LinguaVox

Group Name: **Insight Inspectors**

Group Members:

First name	Last Name	Student number
Avinash	**Ravi**	**C0791941**
Ankur	**Kishorbhai Rokad**	**C0793757**
Sahista	**Patel**	**C0796681**
Ummer	**Shariff**	**C0791871**
Vishal	**Sabarinath Venkatesan**	**C0801202**

Submission date: **December 18th, 2021**

Abstract

As we all know, vocally and hearing-impaired people face difficulties in their daily lives to communicate effectively. Because most individuals do not learn sign language and interpreters are few, we developed an application for translating sign language to text and vice versa. The sign language is detected in real-time for fingerspelling-based American sign language using neural networks and translating text to sign language. If the user selects the sign language to text option, the hand is initially passed through a filter. Our method, following which, is passed through a classifier, which predicts the class of the hand motions to show the corresponding letter. When the user selects text to sign language, the text is processed and translated to related sign language videos. For the 26 letters of the alphabet, our method is 95.7 percent accurate.

Keywords: American Sign Language, Neural Networks, Artificial Intelligence

Table of Contents

Table of Figures

Introduction

The number of hearing and speaking impaired people worldwide is staggeringly high and roughly calculated to be millions. Of these 63 percent, they are said to be born deaf. The others lose their hearing by different accidents. Over one percent of Canada's population or approximately half a million people are physically challenged. In Ontario, an estimated 211,250 such individuals.

Since more than 90 percent of deaf children are born to hearing parents, many adults need to learn sign language, but it is practically impossible. There are few reliable statistics on which signed languages are most spoken or widespread. Still, the top 3 candidates are American Sign Language (ASL), British Sign Language (BSL), or Australian Sign Language (Auslan).

ASL is a good contender for the title; hence we will be using it for our project as it is used in the U.S. And Canada (with some regional differences). We created a program that converts sign language to text and vice versa. The sign language for fingerspelling-based American sign language is detected in real-time using neural networks. When the user chooses the sign language to text option, the user's hand is first filtered.

It is then sent via a classifier in our technique, which predicts the class of the hand motions to reveal the matching letter. We intend to develop a model that can recognize Fingerspelling-based hand motions and combine them to make a whole word. The gestures that we want to improve are depicted in the image below Figure 1.

When The text to sign language option is selected, the text is processed and translated into an appropriate sign language video. On the next page, we have a figure that shows the symbolic representation of American Sign Language.

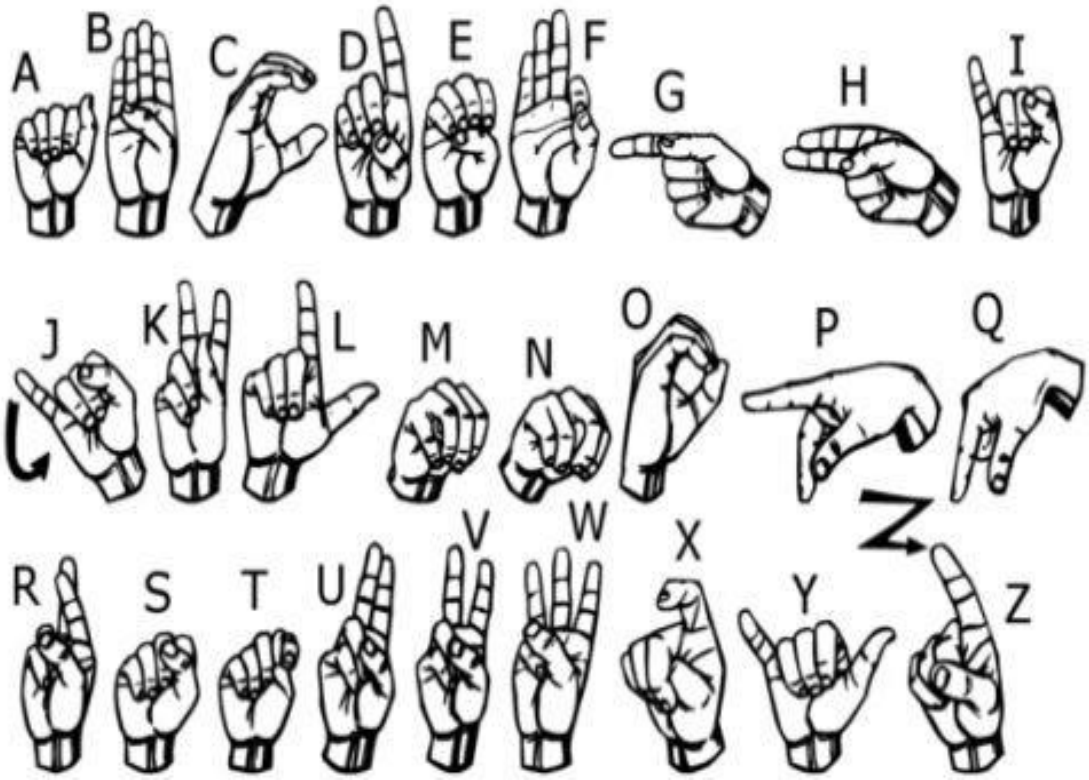


Figure 1: ASL Letters

Methods

We wanted to implement a product that would be useful to everyone who wants to communicate with specially-abled people. We wanted to create an application that will integrate Sign Language to Text and Text to Sign Language Detection. So, we have divided our project into two different modules –

- Sign Language to Text
- Text to Sign Language

These two modules are an integral part of our project, and they are available to users on the landing page of our website itself. We have also represented our LinguaVox architecture below.

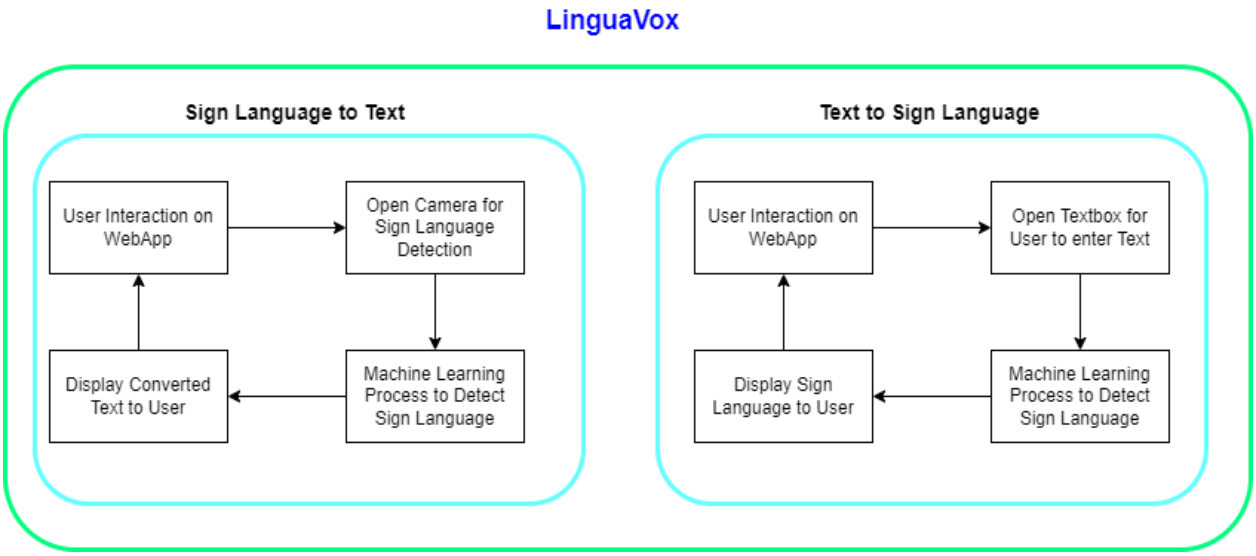


Figure 2: LinguaVox Architecture

Sign to Text

In the first module, we wanted to detect sign language and then capture the sign language portrayed by the user and convert it to text which other users can understand.

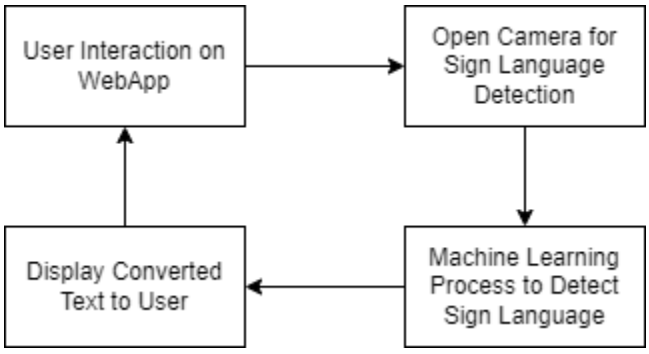


Figure 3: Sign Language To Text Workflow

To achieve this, the flow we have followed is we identified characters, words, and sentences from video input; on top of that, we also included some word suggestions for the signs.

Data Preparation

Selecting multiple data sources, including building our dataset. We finalized the dataset of consist of 156 images of each letter. There are 26 letters + 1 blank frame + 10 digits [0-9]; we finalized 37 folders of 156 images each for testing data and 468 images each for the training dataset for our model in this project.

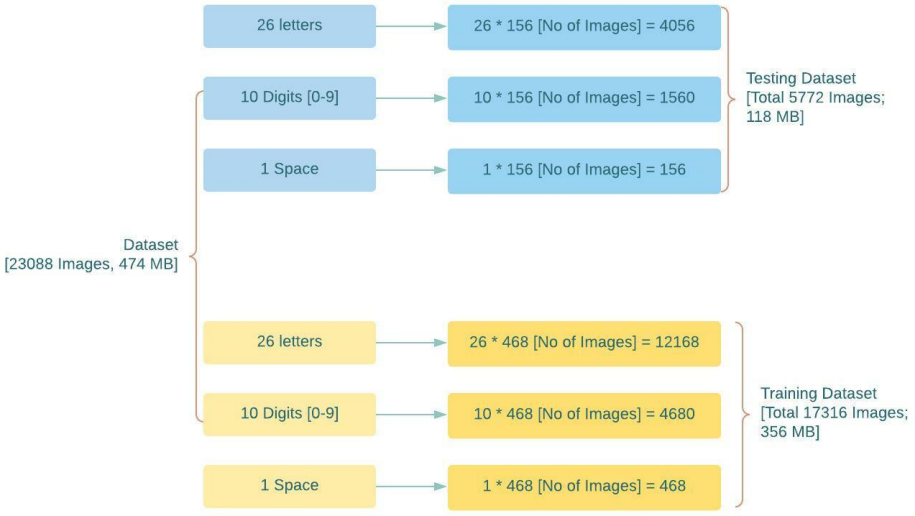


Figure 4: LinguaVox Dataset

After going through all the possible data sources and listing the pros and cons of using those, we chose to go with our own dataset creation and use it. We have implemented a program that opens the machine's camera port and runs a loop by taking frames and storing each. We also set a few labels in this process, looped through the tags, and stored some photos using the webcam.

Figure 5: LinguaVox Data source

While storing the images, we had to identify the sign by clipping out the sign only by removing the remaining frame of the picture. Then after cleaning the images for better performance and specifically for avoiding skewness of data, we used Gaussian Blur Filter and extracted details. Gaussian Blur Filters are mainly used to reduce noise and remove detailed information from the image. In addition to it, we have implemented a [RGB to] black and white filter that will help the

camera identify the handprints and structure quickly. The camera can easily detect hand gestures or any other movement with this. After applying the filter, the image looks like this -

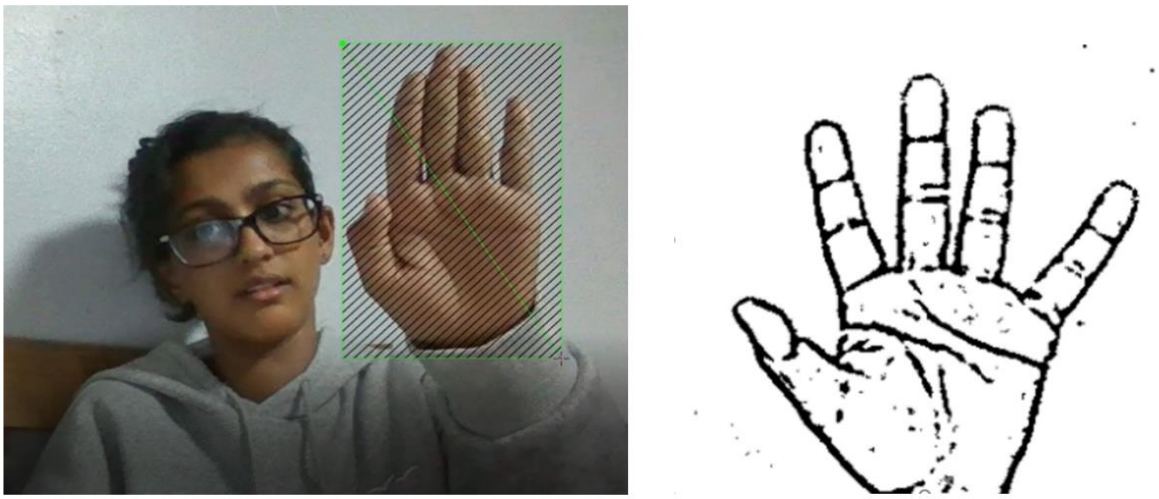


Figure 6: Image Capture with Gaussian Filter

Before data modelling labelling is essential, we labelled each image manually using the labelimg tool. Few of the word labelling is shown in the below image.

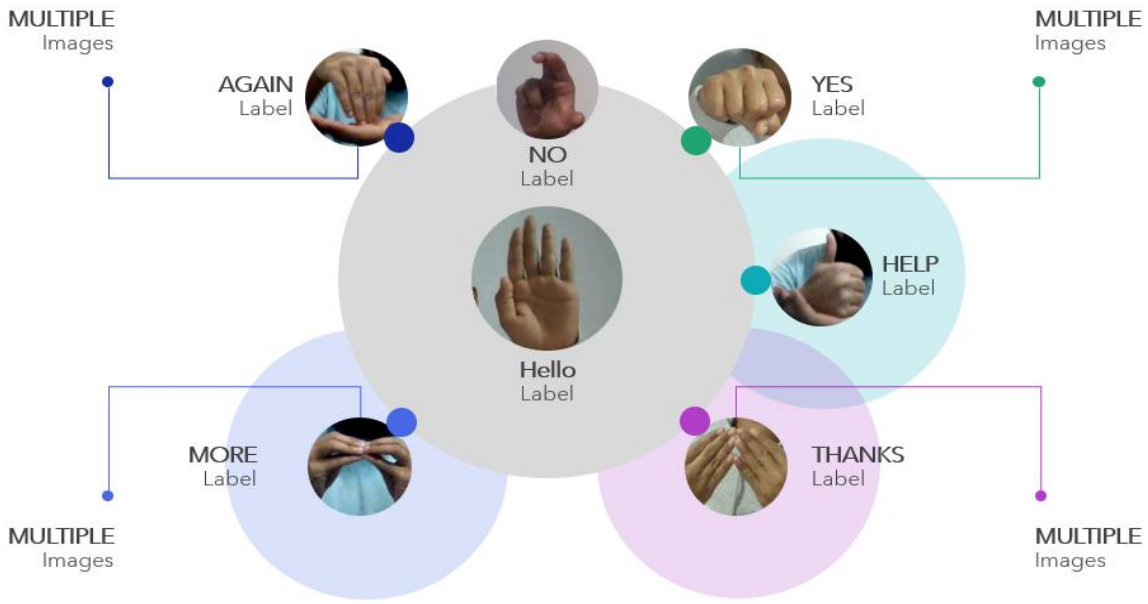


Figure 7: Image Labelling

Data Modelling

Data Modeling is an essential part of building and Machine Learning models. First, we must identify which model would suit our problem and choose accordingly. After collecting, cleaning images, and thoroughly researching the results of multiple model applications, we decided to go with Convolutional Neural Network (CNN) to make predictions from the input. We chose CNN as our go-to model not only on that basis but because CNN is that it arranges the neurons in 3 dimensions: width, height, depth. So the neurons will be connected to a relatively more minor portion of the last neuron instead of fully connected like a regular neural network.

Next is which layering is best suitable for our dataset; this is a trial and error thing to come up with. After many iterations, the final layer structure we come with is below,

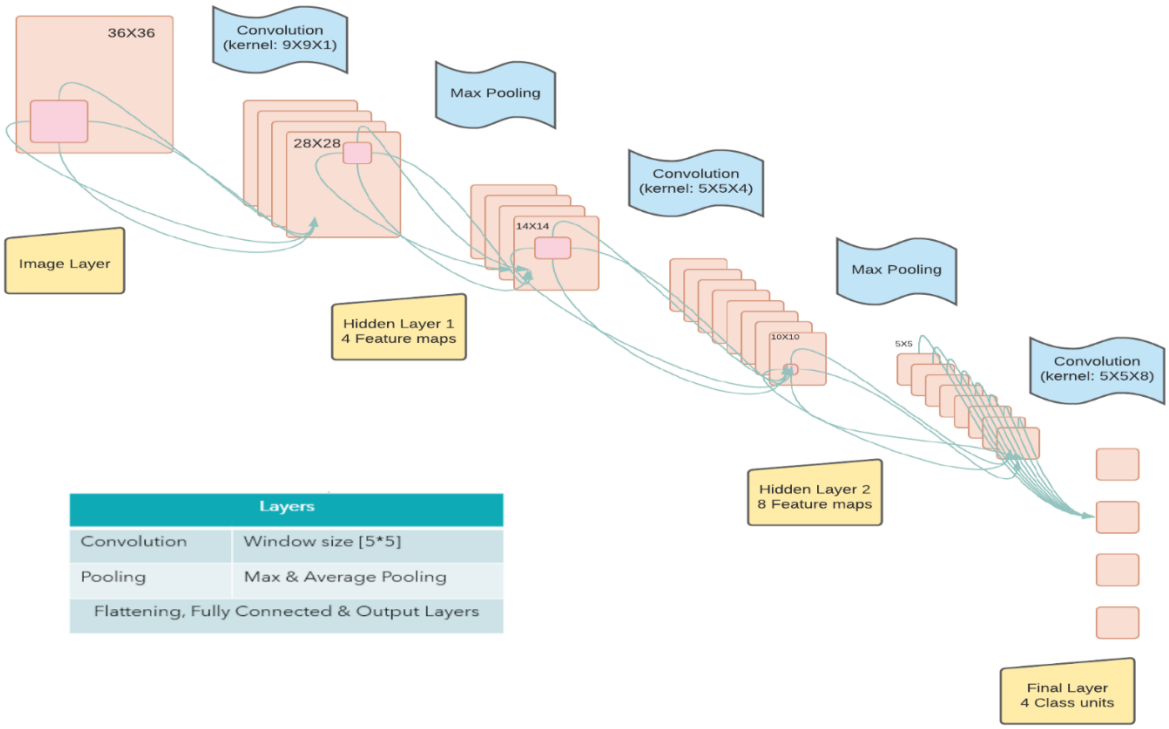


Figure 8: Modelling - Layers

Layers

Convolutional Layer**:**

Here we used a 5*5 window as input that extends to the depth of the input matrix. The learnable filters in the layer are 5*5. We slid the window by one stride and computed the dot product of filter entries and input values at a given position during every iteration. Then we created a 2-Dimensional activation matrix that shows the response of that matrix at every spatial position. The network will train filters that activate when they see features such as an edge of some orientation.

Pooling Layer:

Here we reduced the size of the activation matrix and eventually reduced the learning parameters. In the **Max Pooling,** we take a 2*2 window to get four values from the input, and in **Average Pooling,** we took an average of the window; after pooling, we reduced the size of the activation matrix to half.

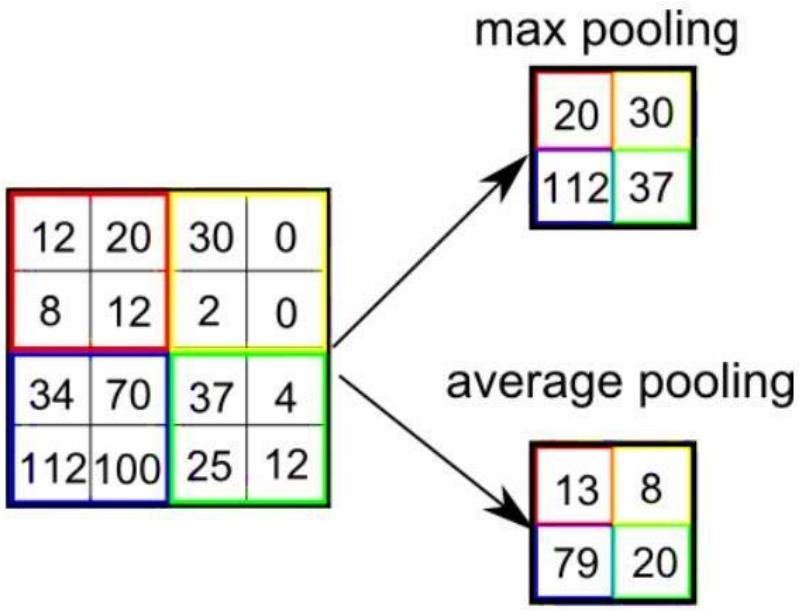


Figure 9: Pooling Layers

Fully Connected Layer:

In the convolution, layer neurons are connected only to a local region, while in a fully connected part, well connect all the inputs to neurons.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4194432
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 96)                12384
_________________________________________________________________
dropout_1 (Dropout)          (None, 96)                0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                6208
_________________________________________________________________
dense_4 (Dense)              (None, 27)                1755
=================================================================
Total params: 4,240,859
Trainable params: 4,240,859
Non-trainable params: 0
```

Figure 10: Model Summary

Training

For model input, we used grayscaled and applied the gaussian blur to remove unnecessary noise. After that, we reduced the image size to 128 x 128 by applying an adaptive threshold to extract hands from the background. We have used the SoftMax function to predict how likely an image will fall under one of the labels. So the normalized output lies between 0 and 1, and the sum of each value in each label is one.

Testing

We found some incorrect predictions during the testing, so we used two layers of algorithms to verify and predict more similar symbols.

Incorrect predictions:

1. For D: R and U
2. For U: D and R
3. For I: T, D, K and I
4. For S: M and N

So, to handle this issue, we made three different classifiers for classifying these sets:

1. {D, R, U}
2. {T, K, D, I}
3. {S, M, N}

Evaluation

After testing the model, we achieved 95.8% accuracy with one layer and 98.0% with two layers. The images below show the confusion matrix comparison of both algorithms, with one and two layers.

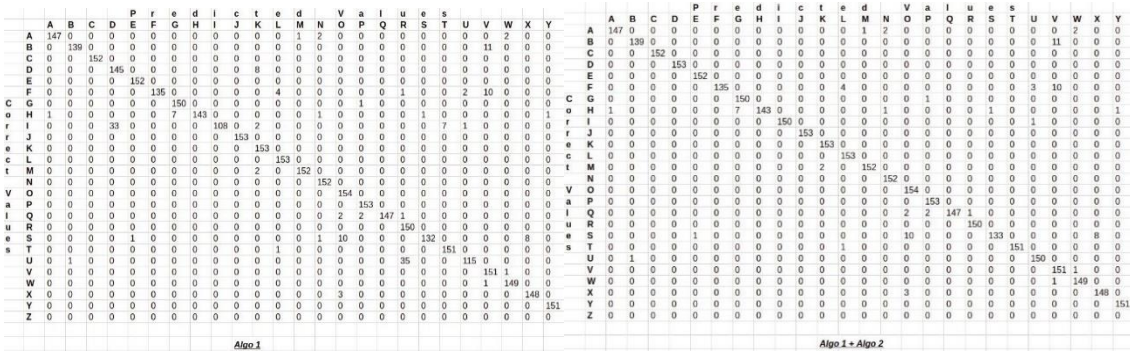


Figure 11: Model Evaluation Results

Text to Sign

We have discussed the translation of sign language to text, and in our product, we facilitate the translation vice versa, i.e., text to sign language. We will be translating the text in English to ASL and displaying the text from the user as a sign language video. The below flowchart describes the flow of text to sign language translation.

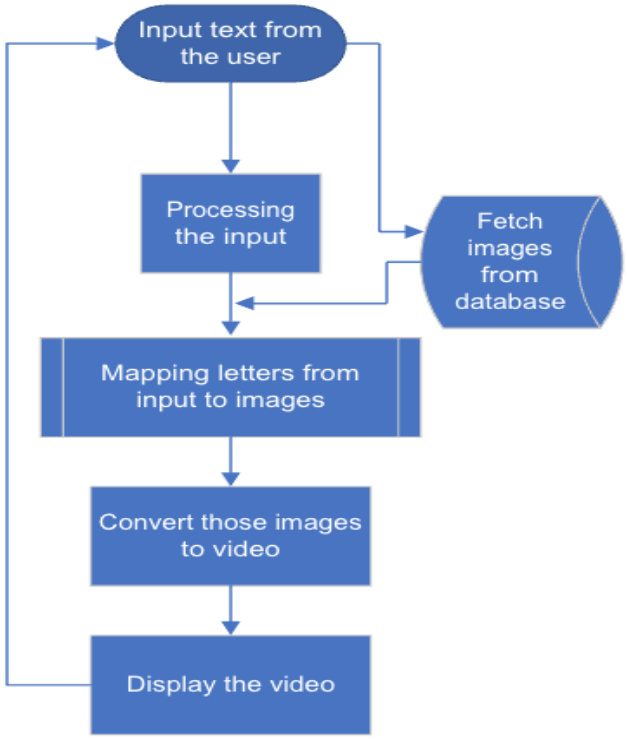


Figure 12: Text to Sign Language Workflow

Dataset Information

There are a lot of images available over the internet for individual ASL letters and Numbers. We have collected all images representing the alphabet and numbers and will be stitching these images to form a video.

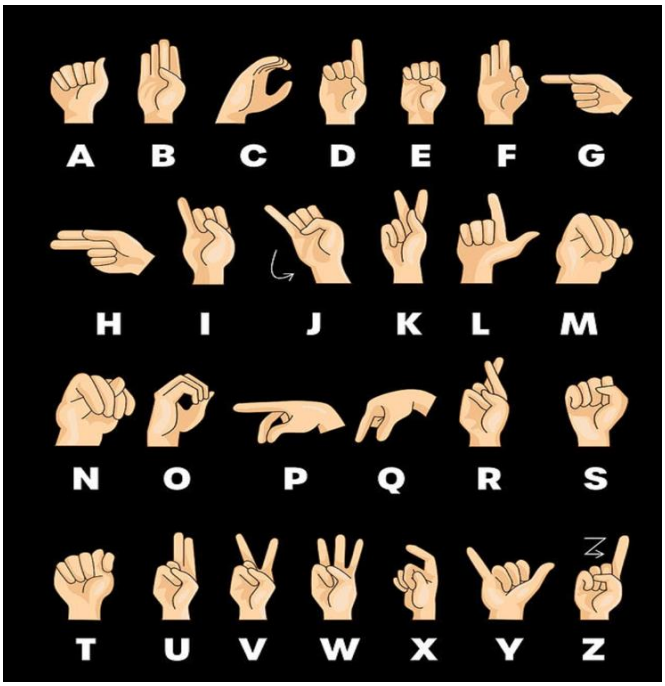


Figure 13: ASL Sign Language Alphabets

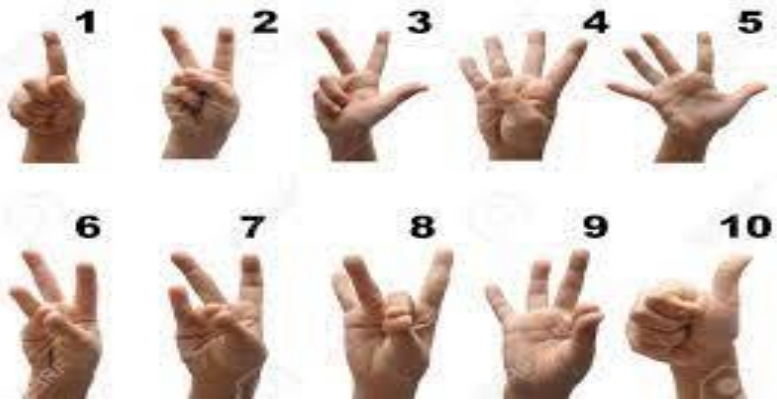


Figure 14: ASL Sign Language Numbers

Data Preparation and Pre-processing

When the text to sign language service is selected, the user is prompted to enter text translated into sign language. The entered text might have special characters and emojis that must

be removed to move forward with the translation. We will only be translating alphabets and numbers to the corresponding sign language. The text should also be converted to string format since the input we take is in a list and then make the characters a small case for easier processing.

Figure 15: Data Pre-Processing Steps

Sign language video generation

The clean text now contains the characters we need to translate. For that, we will map individual characters with the corresponding image and then stitch those images to generate a video using the pillow library in python. For example, the characters are mapped as shown below:

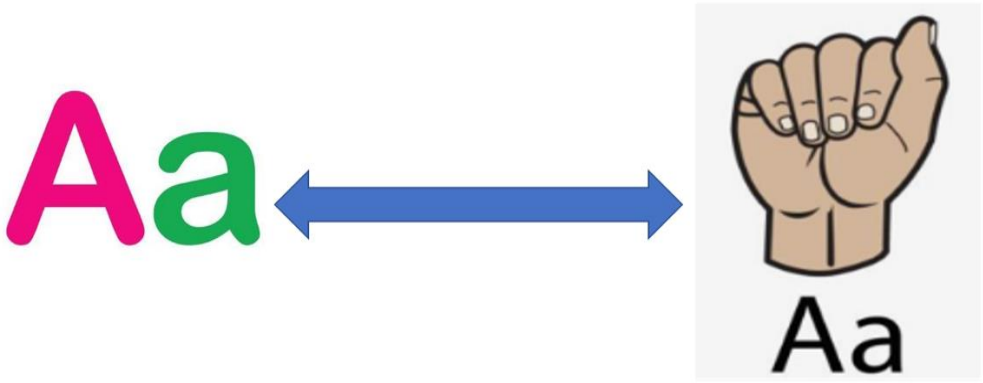


Figure 16: Character Mapping to ASL Image

Keywords & Libraries

The Python Imaging Library PIL "Pillow"

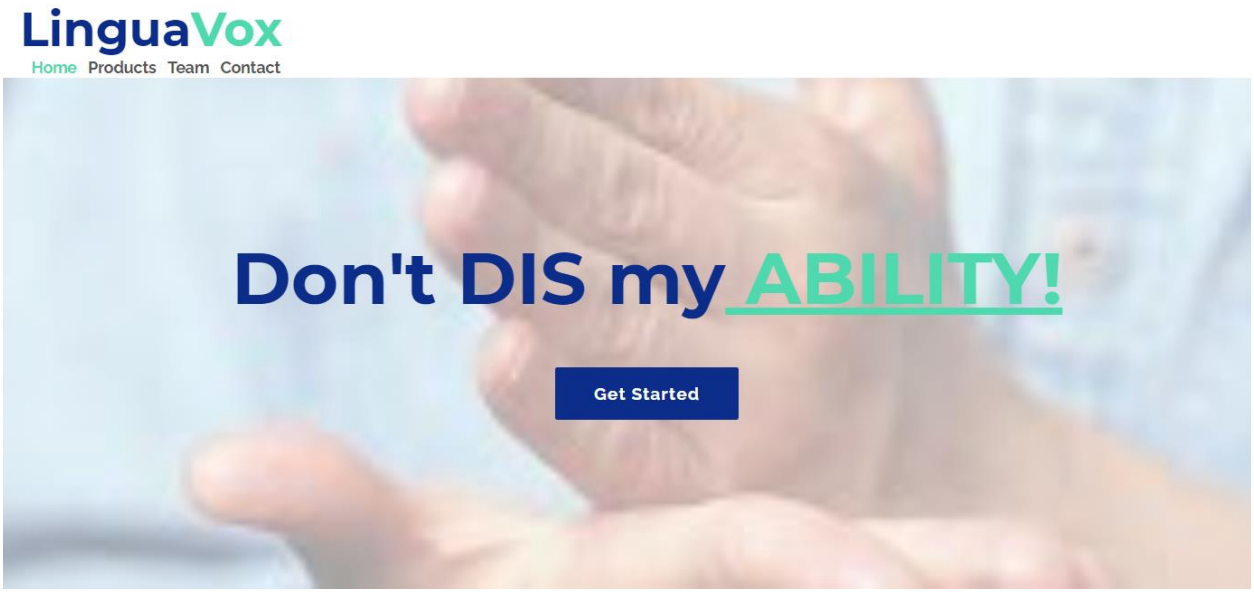
This library extends our Python interpreter's image processing capabilities. It supports various file formats, has a fast internal representation, and can-do complex image processing. The image library's core provides quick access to data saved in a few basic pixel formats. It should serve as a good foundation for an image processing tool, in general.

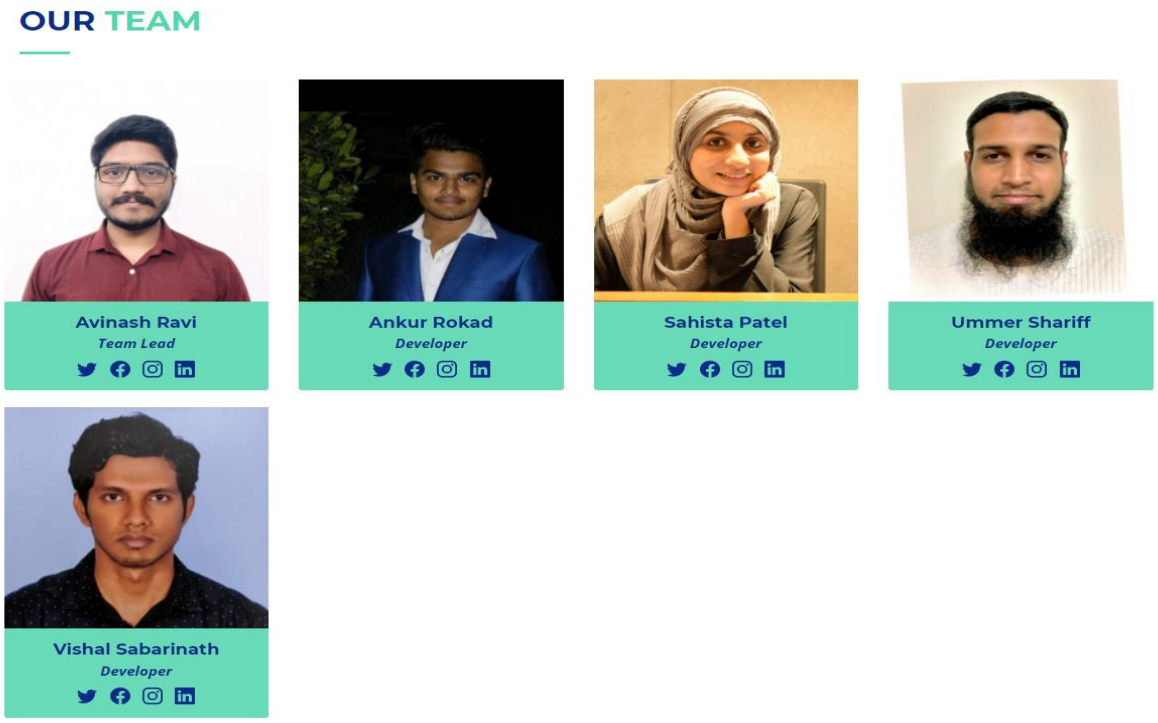
Imageio

A Python library lets you read and write various image data types, including animated images, video, volumetric data, and scientific formats.

Results

We developed a user-friendly UI available on the web and standalone desktop applications as a final deliverable. Users can download the two applications one time and use them without having internet access. The below screenshots illustrate our web application and functional overview of it.





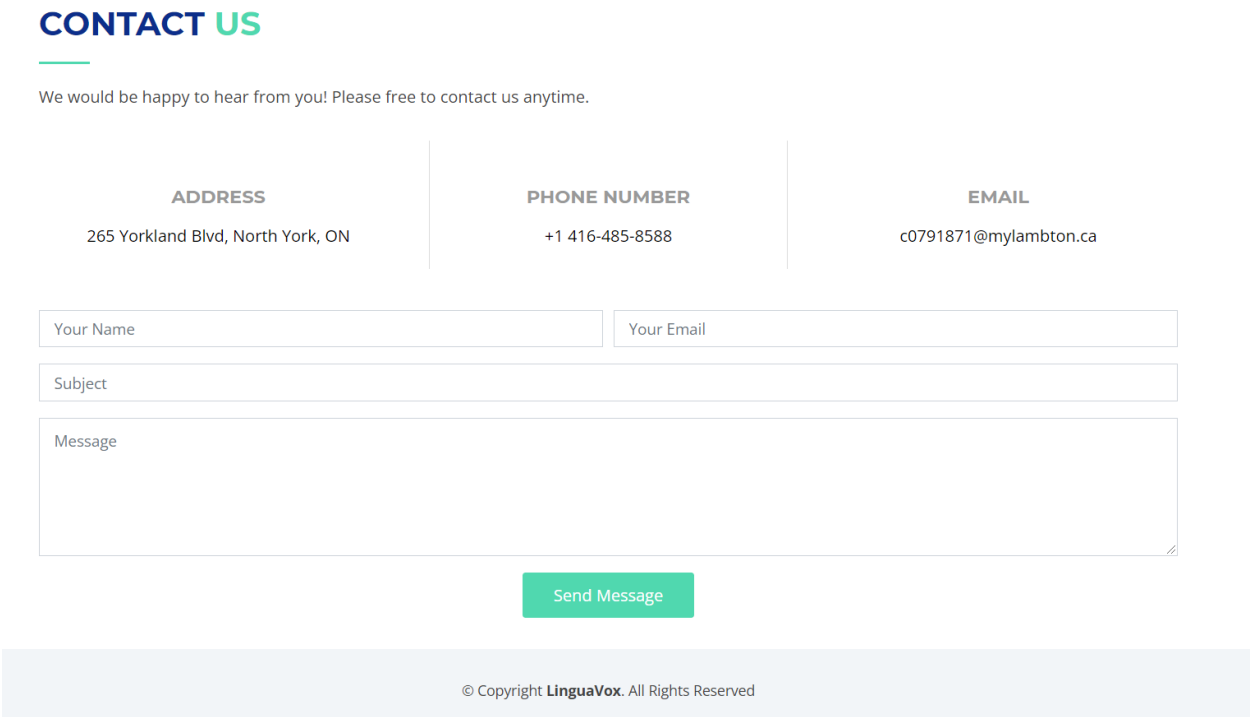


Figure 17: LinguaVox Web-View

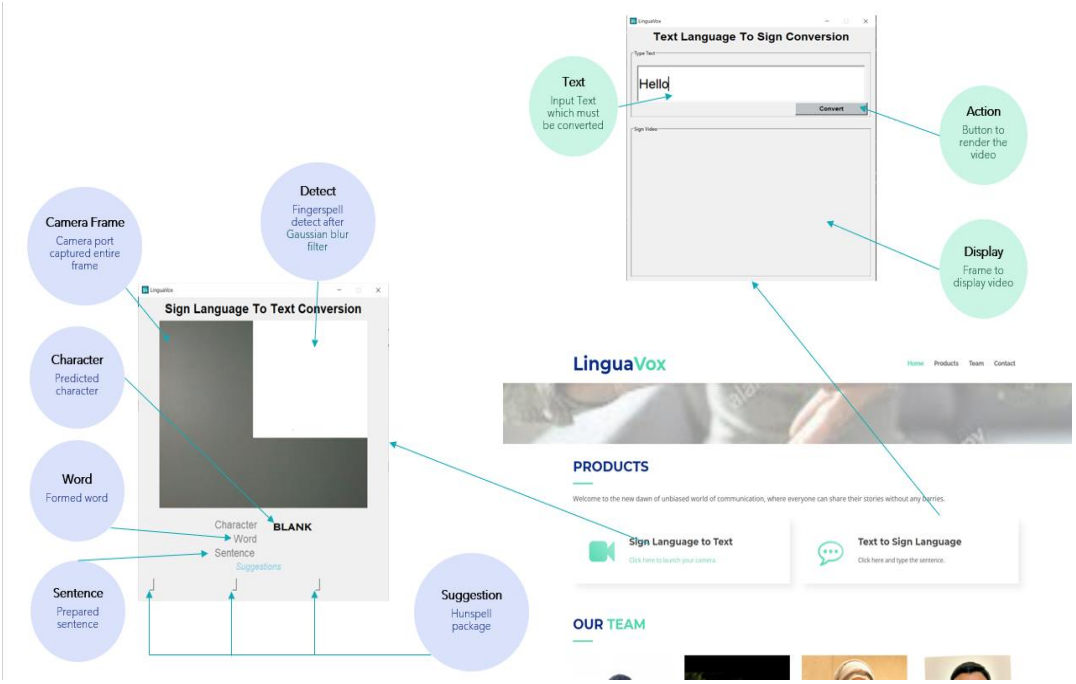


Figure 18: LinguaVox Functionality Description

The above image shows how the website looks like. When a user lands on the website, one has two options: **Text to Sign** and **Sign to Text**. Upon clicking on either a respective pop-up will open and take input as text or sign and show the predicted result.

Once a user clicks on the Sign Language to Text, we get a pop-up that displays user images to portray the ASL and get their desired results.

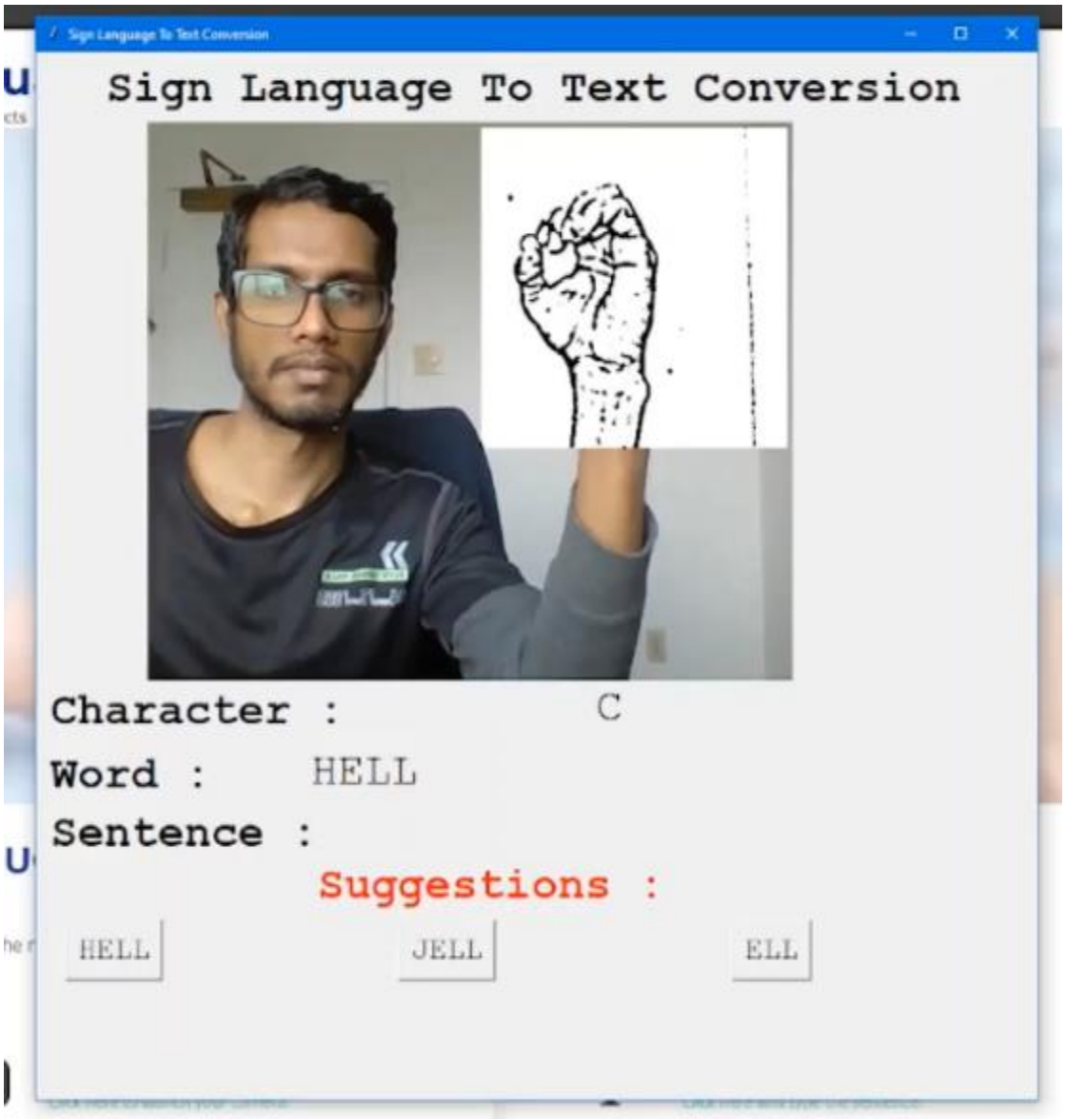


Figure 19: Sign Language to Text

If the user wants to use our Text to Sign Language module, once they click on the hyperlink given on the website, the below pop-up opens up. The user can type down their desired text and receive their result played in a video format.

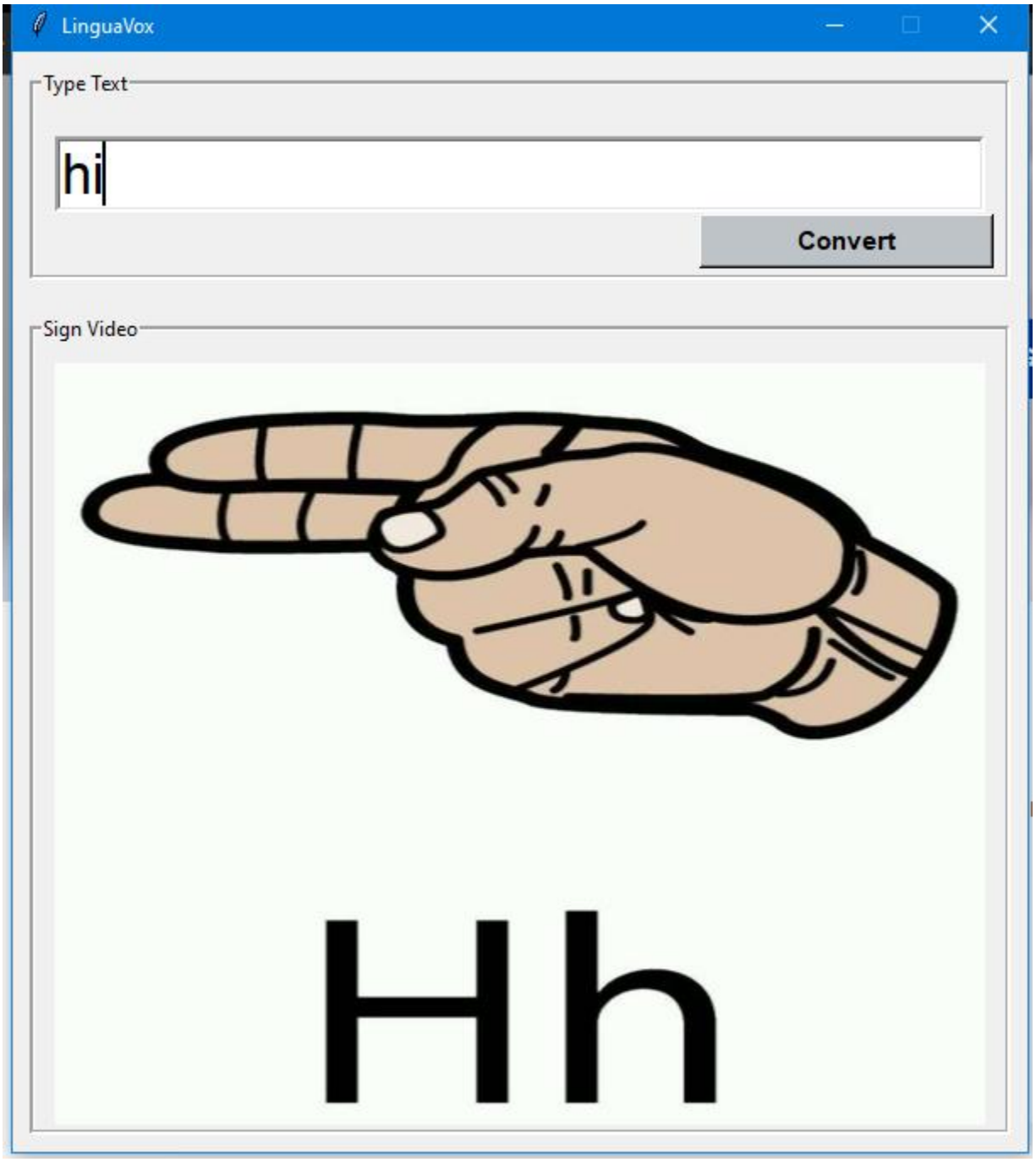


Figure 20: Text to Sign Language

Conclusions and Future Work

We hope to improve accuracy even in complicated backgrounds by experimenting with various background removal methods. We're also considering enhancing the Pre-Processing to better predict gestures in low-light situations. The current project only works with ASL; given the correct data and training, it might be modified to function with other native sign languages. Although this project uses a fingerspelling translator, sign languages are also spoken in context, with each gesture representing an item or a sentence. As a result, detecting this type of contextual signing would necessitate more processing and natural language processing (NLP).

Improvements can be made to this project by creating a mobile application that everyone can download and use on their mobile phones. We can also improve the model performance and train it on other sign languages such as British Sign Language, Chinese Sign Language, etc. Our subject project's primary goal is to ensure that no one should have a communication barrier. We want to eliminate that, and by using our product, the specially-abled people can freely express their views and opinions to others.

References

Bhatia, R. (2018). *Neural Networks Do Not Work Like Human Brains – Let's Debunk The Myth.* Analytics India. https://analyticsindiamag.com/neural-networks-not-work-like-human-brains-lets-debunk-myth/

Byeongkeun K., Subarna T., Truong Q. (2015). *Real-Time Sign Language Fingerspelling Recognition Using Convolutional Neural Networks From Depth Map* – 3rd IAPR Asian Conference on Pattern Recognition (ACPR).

N. Mukai, N. Harada and Y. Chang. (2017). *Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning*. Kyoto, Japan, 2017, pp. 19-24.

Pierson, L. (2019). *Python for Data Science Essential Training Part 1* [MOOC]. LinkedIn Learning. https://www.linkedin.com/learning/python-for-data-science-essential-training-part-1?u=56968457

Pierson, L. (2019). *Python for Data Science Essential Training Part 2* [MOOC]. LinkedIn Learning. https://www.linkedin.com/learning/python-for-data-science-essential-training-part-2?u=56968457

Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015). *Sign Language Recognition Using Convolutional Neural Networks*. Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925.