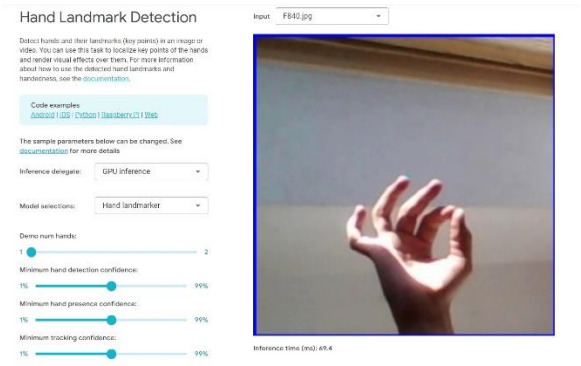

Figure 5

Interesting or Surprising Results

The nothing gesture which was part of the original dataset did not include any hands in the image. However, Google MediaPipe still returned 3 observations where hands were found in the image. Consequently, it indicates that MediaPipe does hallucinate at times. However, given this is 3 out of 3,000 images, its probability of doing so is small.

Although the dataset sourced from Kaggle indicates that it is the ASL alphabet, there are some gestures in the conventional ASL alphabet that include some motion. Two examples include “J” and “Z”. However, given the dataset consists of still images it was impossible to capture motion in the dataset. Therefore, some images do not adhere to the ASL convention for signing the alphabet. that exist in the dataset.

It was assumed that an autoencoder-like model would provide the best results by reducing the representation of the dataset and then expanding it toward the end to identify the class. This was not the case in developing this application. Our experimentation showed the best model implementation was one with a single layer with double the number of neurons as there were inputs.

Summary of Methods used to solve Problems

As stated in the Data Exploration section, there were originally 3 control gestures that we had initially planned to help control the app. The nothing gesture, which had no images of hands, was sometimes miscategorized as having a hand in them and landmarks were returned. To avoid misclassification in the future, the images associated with that class were dropped and replaced with a custom gesture for “clear”.

To enhance interactivity in the application, three additional classes (“speak,” “clear,” and “exit”) were introduced manually. These allow the user to fully control the app without the need to physically interact with it once it has started.

To prevent various x-y coordinates from having too much influence when training the model, the data was normalized forcing all coordinate values to be forced into a scale of 0 to 1 per observation. This was maintained when testing and ultimately when using the app to ensure that coordinates could be properly translated.

In the data exploration section, we identified an issue where not all images were being converted by the MediaPipe environment. Consequently, we ended up with a skewed dataset for training and testing. To avoid the model becoming biased because of unbalanced data, the dataset was resampled to 1,400 observations per class. This resulted in a uniform dataset that was used for training and testing. The number was chosen as this was the smallest quantity of observations found for any of the alphabetic classes, class “N”. Therefore, 1,400 randomly selected observations were selected across all other classes to match.

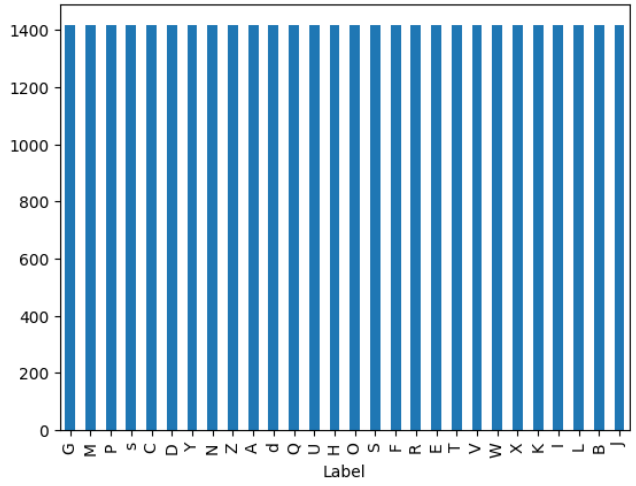

Figure 6

The application proved to be extra sensitive when detecting and classifying gestures from hands in the object. Therefore, we placed a threshold on the application. The application will not accept a gesture unless it has reached a confidence value of 70% and has been held for at least 2 seconds.

Model Selection

Three different neural network configurations were trained and tested for accuracy. The following table shows each model's name, architecture, and reasoning behind its selection.

Model	Architecture	Reasoning
Model1	Input (42), SoftMax (31)	Test the ANN at its most basic i.e. input and output minimums.
Model2	Input (42), Dense (21), Dense (10), SoftMax (31)	Use an autoencoder architecture to reduce the number of neurons to a code of 10 and then inflate the number of neurons in the output.
Model3	Input (42), Dense (84), SoftMax (31)	Increase the number of neurons to get more data representation.

All 3 models were trained for 20 epochs on preprocessed data. We chose model 3 for our final implementation as it provided the best performance average among the different configurations.

Model	Loss	Accuracy	F1
Model 1	24.02%	95.60%	95.70%
Model 2	14.03%	96.79%	96.70%
Model 3	6.51%	98.67%	98.45%

Final Implementation

To fulfill our goal of creating a full-fledged application that someone who is speech-impaired can use, we integrated the model with a user interface using Python. A sample of the final implementation is shown in Figure 7

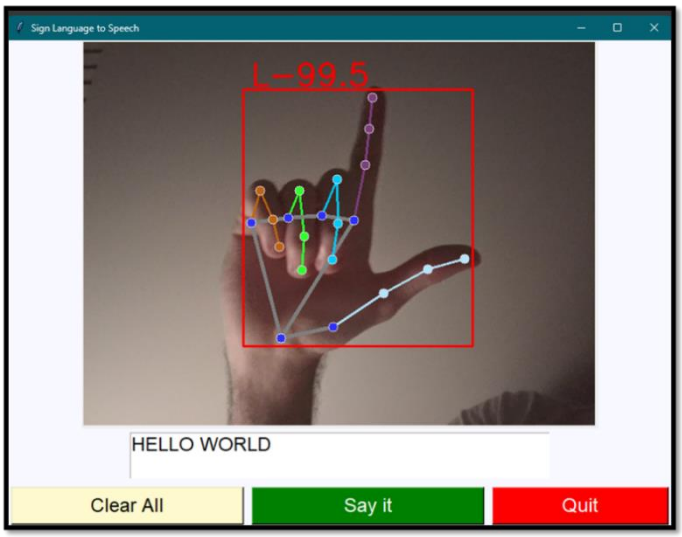

Figure 7

The application interface features a video screen where the model identifies the hand. Having done so, it overlays a red box on the hand and the landmarks on it. It then classifies the gesture with a confidence score. For example, in the image provided, the model classifies the gesture as the letter “L” with a 99.5% confidence.

Below the video feed is a text box where the predicted letters are printed out to form words. For a letter to appear in the text box, the model must classify it with a confidence score higher than 70% and the gesture must be held for at least 2 seconds. These parameters can be changed in the code.

Finally, the interface includes convenience buttons to clear the message, quit the application, and make the app "say" the message. The "Say it" option sends the message to the Google Text-to-Speech (GTTS) API, which returns an MP3 recording of the message in English and plays it aloud. These three actions (“clear”, “quit”, and "say it") can also be triggered using specific gestures.

Limitations and Future Research

Although the project did achieve its intended goals, we did see some limitations and room for improvement. One limitation of our project is the lack of data augmentation techniques. This was not done due to time constraints. As a result, the model is restricted to recognizing signs performed only with the right hand, which must remain straight and free of rotation. Future updates could address these challenges by incorporating data augmentation techniques.

Additionally, it is well known that American Sign Language (ASL) also has motion-based signs, signs for complete words and/or sentences. Due to the chosen architecture of using a Feedforward Neural Network instead of an RNN, we are unable to capture motion-based signs. Future research may use different architectural approaches, such as Convolutional Recurrent Neural Networks (CRNN), to effectively process and interpret dynamic gestures.

References

Amazon. "What is Tap to Alexa?" n.d. *Amazon.* <https://www.amazon.com/gp/help/customer/display.html?nodeId=G69UDERKYBL55DW9>.

Apple. "Accessibility features for speech on iPhone." n.d. *Apple Support.* <https://support.apple.com/guide/iphone/overview-of-accessibility-features-for-speech-iph8b6c223ac/18.0/ios/18.0>.

Google. "Accessibility in Our Products & Features." 23 12 2021. *Google.* <https://about.google/belonging/disability-inclusion/product-accessibility/?q=speech#module-modal-embed-accessibility-android-pixel__link-anchor>.

—. "Cloud Text-to-Speech." 3 November 2024. *Google Cloud.* 09 12 2024. <https://cloud.google.com/text-to-speech?hl=en>.

—. "Hand landmarks detection guide." 21 May 2024. *Google AI for Developers.* 09 12 2024. <https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker>.

Nagaraj, Akash. *ASL Alphabet [Data set]*. 2018. <Akash Nagaraj. (2018). ASL Alphabet [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/29550>.

OpenCV. *OpenCV*. 3 November 2024. <https://opencv.org/>.