Team Number: 1

Team Members:
Bolu Adubi
Alex Anderson
Caleb Bennetts
Matrim Besch
Ben Lottes

Project Name: TBD
SLI (Sign Language Interpreter)
SLAI (Sign Language AI)
VisualLanguages
Sign Language Tokenizer
Interpreting Signs

Project Synopsis:
Online application that recognizes sign language signs and interprets to text and vice versa for communication between people

Architecture:

To our knowledge there is not a currently available application that successfully translates sign language to complete sentences or phrases. From our research, the only thing that is available is being able to recognize individual signs in sign language using machine learning. Our goal is to design an application that allows a user to translate a sign or group of signs into a complete readable thought. One of the most important factors in the implementation of our translator software is the machine learning model. This is a crucial aspect of our project because the machine learning model will define how our translator learns and improves. Our group envisions a model that can perpetually improve provided that input is continuously received by the model. The program will primarily be written in Python, this decision was made due to its compatibility with various machine learning tools and libraries that will be invaluable to our project. Python also allows us to quickly make a prototype since it is a higher level language than something like C or C++. Our frontend will likely be made with Javascript, more specifically React since it is purpose built to create user interfaces quickly and easily. Our program will function by taking in video input of the user doing a sign or set of signs. Each pixel or pixel group represents an area our program will try to match with the correct sign. Our software will search through all of the available data to find the most accurate match. Illustration two below demonstrates our idea of how the neural network will process input. We will use OpenCV to facilitate video capture and filtering of the video input to reduce background noise and isolate important features/elements of the video stream. We will also use Tensorflow to aid with the machine learning necessary to determine what sign is being shown in the video feed and record that information in our program to be later translated. We plan on tokenizing the English language by, firstly, mapping letters and numbers to their respective signs. This will consist of understanding the structure and syntax of sign language and producing an advanced machine learning model that can precisely and accurately produce the correct word, letter, or number in English. Sign language has many subtle gestures that can be difficult to track, so we will need to ask the help of a professional to get a better grasp of the exact placement of certain parts of the hand or arms. Once our program is trained to recognize each individual letter and number we will then move to training it to recognize popular words, starting with the 50 most popular words in the English language. After letters, numbers, and popular words, we will tokenize popular phrases that can be articulated with a single sign. We can use these "tokens" to form an English sentence. To do this, we plan on using our knowledge of compilers to form a pseudo-compiler from Sign-language to English. We can essentially use the tokens to build up a parse tree that has leaves forming a complete sentence. One problem we will have to overcome with this approach is that sign language sometimes leaves out various supporting words that are explicitly spoken in English. We will have to find a way to identify when these words have been excluded and automatically add them into our sentence. Since the structure of sign language and English is slightly different like the example previously given along with many others, we will need to consult experts in both sign language and English to better understand what will be necessary to make an accurate and faithful translation. If we can accomplish basic translations and we still have time left in the project, we will train our translator by giving it more sign language data and more complicated Sign-Language sentences and phrases. Our group will continue to add to our database of signs until our program can accurately tokenize any single sign we input. To increase the efficiency of this process we can simply feed the inputs we test the software with into the training dataset. Illustration three below demonstrates this idea of adding to the data by having a function that turns images into network ready data. Eventually, our group will add features to our user interface so that the overall user experience is as positive as we can make it. For example, along with the output in text we can display the original input to the user so that it is clear what is being translated. To further better the user experience we may include features such as sign correction. A sign correction feature would involve giving the user our best guess of what they were trying to sign and then offering them a video tutorial of how to properly do the sign. This can be implemented through our machine learning model as our model can detect when signs are invalid. However, a feature like this would require extensive training. Another feature we hope to implement is being able to translate written English text into Sign-Language. This type of feature would allow the user to type an english phrase into a text box, and then our program will present images or videos of the correct signs to say the phrase in sign language. A tool like this can be as simple as a search bar that pulls up various sign images or videos based on keywords in the search. To implement this effectively we will pull the search results from our database of signs and present them to the user. Another potential feature that would offer an additional layer of functionality is a text to speech option. This type of feature can be added fairly easily with existing libraries such as gtts. After the user signs their input and receives the output in text format we hope they can select an option to have this sentence read aloud to them. Our application would be useful both for learning and practicing sign language as well as for public speakers using sign language. If our product was developed long term it could be applied to a language learning app similar to Duolingo, or even built into video conferencing apps like Zoom.

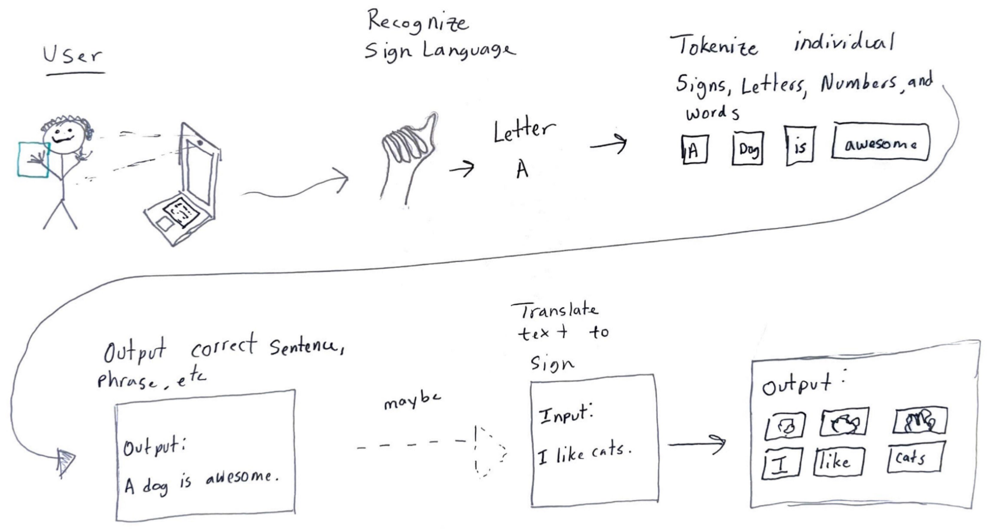

Illustration 1: Shown above is the overall procedure and functionality our program should have.

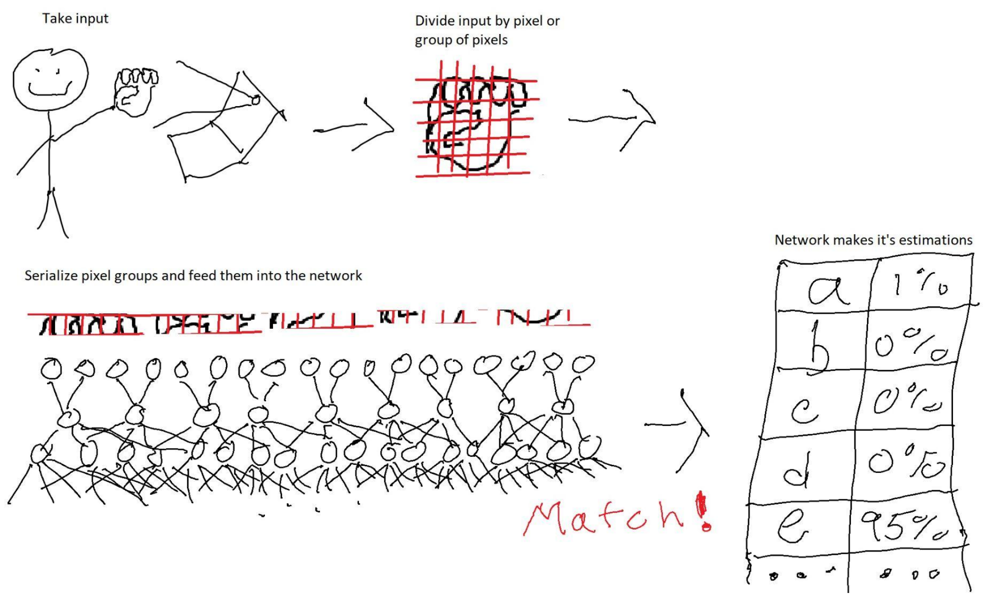
Take input
Divide input by pixel or group of pixels
Serialize pixel groups and feed them into the network
Network makes it's estimations
a 1%
b 0%
c 0%
d 0%
e 95%
Match!

Illustration 2: Here we demonstrate how our program will take input and match this input with its corresponding English token based on the best guess.

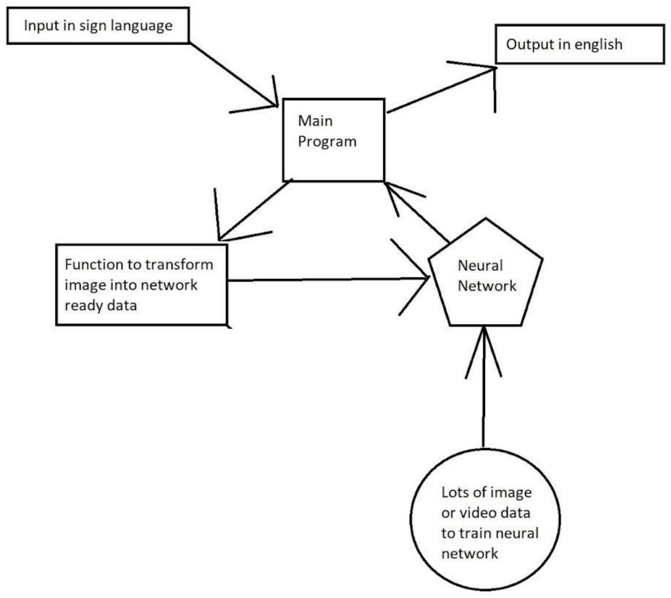
Input in sign language
Output in english
Main Program
Function to transform image into network ready data
Neural Network
Lots of image or video data to train neural network

Illustration 3: This diagram represents the flow of data throughout our program's execution.