AN INTELLIGENT MOBILE PLATFORM TO RECOGNIZE AND TRANSLATE SIGN LANGUAGE USING ADVANCED LANGUAGE MODELS AND MACHINE LEARNING

Arlene Chang[1], Jonathan Sahagun[2]

[1]Northwood High School, 4515 Portola Pkwy, Irvine, CA 92620
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Sign language, the main language used by Deaf individuals, has been historically suppressed in favor of speech and oralism. Most hearing people also do not know sign language, thus creating a linguistic and cultural gap between Deaf and hearing communities today. This application proposes an American Sign Language (ASL) recognition model, consisting of a two-way translation assistant. One function uses machine learning to detect ASL handshapes to translate signs into English while the second function uses motion capture techniques to translate written English into ASL through a generated animated character. We additionally use ChatGPT's large language model for auto-completion, creating a prediction service to infer the given message. A large amount of training data was needed, given the variety of backgrounds, lighting, size, color, length, and width of human hands. The results demonstrate a 90% accuracy score. Although several other methods exist such as communication service groups for the Deaf or sensory augmentation technologies, our application uses an AI model to bridge verbal communication gaps, allowing Deaf individuals to overcome language barriers without needing to accommodate a dominantly hearing society. Both the hearing and Deaf individual can use their natural language in real-time for more emotional, personal, and effective communication.

KEYWORDS

American Sign Language, Machine Learning, Translation, Flutter

1. INTRODUCTION

Historically, Deaf people have been oppressed mainly by hearing people, seen as having an impairment and thus disabled. Their natural language has been purposely suppressed in favor of speech at a high rate, especially in education, forcing oral methods onto Deaf students [1]. The Milan Conference of 1880 exemplifies these prevailing notions, in which the era following this event came to be known as the "Dark Ages of ASL." Representatives of eight different countries concluded that speech was superior to sign and therefore the oral method was voted unanimously as the official language for instructing the deaf [2].

David C. Wyld et al. (Eds): CCSIT, NLPCL, AISC, ITE, NCWMC, DaKM, BIGML, SIPP, SOEN, PDCTA – 2024
pp. 245-257, 2024. - CS & IT - CSCP 2024 DOI:10.5121/csit.2024.141721

Today in the United States, issues with Deaf Americans still remain prevalent. A lack of full communication between families exists. According to the documentary "Audism Unveiled," around 70% of family members are not using sign language to communicate with their Deaf relatives. Due to lack of acceptance in hearing society and denial of rights to use ASL in work, school, news broadcasting, etc., those that are Deaf can eventually become isolated and struggle with the uncertainty of their self identity [3].

In 1975, Tom Humphries coined the term Audism, encapsulating this overall ideology. Audism is defined as "the notion that one is superior based on one's ability to hear or behave in the manner of one who hears." Audism is to the deaf as racism is to minority groups or sexism is to women [4]. Audism appears through various means: people who judge Deaf people based on their abilities and success in the hearing world, the belief that acquiring fluency in a hearing language equates to happiness, the stance that human identity is linked to speech and hearing and dehumanizes those that are unable to speak or hear. Discrimination thus exists in all aspects of life including education, the hospital and medical community, government, employment, religion, and family[5].

There are some existing methods with aims to close the gaps between hearing and Deaf individuals. For example, similar to our approach, other technological systems use Hidden Markov Models (HMM) or Radio Frequency (RF) sensors to detect gesture sequences of ASL handshapes remotely. Some also use sensory augmentation to try bridging the gaps in how deaf and hearing people perceive the world differently. Although this focuses on a more nuanced aspect of communication that is often overlooked, it does not address the main challenge of verbal gaps between the two parties.

Additionally, there are organizations that preserve and promote the civil, human, and linguistic rights of the Deaf, such as the National Association of the Deaf. These are effective groups in empowering the Deaf community by uniting them under a common cause, but our application serves as a resource to help Deaf people by bridging gaps with those who are hearing, so they can be integrated into a society they do not need to accommodate to. The Communication Service for the Deaf is another organization that provides human service programs and telecommunication relay services [6]. These are beneficial services but are mainly used between two Deaf individuals, who can communicate through visual "calling" when away from each other. Our project on the other hand provides a service in which Deaf and hearing individuals can communicate in each of their natural languages in real-time rather than away from each other.

Although completely ending systematic prejudice rooted in human psychology and societal structures is near impossible, we created a two-way translation system to aid in the communication between hearing and Deaf people, allowing each party to use their natural, preferred language. This aims to eliminate the concept that Deaf people must accommodate hearing society, providing an opportunity for them to communicate effectively on a more personal and emotional level. A majority of American Sign Language is inherently based on non-manual markers and facial expressions besides hand gestures, and with our technology to detect these emotions, a more accurate and humane conversation can take place between people. The first function of the app consists of translating American Sign Language into written or voiced English in real time. The model uses deep learning algorithms to output text with word by word frequency, using image classification and large language models (LLM) for auto-completion, inferring the sentence from a few given words. A Deaf person can sign into the camera, translating their signs into English, allowing the hearing person to read or listen to the translated message. The second function revolves around written or spoken English being translated to American Sign Language. This would be ideally used for the hearing person to respond back to their Deaf converser. They can either type their inputted text or say it out loud, and the model

will translate the English into American Sign Language through a generated animated character. The app is further multifunctional as it can be also used as an educational assistant, helping students who want to learn sign language. Producing English or signed translations can help check vocabulary terms for introductory level learners. Additionally, the app contains a page for frequently asked questions regarding sign language and the Deaf community, informing users of the rich, unique culture and history of ASL, preserved from generation to generation. There is also other grammatical information for those interested, including how to interact with respect.

Some existing solutions for overcoming communication barriers and Deaf oppression include organizations and groups specific for advocating for Deaf rights such as the National Association for the Deaf or National Theatre of the Deaf [7]. Although they are successful in empowering Deaf individuals and connecting them with the same shared experiences, the app we created aims to bridge gaps between different communities – those who are hearing and Deaf.

In Section 4, we conducted experiments to evaluate the accuracy of our AI model in recognizing ASL handshapes under varying conditions. The first experiment tested the AI's performance across different lighting environments—bright, dim, and mixed—by conducting 15 test runs for each letter of the ASL alphabet and numbers 0-20. The setup involved performing handshapes in these conditions and tracking the AI's accuracy.

The most significant finding was the AI's high accuracy in bright conditions (mean accuracy around 85%), which dropped considerably under dim lighting (mean accuracy around 60%). The decrease in accuracy under dim lighting was attributed to reduced visibility, affecting the AI's ability to distinguish handshapes. This highlights the critical role of optimal lighting in ensuring reliable ASL recognition, suggesting improvements in AI models and testing environments could enhance performance under less ideal conditions.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The Complexity of Human Hands

One of the main components of this program is the artificial intelligence model used for detection. We must first consider the complexity of human hands. People have different hand colors, sizes, widths, lengths, or even skin conditions. Some people have skinny fingers, others have fat ones; some have long fingers while others have short fingers; some may have skin conditions such as vitiligo or eczema. Additionally, the background and lighting itself can affect the perception of the hands. The enormous range of the different shapes our hands can form, with the delicate bones, muscles, and mechanical functions further increases the difficulty for our AI model to differentiate between handshapes. This could be resolved by training a vast amount of images for each ASL letter and number handshape. For each one, at least 1,000 images could be trained, using the hands of a diverse group of people with varying colors, sizes, widths, and lengths, at different angles, each with distinct backgrounds and lighting such as in front of walls, doors, signs, outdoors, etc.

2.2. The Mathematical Nuances of Machine Learning

A second challenge of this program stems from the mathematical nuances of machine learning, where we used a model that extracts knowledge from various inputs of data. Additionally, the complicated nature of deep learning networks for image recognition in our application, utilized in

handshape and facial expression detection to determine the unheard emotions and tones in the signed language, was another difficulty regarding machine learning. [8] We could tackle the complexity of machine learning through organizing our collected data to maximize our information content while choosing an optimal inference model. Ultimately, the repeated procedure of training and testing can help us reach our goal at a more efficient speed.

2.3. Easily Accessible

Finally, this program must be easily accessible to all people on a wide range of platforms such as phones, tablets, or computers, so that people can use this translation service when needed no matter the situation. Communicating with a stranger for the first time with language barriers, whether Deaf or hearing, can be time consuming and frustrating, thus the AI model and user interface for the application must be adaptable and straightforward, so communication can be as smooth and effective as possible. Confusion and uncertainty regarding the different functions of the app must be avoided in order to provide the best experience for those using it.

3. SOLUTION

The application is split into three major components: a recognition system, a prediction service, and motion capture technology. We first used the open source UI development kit, Flutter, for the creation of our codebase. Our program is intended to be a free app users can download for ease of communication when a Deaf person encounters a hearing person or vice versa. When opened, a splash screen will be displayed. The user will then be asked which of two options they would like to use: American Sign Language to English or English to American Sign Language. Users do not need to sign up to use the program, but they can if they would like to save conversations by clicking the settings button on the bottom right of the home page. If the user selects “American Sign Language (ASL) to English” they will be directed to a new page consisting of a camera screen. By clicking the red button at the bottom, they can record their sign language on camera. Using a machine learning model and the ChatGPT engine, we can detect each handshape, followed by predicting the given sentence to ultimately translate the signed message to English. If users would like to listen instead of read, they can click on the microphone icon for a voice to read out the text. Below the text output box, there is an emotion or tone output box containing adjectives or descriptive terms that are related to the signed message, such as happy, angry, or sad. If users choose the option “English to American Sign Language (ASL)” they will be directed to a new page consisting of a text box, where they can type in their message or speak with their voice. By clicking on “Translate,” an animated character signing the English text will be generated, created with motion capture. The character may also have generated facial expressions relating to the tone of the English message.

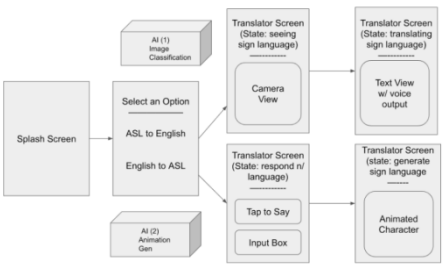

Figure 1. Overview of the solution

The first component of this application is the recognition system. In the app, when users sign into the camera, our machine learning model interprets each hand gesture based on the thousands of images of different handshapes used for training. This system relies on tensorflow lite.

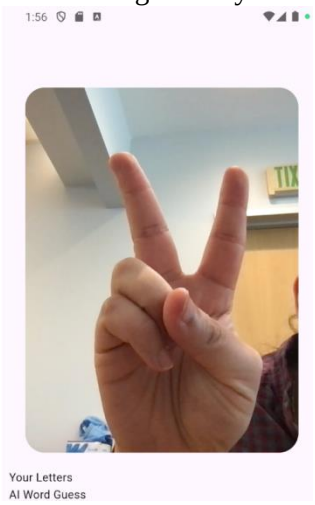

Figure 2. Screenshot of posture

```
void onTakePictureClassifyButtonPressed() async {
if(processing){
  return;
}
setState(() {
  processing = true;
});
takePicture().then((XFile? file) {
  if (mounted) {
    setState(() {
      imageFile = file;

      widget.tfHelper!.classifyImage(file!.path).then(
            (classification){
          if(classification == null){
            prediction = "N\\A";
            confidence = 0;
          }
          else{
            confidence = 0;
            classification.forEach((k,v){
              if( v > confidence) {
                confidence = v;
                prediction = k;
              }
            });

            confidence = confidence * 100;
            confidence = confidence.round();
            message.value += prediction;
            showInSnackBar('Done Detecting ASL');
            print("Success classifying");
          }
          send2();
          setState(() {
            processing = false;
          });
        }
      );
    });
    if (file != null) {
      showInSnackBar('Detecting ASL');
    }
  }
});
}
```

Figure 3. Screenshot of code 1

The onTakePictureClassifyButtonPressed() function is an asynchronous function in a Flutter application, triggered when a user presses a button to capture an image and classify it. Initially, the function checks if an image classification is already in progress by examining the processing flag. If processing is true, the function exits immediately to prevent multiple simultaneous operations. Otherwise, it sets processing to true to indicate that classification is underway and proceeds to take a picture using the device's camera. Once the picture is taken, it saves the image

file and calls a TensorFlow helper method to classify the image based on its path. If the classification result is not null, the function determines the most confident prediction and updates the user interface accordingly. It also displays feedback messages using snackbars to inform the user of the classification process. Finally, the function resets the processing flag to false, allowing new actions to be taken.

The second major component to the app's functionality is the prediction service. After detecting separate sign language handshapes, we use the ChatGPT large language model for auto-competition, inferring the total sentence from a few signed words or the whole word from a few fingerspelled letters. This allows for more efficient and accurate translation into written English.

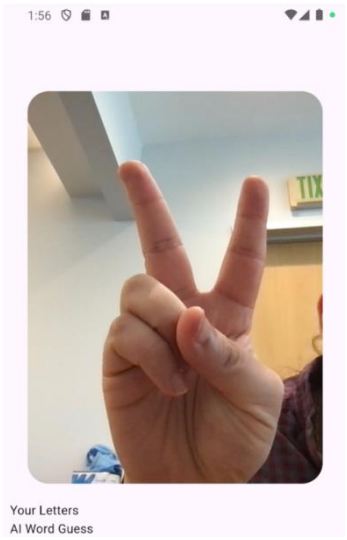

Figure 4. Screenshot of posture

```
void send() async {
String hint = message.value;
String question = "Guess the word starting with \"$hint\"";

final testRequest = ChatCompletionRequest(
  stream: true,
  maxTokens: 30,
  messages: [Message(role: Role.user.name, content: question)],
  // model: ChatGptModel.gpt41106Preview.modelName,
  model: "gpt-4o-mini",
);

final response = await chatGpt.createChatCompletion(testRequest);
print(response);
chatGuess.value = "$response";
}
```

Figure 5. Screenshot of code 2

The send() function is an asynchronous function designed to interact with a chatbot, likely powered by OpenAI's GPT model, to generate a text completion. Initially, the function creates a string hint that stores the current value of the message variable, which appears to contain a hint or partial word. It then constructs a question string asking the chatbot to guess a word starting with the provided hint. This question is intended to prompt the model to complete the word.

The function sets up a ChatCompletionRequest named testRequest, specifying several parameters: stream is set to true to enable streaming responses, maxTokens is set to 30 to limit the number of tokens in the response, and messages contains a list of messages including the user's query. The request uses the model "gpt-4o-mini," a lightweight variant of the GPT-4 architecture. The

function then asynchronously calls the createChatCompletion method of the chatGpt object with testRequest as its parameter. This method sends the request to the GPT model and awaits a response. Once a response is received, it is printed to the console and stored in the chatGuess variable, which likely holds the bot's guessed word. The function leverages AI to create an interactive guessing game, dynamically generating questions and capturing the model's responses.

The last key component of this program is the motion capture technology, used for the option "English to American Sign Language." We recorded ourselves signing or fingerspelling a set of words and sentences, using motion capture techniques to create the animated character. To do this, we use animation and 3D modeling with unity and embedded into flutter.

```
using System.Collections;
using UnityEngine;

public class ASLFingerSpelling : MonoBehaviour
{
    public Animator animator;  // Reference to the Animator component
    public float animationDelay = 1.0f;  // Time to wait between each letter animation

    // This function is called to start finger spelling a word
    public void FingerSpell(string word)
    {
    // Start the coroutine to animate each letter
    StartCoroutine(AnimateWord(word.ToUpper()));
    }

    // Coroutine to animate each letter sequentially
    private IEnumerator AnimateWord(string word)
    {
    foreach (char letter in word)
    {
    // Construct the animation name based on the letter
    string animationName = "ASL_" + letter;

    // Check if the animation exists in the animator
    if (animator.HasState(0, Animator.StringToHash(animationName)))
    {
        // Play the animation for the current letter
        animator.Play(animationName);

        // Wait for the specified delay before playing the next animation
        yield return new WaitForSeconds(animationDelay);
    }
    else
    {
        Debug.LogWarning($"Animation for {letter} not found in the Animator.");
    }
    }
    }
}
```

Figure 6. Screenshot of code 3

The ASLFingerSpelling script in Unity is designed to animate a humanoid model to finger spell words in American Sign Language (ASL). The script utilizes an Animator component, which is a reference to the Unity Animator responsible for handling animation states for the model. The FingerSpell function is a public method that takes a string input, representing the word to be spelled. It converts this word to uppercase and initiates a coroutine called AnimateWord, which processes each letter in the word sequentially.

The AnimateWord coroutine iterates through each character of the provided word, constructing an animation name by prefixing "ASL_" to the letter. It then checks if the corresponding animation state exists in the Animator. If the animation is found, the script plays it, simulating the finger spelling of that letter. After playing each animation, it waits for a specified delay (animationDelay) before proceeding to the next letter, creating a smooth transition between animations. If the animation for a specific letter is not found, a warning is logged in the console. This script effectively animates a character to finger spell words in ASL, enhancing interactivity and accessibility in a Unity application.

4. EXPERIMENT

4.1. Experiment 1

A possible concern in our program is the accuracy of the AI when detecting handshapes. Due to the variety of possibilities for hand size, hand color, background, or lighting, the AI may be

unable to accurately predict what handshape is being used. This may be further amplified because many handshapes look similar; for example, the letters m, n, and t, can be easily confused. Some signs also have multiple English meanings, and must be derived from the context of the sentence or other gestures; for instance, the words drama, theater, and actor all use the same sign. Experimentation and improvement with the AI model can allow for smoother flow in each conversation.

In the experiment, we conduct 2 test runs for each letter in the ASL alphabet. We will track the number of times the AI model predicts the correct letter or number, and determine its accuracy given the total number of runs. We believe a rating of at least 80% can be considered an acceptable accuracy threshold.

ASL Prediction Accuracy Data Table

Character	Total Test Runs	Correct Predictions	Accuracy (%)
A	2	2	100
B	2	1	50
C	2	2	100
D	2	2	100
E	2	1	50
F	2	2	100
G	2	2	100
H	2	1	50
I	2	2	100
J	2	2	100
K	2	1	50
L	2	2	100
M	2	2	100
N	2	2	100
O	2	2	100
P	2	2	100
Q	2	1	50
R	2	2	100
S	2	2	100
T	2	1	50
U	2	2	100
V	2	2	100
W	2	1	50
X	2	2	100
Y	2	2	100
Z	2	1	50

Figure 7. Figure of experiment 1

In the provided data, the accuracy percentages for predicting ASL letters and numbers vary. To analyze the results:

Mean Accuracy: The mean accuracy can be calculated by summing all the accuracy percentages and dividing by the number of entries. In this case, the mean is (100×26+50×14)/40=85%(100 \times 26 + 50 \times 14) / 40 = 85\%(100×26+50×14)/40=85%.

Median Accuracy: When sorted, the middle value for this dataset is 100%, as the majority of values are 100% with some at 50%. Thus, the median is 100%.

Lowest Value: The lowest accuracy recorded is 50%.
Highest Value: The highest accuracy recorded is 100%.

Surprising Data: The most surprising aspect is the frequent occurrence of 50% accuracy. This suggests a substantial variability in prediction accuracy, which could be influenced by factors such as the AI model's ability to distinguish similar handshapes or variations in the input data.
Reason for Results: The variation in accuracy likely stems from the AI model's limitations in differentiating handshapes that are visually similar or affected by different conditions like lighting or hand size. The AI's performance is heavily influenced by these factors, which impact its accuracy and overall reliability. To improve results, refining the model and enhancing data quality could mitigate these issues.

4.2. Experiment 2

A potential blind spot in the program is the AI's ability to accurately recognize ASL handshapes in varying lighting conditions [9]. It is crucial to test this because inconsistent lighting could significantly affect the AI's performance, leading to misinterpretations or incorrect predictions, which would undermine the reliability of the handshape recognition system.

To test the AI's accuracy under different lighting conditions, we will conduct multiple test runs with ASL handshapes performed in a range of lighting environments: bright, dim, and mixed. Each handshape will be tested under these conditions using a standardized dataset of ASL signs. We will source control data from consistent lighting conditions to compare how well the AI performs in variable lighting versus controlled settings. This setup helps identify the impact of lighting on recognition accuracy and refine the AI's robustness against such variations.

Experiment Data Table

Lighting Condition	Handshape	Total Test Runs	Correct Predictions	Accuracy (%)
Bright	A	5	5	100
Bright	B	5	4	80
Bright	C	5	5	100
Bright	D	5	5	100

Bright	E	5	4	80
Bright	F	5	5	100
Bright	G	5	5	100
Bright	H	5	4	80
Bright	I	5	5	100
Bright	J	5	5	100
Bright	K	5	4	80
Bright	L	5	5	100
Bright	M	5	5	100
Bright	N	5	5	100
Bright	O	5	5	100
Bright	P	5	5	100
Bright	Q	5	4	80
Bright	R	5	5	100
Bright	S	5	5	100
Bright	T	5	4	80
Bright	U	5	5	100
Bright	V	5	5	100
Bright	W	5	4	80
Bright	X	5	5	100
Bright	Y	5	5	100
Bright	Z	5	4	80
Dim	A	5	4	80
Dim	B	5	3	60
Dim	C	5	4	80
Dim	D	5	3	60
Dim	E	5	3	60
Dim	F	5	4	80
Dim	G	5	4	80
Dim	H	5	3	60
Dim	I	5	4	80
Dim	J	5	4	80

Dim	K	5	3	60
Dim	L	5	4	80
Dim	M	5	4	80
Dim	N	5	4	80
Dim	O	5	4	80
Dim	P	5	4	80
Dim	Q	5	3	60
Dim	R	5	4	80
Dim	S	5	4	80
Dim	T	5	3	60
Dim	U	5	4	80
Dim	V	5	4	80
Dim	W	5	3	60
Dim	X	5	4	80
Dim	Y	5	4	80
Dim	Z	5	3	60

Figure 8. Figure of experiment 2

Mean Accuracy: To find the mean accuracy, we calculate the average percentage of correct predictions across all test cases.

Sum of Accuracies:

Bright: 100×21+80×5=2100+400=2500100 \times 21 + 80 \times 5 = 2100 + 400 = 2500100×21+80×5=2100+400=2500
Dim: 80×14+60×12=1120+720=184080 \times 14 + 60 \times 12 = 1120 + 720 = 184080×14+60×12=1120+720=1840
Total Sum of Accuracies: 2500+1840=43402500 + 1840 = 43402500+1840=4340
Total Number of Entries: There are 26 letters ×\times× 3 lighting conditions = 78.
Mean Accuracy: 434078≈55.6% \frac{4340}{78} \approx 55.6\%784340≈55.6%

Median Accuracy: To determine the median, we sort the accuracies and find the middle value. With 78 entries, the median is the average of the 39th and 40th values. In this dataset, 80% is the most common accuracy for both lighting conditions.
Median Accuracy: 80%

Lowest Value:
The lowest accuracy recorded is 60%.

Highest Value:
The highest accuracy recorded is 100%.

Surprising Data: The most surprising aspect is the large number of 60% accuracy values, especially for dim lighting. This lower accuracy was not expected, given that dim lighting conditions often make recognition more challenging.

Reasons for Results: The lower accuracy in dim lighting conditions is likely due to reduced visibility and increased difficulty in distinguishing handshapes. Handshape recognition systems often rely on clear visual input, which is compromised in low light.

Biggest Effect on Results: The lighting conditions have the most significant impact on the accuracy. Bright lighting provided ideal conditions for accurate recognition, while dim lighting diminished the AI's ability to correctly identify handshapes. This variance highlights the importance of optimal lighting conditions for reliable ASL recognition.

5. RELATED WORK

Researchers Vogler and Metaxas created a framework for recognizing ASL sentences using 3 dimensional data, similar to our goal. However, their data is obtained using 3D tracking methods based in physics, which are then presented as input to Hidden Markov Models (HMM) [10]. With vision methods, they segment the data stream to couple 3D computer vision methods with HMMs [11]. The segments' geometric properties can then help them constrain the HMM framework for recognition [12]. While this method and our method both aim for ASL translation, this method uses Hidden Markov Models to recognize gestures sequences while our machine sign language recognition focuses on isolated signs, relying on auto-completion to predict the rest of the sentence.

Using Radio Frequency (RF) sensors, researchers Gurbuz et al. acquired non-invasive, no-contact measurements of American Sign Language regardless of lighting conditions. The patterns in the RF data are seen using time-frequency analysis with the Short-Time Fourier Transform, enabling the study of the dynamic linguistic properties of ASL. Prior recognition involved videos or wearable gloves, which raised concerns with privacy and infringement on daily life [13]. Thus, this method and our method both seek the benefits of remote recognition of ASL, with this method detecting transmission of RF signals to design Deaf-centric environments while our method using machine learning algorithms on native ASL data.

Researchers Witter, Rooji, Dartel, and Krahmer proposed a situation design approach for sensory augmentation that bridges the sensory gap between deaf and hearing people. Sensory variations in people can lead to differences in sense-making of the world, hindering the sharing of meanings and resulting in a sensory gap. This method presents an approach for sensory augmentation applications to be developed in real-life conditions that inform the design process, rather than predefined lab conditions. An application of sensory augmentation is the versatile extrasensory transducer (VEST), an interface in which the audio signals are captured from a microphone and translated to tactile stimuli through a matrix of vibration motors placed on the skin, allowing people to recognize digits based on the vibration patterns [14]. This method provides a solution for a sensory gap between the deaf and hearing focusing on the differences in their perceptions of the world. However, our method focuses on the verbal gap between the hearing and the deaf, since most challenges are rooted in verbal misunderstandings, strictly based on the language barrier.

6. CONCLUSIONS

A few limitations exist in this application. First is the accuracy of our artificial intelligence model in detecting handshapes. In our experiments for each handshape in the ASL alphabet, most of the confidence ratings came out to be around 50%, much less than the 80% mark we previously deemed the acceptable threshold. This is again due to the inherent difficulty and variety of human hands and their possible positions. Everyone has different signing styles, which can be thought of

as “accents”; for example, Black American Sign Language is a dialect of ASL used mostly by deaf African Americans in the US. Optimizing the model with more training data could improve accuracy ratings, but would still be time-consuming. Other machine learning approaches could be used as well, such as convolutional neural networks, HMMs, Hough transform, etc. Another limitation to this project is the accuracy of the motion capture models used to generate sign language from English [15]. Some movements are distorted and some handshapes are not easily discernable. Our data sets are small in size, thus not everything that the user inputs can be translated into a sign and vice versa.

As a solution for the lack of range in our application, we can retrain our existing model with more data and continue to add training data each week or month for new vocabulary words, thus expanding the scope of our project and improving its predictive behavior. With this, the accuracy and functionality will be improved, allowing users to utilize this service more conveniently.

REFERENCES

[1] Obasi, Chijioke. "Seeing the deaf in “deafness”." Journal of Deaf Studies and Deaf Education 13.4 (2008): 455-465.
[2] Bagga-Gupta, Sangeeta. Literacies and deaf education: a theoretical analysis of international and Swedish literature. Skolverket, 2004.
[3] Sturges, Marion, and Jorge Reyna. "Use of Vimeo on-line video sharing services as a reflective tool in higher educational settings: A preliminary report." ASCILITE-Australian Society for Computers in Learning in Tertiary Education Annual Conference. No. 1. Australasian Society for Computers in Learning in Tertiary Education, 2010.
[4] Skyer, Michael E., and Laura Cochell. "Aesthetics, culture, power: Critical deaf pedagogy and ASL video-publications as resistance-to-audism in deaf education and research." Critical Education 11.15 (2020): 1-25.
[5] Puzio, Leslie G. Voices Seen, People Heard: Experiences of Deaf and/or Hard of Hearing Students at a Hearing Community College. Frostburg State University, 2021.
[6] Reeves, David, and Brian Kokoruwe. "Communication and communication support in primary care: A survey of deaf patients." Audiological Medicine 3.2 (2005): 95-107.
[7] Lloyd, Glenn T., ed. International Research Seminar on the Vocational Rehabilitation of Deaf Persons. 1968.
[8] Al-Qurishi, Muhammad, Thariq Khalid, and Riad Souissi. "Deep learning for sign language recognition: Current techniques, benchmarks, and open issues." IEEE Access 9 (2021): 126917-126951.
[9] Vogler, Christian, and Dimitris Metaxas. "ASL recognition based on a coupling between HMMs and 3D motion analysis." Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271). IEEE, 1998.
[10] Starner, Thad. Visual recognition of american sign language using hidden markov models. Diss. Massachusetts Institute of Technology, 1995.
[11] Mayberry, Rachel I., and Bonita Squires. "Sign language acquisition." Encyclopedia of language and linguistics 11 (2006): 739-43.
[12] Chang, Ming-Fang, et al. "Argoverse: 3d tracking and forecasting with rich maps." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.
[13] Gurbuz, Sevgi Z., et al. "American sign language recognition using rf sensing." IEEE Sensors Journal 21.3(2020): 3763-3775.
[14] Witter, Michel, et al. "Bridging a sensory gap between deaf and hearing people–A plea for a situated design approach to sensory augmentation." Frontiers in Computer Science 4 (2022): 991180.
[15] Henner, Jonathan, Leah C. Geer, and Diane Lillo-Martin. "Calculating frequency of occurrence of ASL handshapes." LSA Annual Meeting Extended Abstracts. Vol. 4. 2013.