December 18, 2021

Professor Vahid Hadavi

Lambton College

400-265 Yorkland Blvd

Toronto, ON M2J 1S5

Dear Professor Hadavi,

Re: LinguaVox – Project Report

We take this opportunity to present our project report on our product **LinguaVox**. LinguaVox is one of a kind product that we intend to serve the challenged community. Our product will allow them to break free of the communication barrier. This report will give a detailed analysis of the project – datasets, data cleaning, data modelling, user interface, middleware API and the results. We used different neural network models to predict and interpret the sign language and display the sign language that the user wants to express.

We have used the necessary methods under each step of the machine learning model to maximize performance. We ensured that our product effectively predicts using various resources, have a seamless user experience, and we thank you for this opportunity.

Sincerely,

Insight Inspectors
Ankur Kishorbhai Rokad (C0793757)
Avinash Ravi (C0791941)
Sahista Patel (C0796681)
Ummer Shariff (C0791871)
Vishal Sabarinath Venkatesan (C0801202)

LinguaVox

Group Name: **Insight Inspectors**

Project Link: GitHub / OneDrive / LinguaVox WebApp

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**

Instructor: **Prof. Vahid Hadavi**

Abstract

As we all know, physically challenged people especially the hearing and speaking have difficulty communicating effectively in their daily lives. We have created a web-based application to translate sign language to text and vice versa because most people do not learn sign language, and interpreters are few. Using neural networks, the sign language for fingerspelling-based American sign language is detected in real-time, and text is translated to sign language. When the user chooses the sign language to text option, the user's hand is first filtered. Following that, our approach is sent via a classifier, which predicts the class of hand motions that will be used to show the appropriate letter. When the user selects text to sign language, the text is processed and translated into appropriate sign language video. Our method is 95.7 percent accurate for the alphabet's 26 letters.

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), Australian Sign Language (Auslan) or British Sign Language (BSL).

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 to 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.

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.

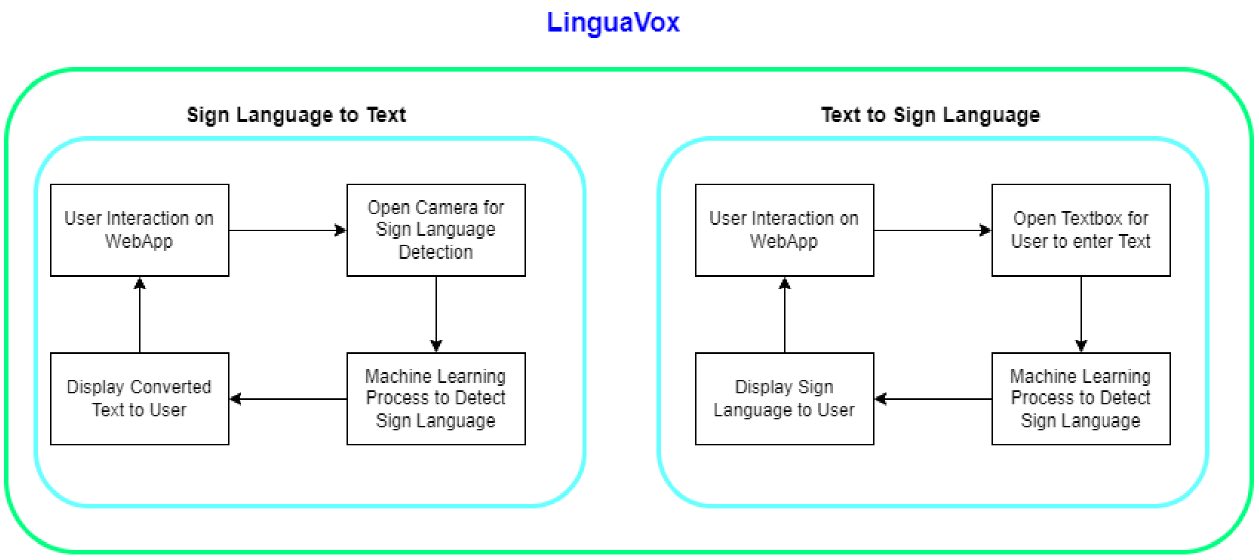

LinguaVox
Sign Language to Text
User Interaction on WebApp
Open Camera for Sign Language Detection
Machine Learning Process to Detect Sign Language
Display Converted Text to User
Text to Sign Language
User Interaction on WebApp
Open Textbox for User to enter Text
Machine Learning Process to Detect Sign Language
Display Sign Language to User

Figure 1: 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.

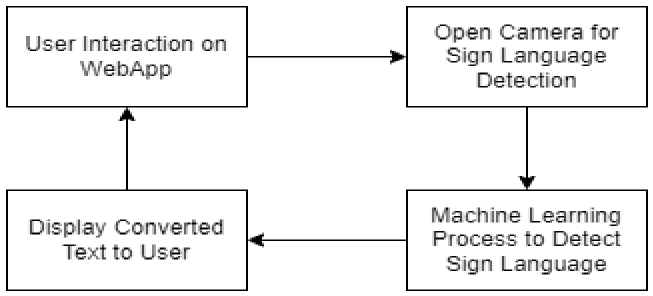

User Interaction on WebApp
Open Camera for Sign Language Detection
Machine Learning Process to Detect Sign Language
Display Converted Text to User

Figure 2: 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.

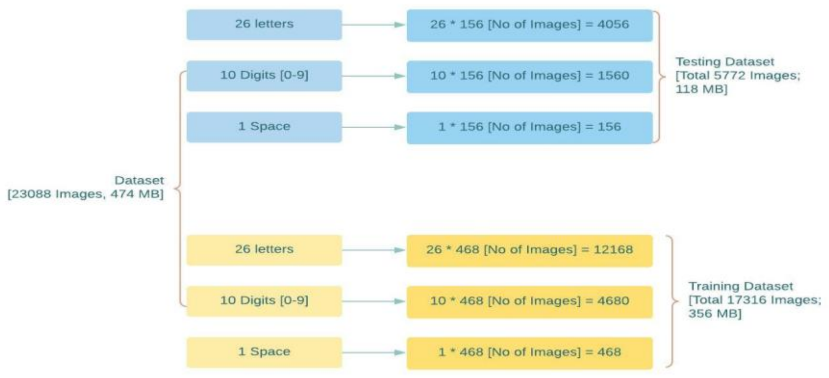

Dataset [23088 Images, 474 MB]
26 letters
26 * 156 [No of Images] = 4056
10 Digits [0-9]
10 * 156 [No of Images] = 1560
1 Space
1 * 156 [No of Images] = 156
Testing Dataset [Total 5772 Images; 118 MB]
26 letters
26 * 468 [No of Images] = 12168
10 Digits [0-9]
10 * 468 [No of Images] = 4680
1 Space
1 * 468 [No of Images] = 468
Training Dataset [Total 17316 Images; 356 MB]

Figure 3: LinguaVox Dataset

After going through all the possible data sources and listing the pros and cons of using them, we chose to create our own dataset 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.

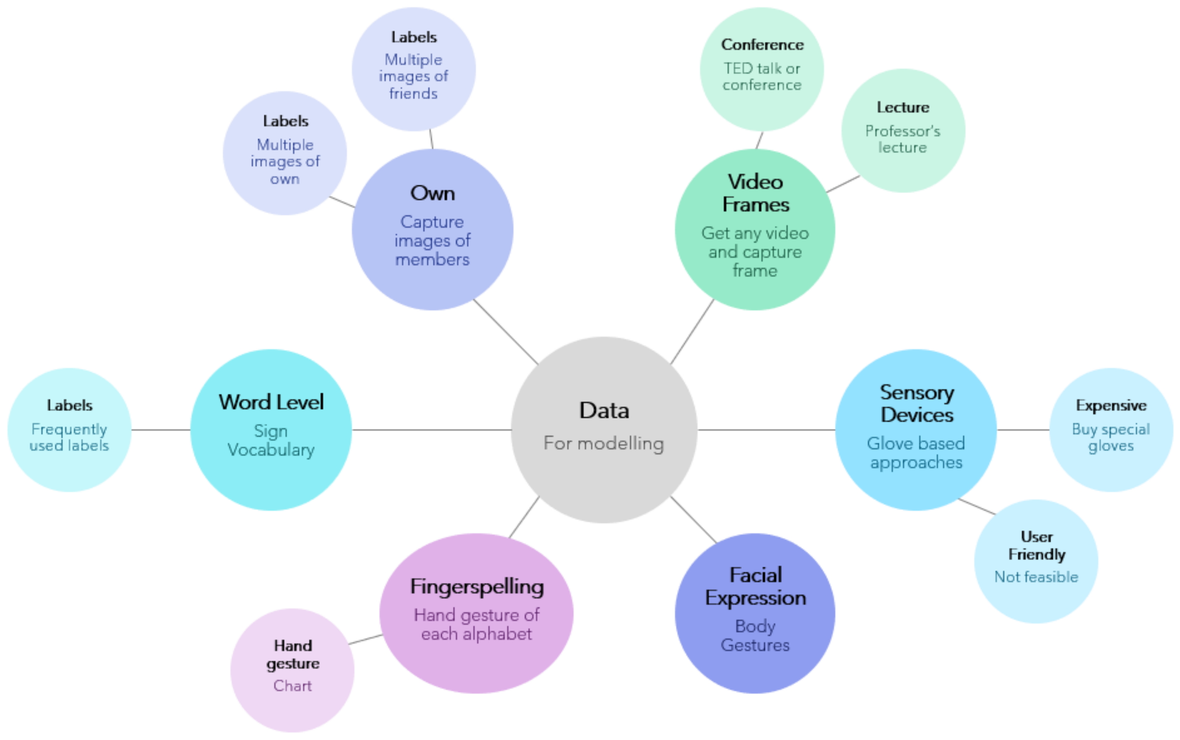


Figure 4: LinguaVox Data source

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 -

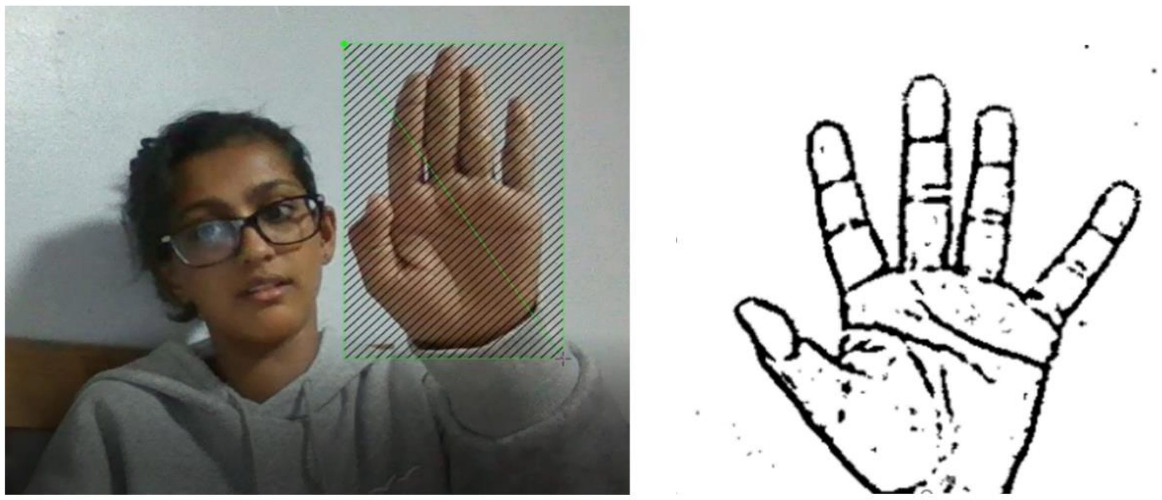


Figure 5: Image Capture with Gaussian Filter

Before data modelling labelling is essential, we labelled each image manually using the labelimg tool. First, we have to filter out the best images from all the images collected by our team members and the images captured from various other sources such as TED talks and class lectures. Once we had the processed images, we had to label each one of them manually. In American Sign Language, we have signs different for words, letters and sentences; So, we had to make sure that we capture images for all 26 alphabets and also a few words such as ‘Hi,’ ‘Hello,’ ‘Thank you, ‘Sorry,’ etc. Few of the word labelling is shown in the below image.

Once we label the images with the help of the labelimg tool, the tool generates XML files and saves them to the desired location. This XML file has features that are important to our machine learning model. These features include the images’ width, height, depth, etc.

On the next page, we have shown a few shots that we have labelled and further moved towards the layering and how we have utilized or fed them to our model.

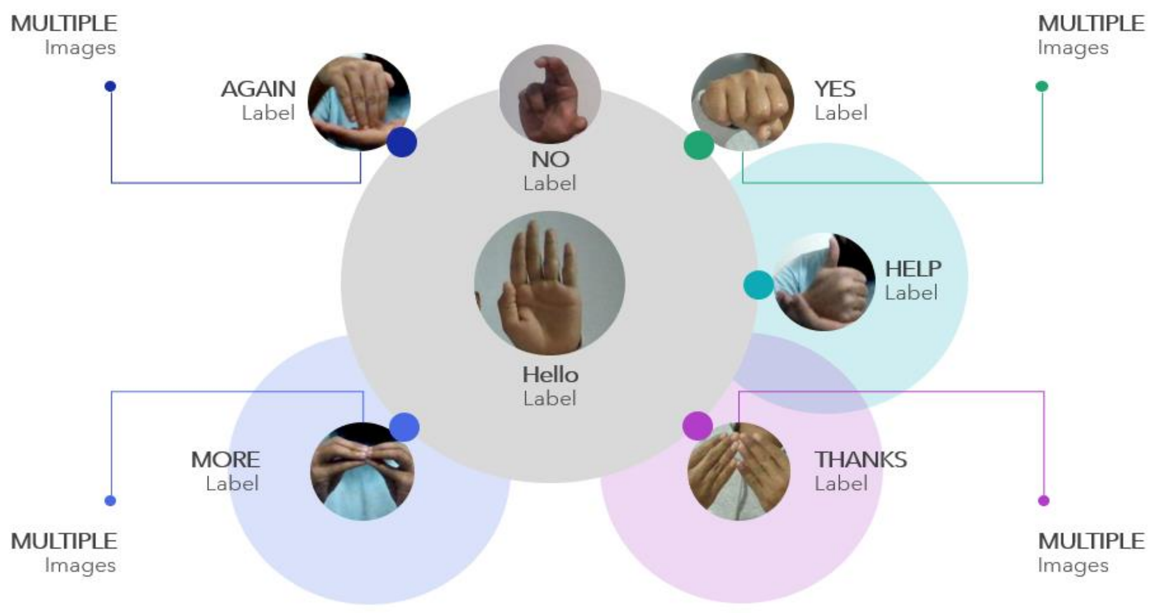
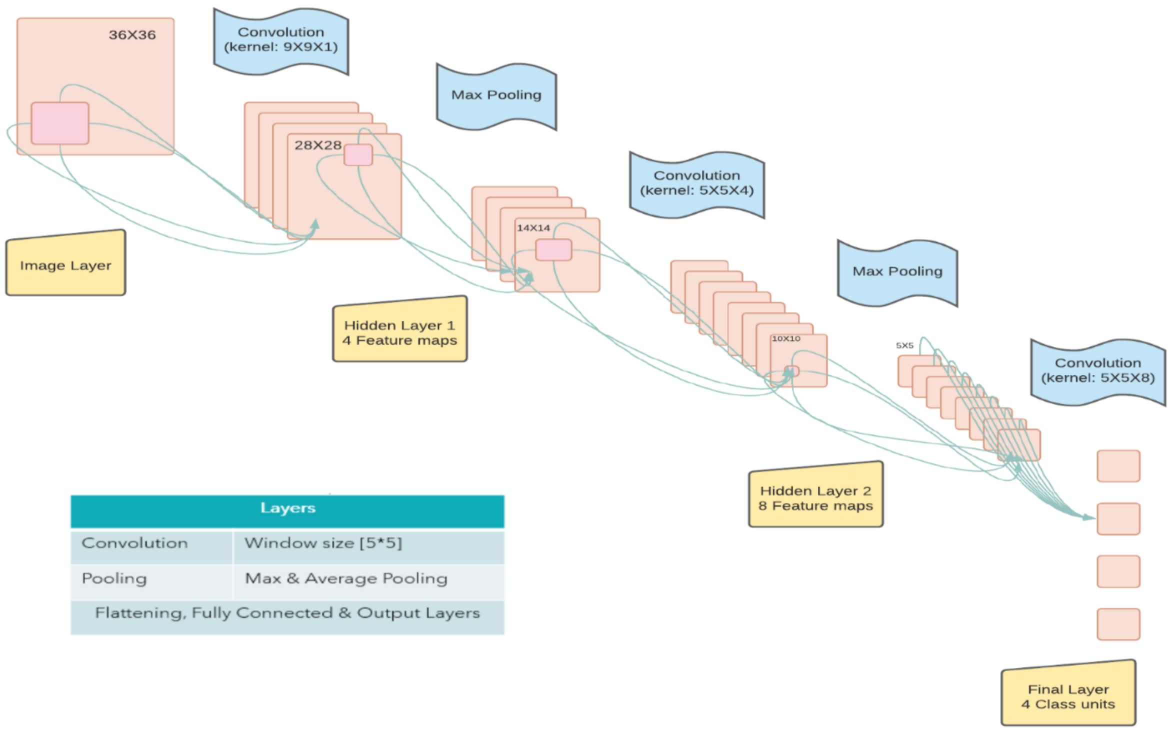


Figure 6: Image Labelling

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, depth, height. 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,


36X36
Convolution (kernel: 9X9X1)
Max Pooling
28X28
Convolution (kernel: 5X5X4)
14X14
Image Layer
Max Pooling
Hidden Layer 1 4 Feature maps
10X10
5X5
Convolution (kernel: 5X5X8)
Hidden Layer 2 8 Feature maps
Final Layer 4 Class units

Layers	
Convolution	Window size [5*5]
Pooling	Max & Average Pooling
Flattening, Fully Connected & Output Layers	

Figure 7: Modelling - Layers

Layers

Convolutional Layer**:**

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

Pooling Layer:

We have different pooling layers in our modelling, such as Max and Average pooling.

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.

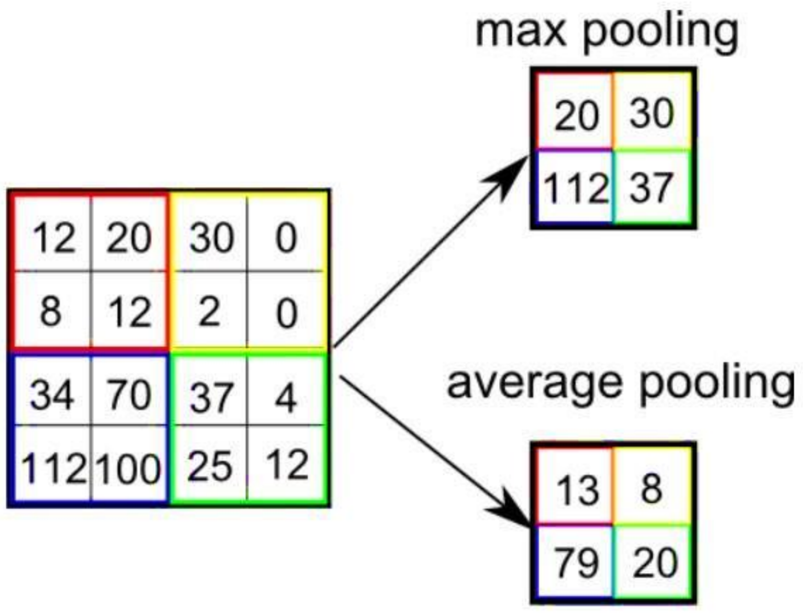


Figure 8: Pooling Layers

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.

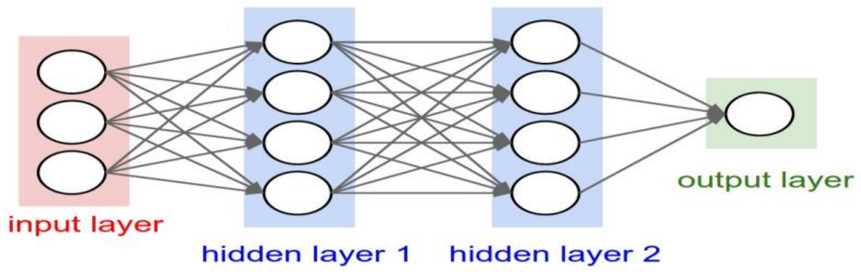
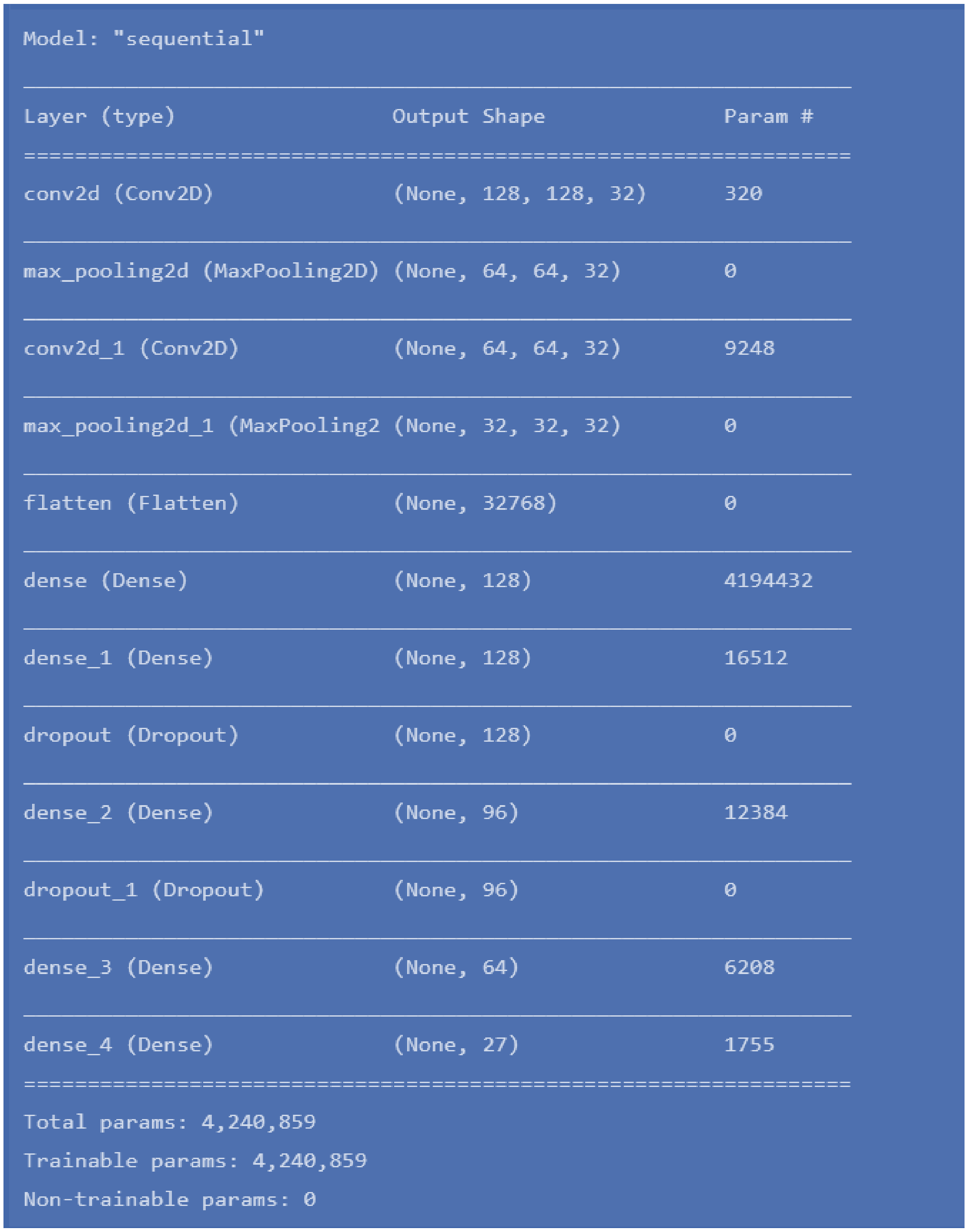


Figure 9: Fully Connected Layers

The model summary will give us complete and detailed information about the neural network structure we have used.



```
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, 1, and the sum of particular value in each label is one.

Testing

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

Incorrect predictions:

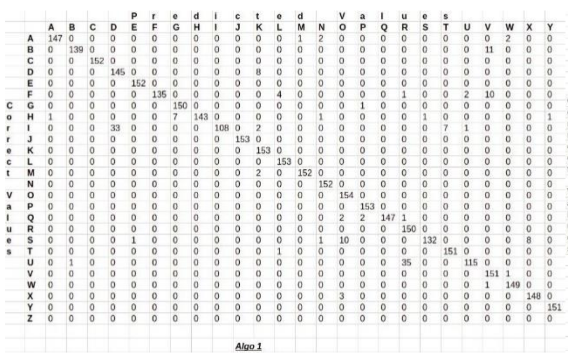
1. For I: T, D, K and I
2. For D: R and U
3. For S: M and N
4. For U: D and R

So, to handle this issue, we made three different classifiers that will help us classifying these sets:

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

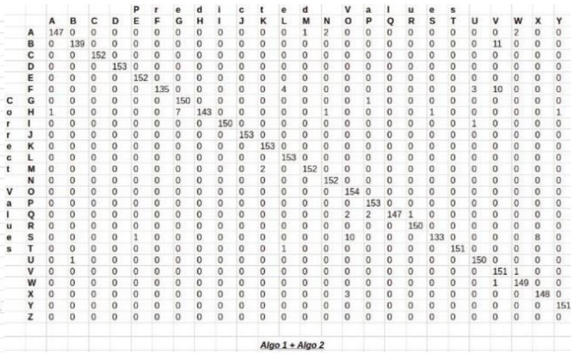
Evaluation

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.



						P	r	e	d	i	c	t	e	d		V	a	l	u	e	s					
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	A	147	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0
	B	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
	C	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	145	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	1	0	0	2	10	0	0	0
C	G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
o	H	1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
r	I	0	0	0	33	0	0	0	0	108	0	2	0	0	0	0	0	0	0	0	7	1	0	0	0	0
r	J	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	K	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	L	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
t	M	0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
V	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
a	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
l	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
u	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
e	S	0	0	0	0	1	0	0	0	0	0	0	0	0	1	10	0	0	0	132	0	0	0	0	8	0
s	T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
	U	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	115	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algo 1



						P	r	e	d	i	c	t	e	d		V	a	l	u	e	s					
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	A	147	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0
	B	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
	C	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	0	0	0	3	10	0	0	0
C	G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
o	H	1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
r	I	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
r	J	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	K	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	L	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
t	M	0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
V	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
a	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
l	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
u	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
e	S	0	0	0	0	1	0	0	0	0	0	0	0	0	0	10	0	0	0	133	0	0	0	0	8	0
s	T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
	U	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algo 1 + Algo 2

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.

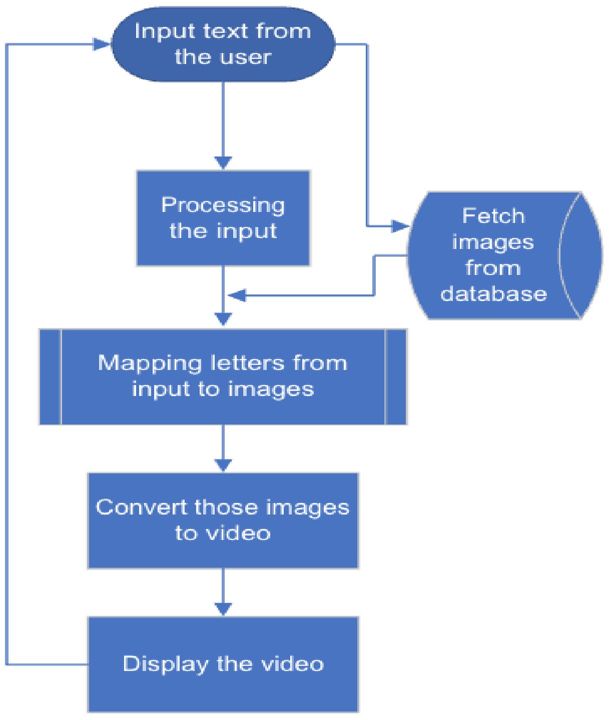

Input text from the user
Processing the input
Fetch images from database
Mapping letters from input to images
Convert those images to video
Display the video

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.

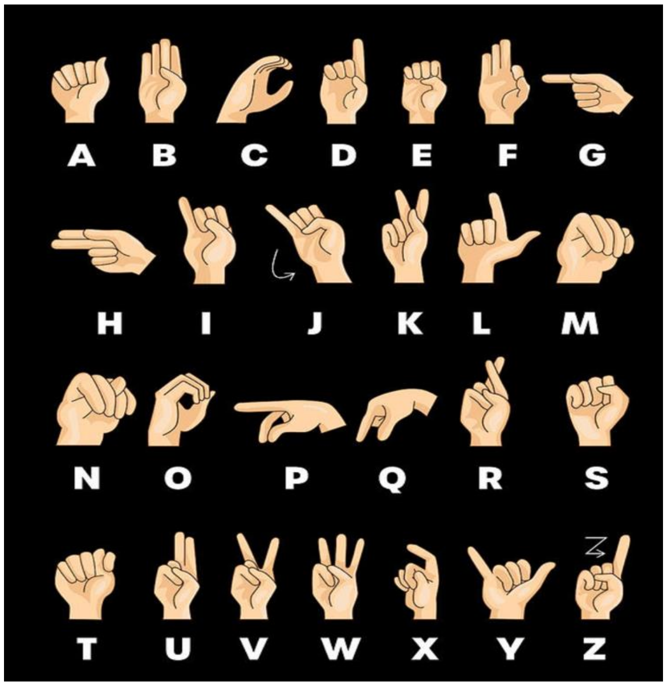


Figure 13: ASL Sign Language Alphabets

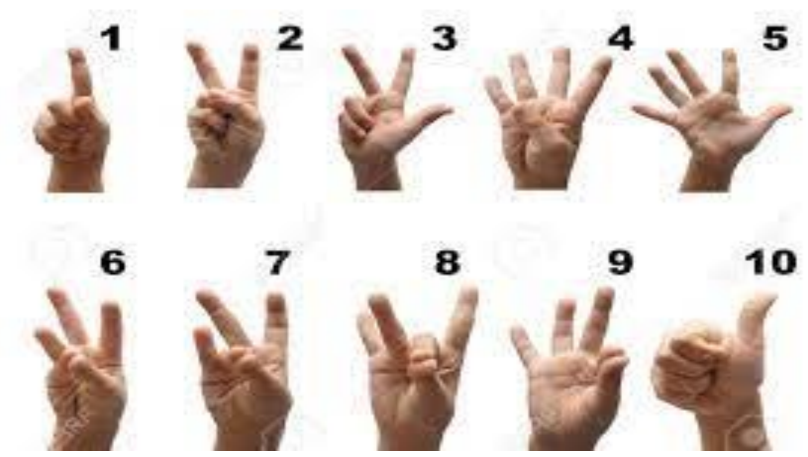


Figure 14: ASL Sign Language Numbers

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.

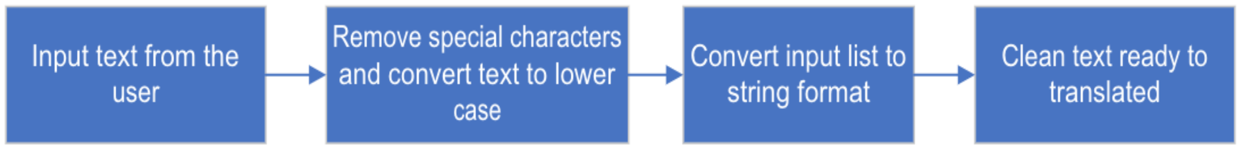

Input text from the user
Remove special characters and convert text to lower case
Convert input list to string format
Clean text ready to translated

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:

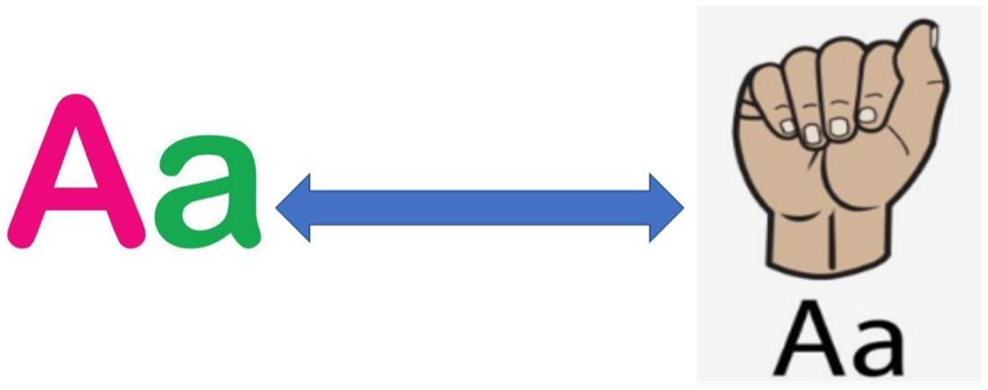


Figure 16: Character Mapping to ASL Image

Keywords & Libraries

The Python Imaging Library PIL "Pillow"

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

Figure 17: Python Imaging Library

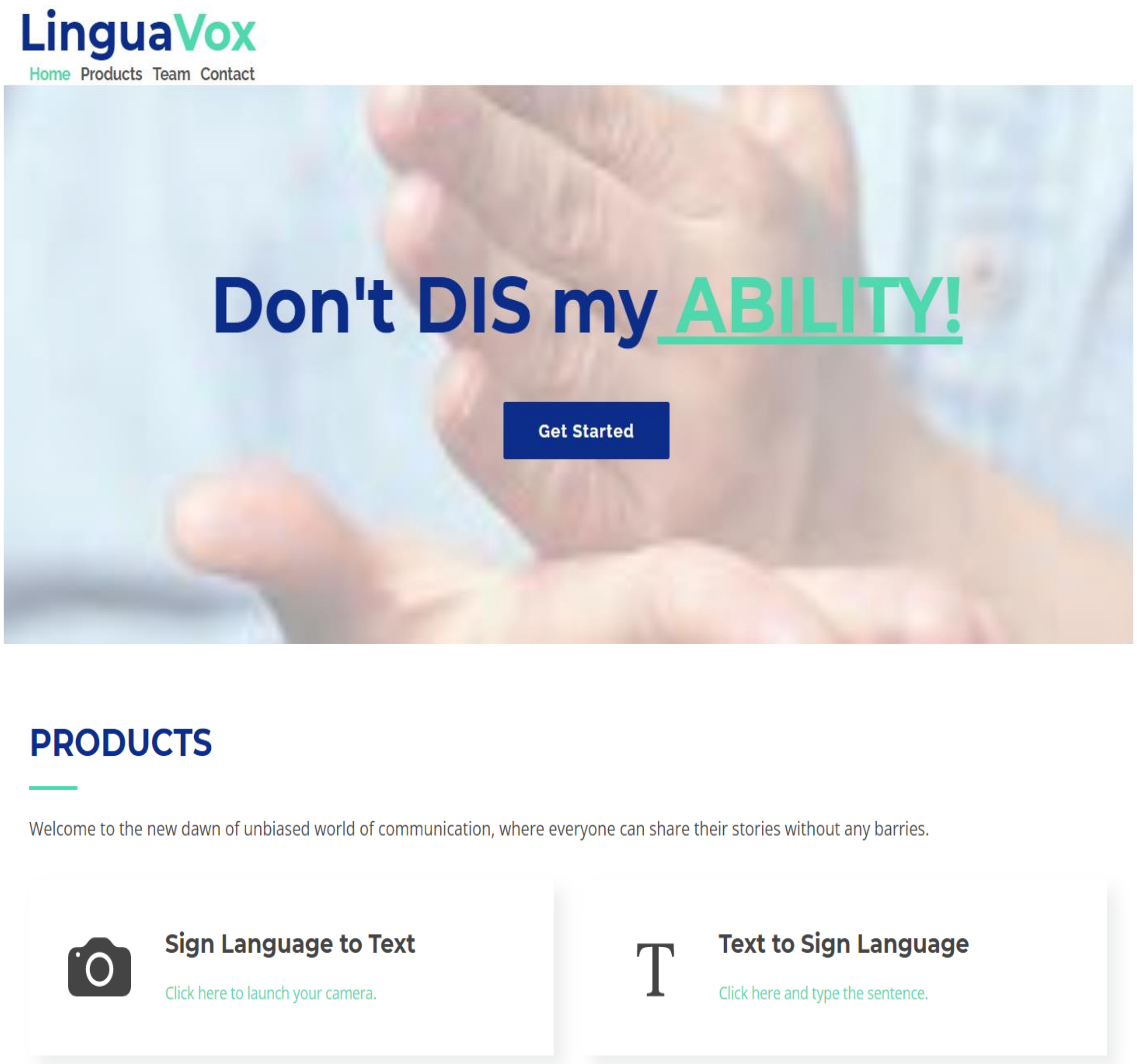
Imageio

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

Figure 18: Imageio

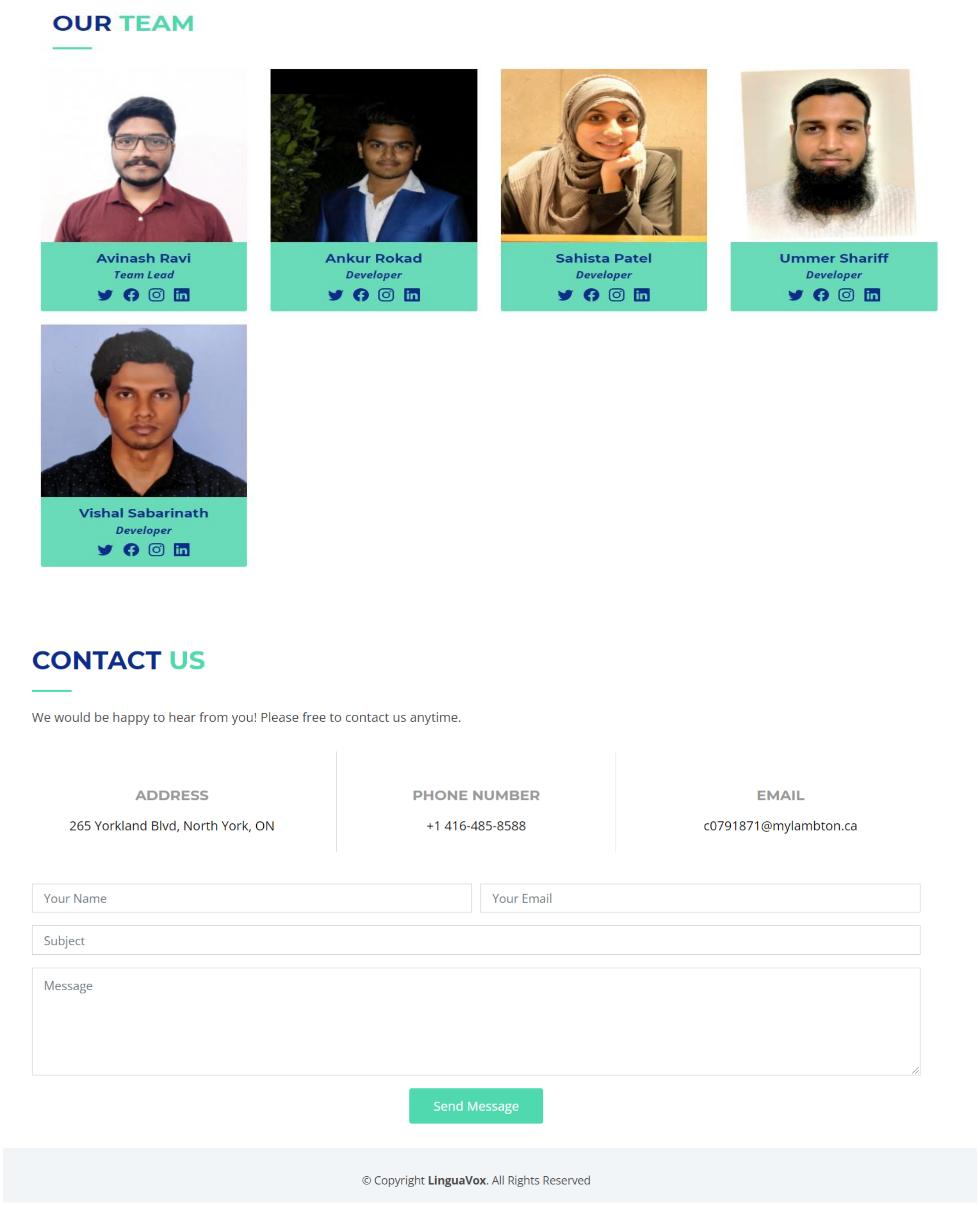
Results

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.



Welcome to the new dawn of unbiased world of communication, where everyone can share their stories without any barries.

Sign Language to Text
Click here to launch your camera.
Text to Sign Language
Click here and type the sentence.



OUR TEAM

Avinash Ravi
Team Lead

Ankur Rokad
Developer

Sahista Patel
Developer

Ummer Shariff
Developer

Vishal Sabarinath
Developer

CONTACT US

We would be happy to hear from you! Please free to contact us anytime.

ADDRESS	PHONE NUMBER	EMAIL
265 Yorkland Blvd, North York, ON	+1 416-485-8588	c0791871@mylambton.ca

© Copyright **LinguaVox**. All Rights Reserved

Figure 19: LinguaVox Web-View

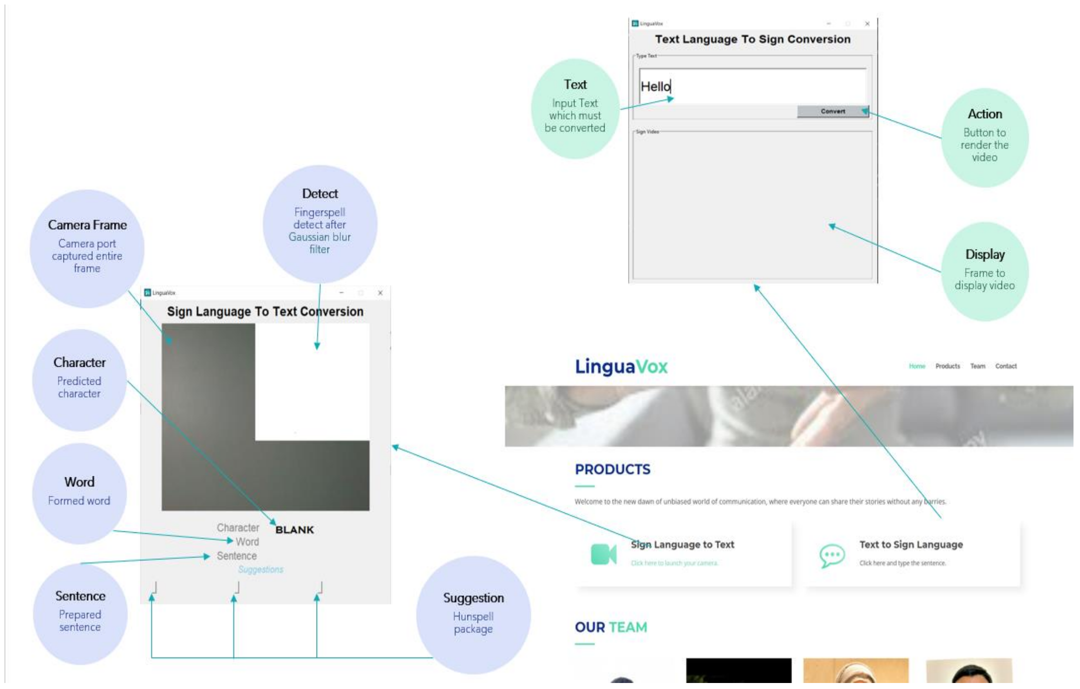

Text Language To Sign Conversion
Text
Input Text which must be converted
Hello
Convert
Action
Button to render the video
Display
Frame to display video
Detect
Fingerspell detect after Gaussian blur filter
Camera Frame
Camera port captured entire frame
Sign Language To Text Conversion
Character
Predicted character
Word
Formed word
Character
BLANK
Word
Sentence
Suggestions
Sentence
Prepared sentence
Suggestion
Hunspell package
LinguaVox
Home
Products
Team
Contact
PRODUCTS
Welcome to the new dawn of unbiased world of communication, where everyone can share their stories without any barries.
Sign Language to Text
Click here to launch your camera.
Text to Sign Language
Click here and type the sentence.
OUR TEAM

Figure 20: 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. Our model invokes the camera port on your device, captures your image continuously, and checks the result given by model – predicted letter. If the same result is thrown from the model continuously same letter, it will consider the letter and get their desired results. With the help of the Hunspell package, our model will also give the three suggestions which complete the word spell out by the signs given by the user.

Below is the screenshot of the application where the user is trying to spell out the 'Hello' sign to the system.

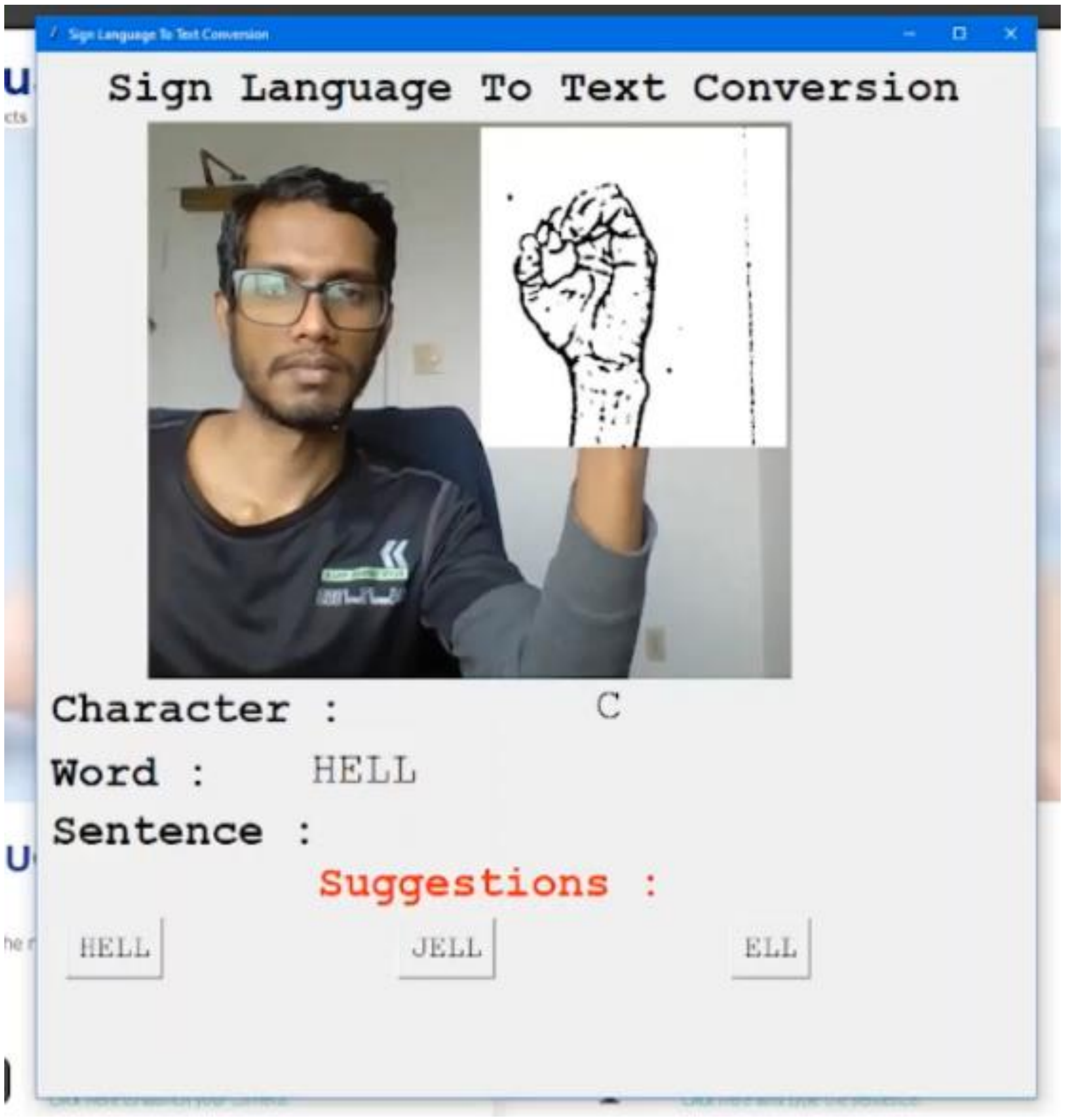

Sign Language To Text Conversion
Sign Language To Text Conversion
Character : C
Word : HELL
Sentence :
Suggestions :
HELL
JELL
ELL

Figure 21: 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.

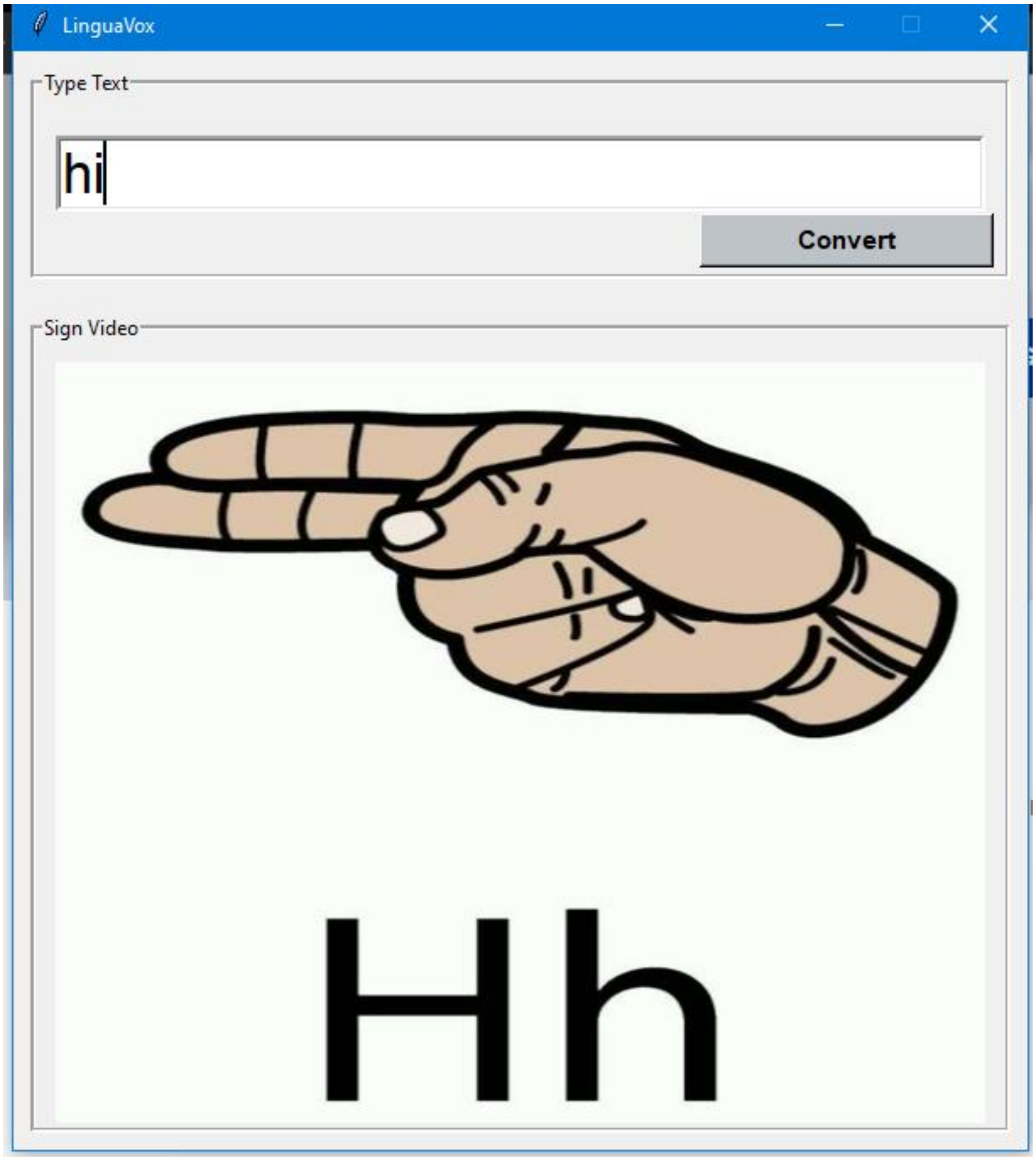


Figure 22: Text to Sign Language

Conclusions and Future Work

We intend to increase accuracy even in cluttered backgrounds by experimenting with various background removal approaches. We're also thinking about improving the Pre-Processing to make predicting gestures in low-light conditions much easier. The present work only works with ASL, but that might be extended to work with other native sign languages with the correct data and training. Although fingerspelling translator is used in this project, sign languages are also spoken in context, with each gesture indicating an item or a sentence. As a result, detecting contextual signing would demand further 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 Chinese Sign Language, British 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/

Bohra T.,Sompura S., Parekh K., Raut P. (2019). "Real-Time Two Way Communication System for Speech and Hearing Impaired Using Computer Vision and Deep Learning", 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT).

Bulus P. Bala, Song L. (2020). "Android App for Improvising Sign Language Communication in English and Hausa", International Journal of Advances in Scientific Research and Engineering.

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.

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.