AN INTELLIGENT SIGN LANGUAGE LEARNING AND PROMOTION STATION SYSTEM USING ARTIFICIAL INTELLIGENCE AND COMPUTER VISION

Junhan Wang[1], Jonathan Sahagun[2]

[1]Santa Margarita Catholic High School, 22062 Antonio Pkwy, Rancho Santa Margarita, CA 92688
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

In this paper, we tackle the pressing communication gap between the Deaf and hearing communities, an issue affecting millions of individuals worldwide [1][2]. The proposed solution is a machine learning-powered application that translates sign language into text in real-time, allowing Deaf and hearing individuals to communicate directly [3]. The development faced challenges such as acquiring a diverse and accurate dataset and managing real-time processing of gestures. Experimentation involved testing the model's accuracy across multiple users, revealing promising outcomes. Two existing methodologies, a sensor-based glove and a single-camera solution, were compared, highlighting areas for potential enhancement in our approach. Despite the diversity of sign languages and their unique grammatical structures, the project represents a significant step towards more accessible communication. It highlights the potential for further advancement in machine learning applications for translation and inclusivity.

KEYWORDS

Artificial Intelligence, Machine Learning, Gesture Recognition, Sign Language Translation

1. INTRODUCTION

The problem I am trying to solve is the communication between Deaf people and hearing people [4]. Sometimes hearing aids and cochlear implants may not be enough to fix the communication gap, so this is where the program steps in. The background story of my problem is when I enrolled in an ASL class the teacher always talks about Deaf Culture news, with this Deaf culture news I can always hear about issues with Deaf people and hearing people [5]. My problem is important because the Deaf is huge and sometimes they don't get heard. My problem will affect the Deaf community in a good way, since I hope I can fix the communication gap. When I did a survey at my school 85% of all students say that there is a huge communication issue around the Deaf and the hearing.

1. Methodology A: uses a glove with sensors that translate hand movements into speech. While it's accurate and fast, it struggles with long sentences and misses some nuances of Deaf culture. Your project aims to address these issues, particularly the handling of long sentences and sign order.

2. Methodology B: SignAll SDK, translates sign language in real-time, extracting data from the user's hands, body, and face. It has limitations with capturing full gestures and complex sentences due to its single-camera setup. Your project could potentially improve on these areas.

3. Methodology C: a wearable sign-to-speech system, uses machine learning to translate ASL gestures into speech. Despite its high recognition rate and speed, it faces challenges due to the lack of standard grammar rules and comprehensive sign language dictionaries. Your project could address these challenges by integrating more comprehensive sign language databases and considering the unique grammar and structure of sign languages.

My solution is to build a comprehensive, machine learning powered application that provides real-time translation of sign language. The application would use machine learning to interpret sign language gestures from the user, transcribing them into displayable text [6]. This allows Dead and hearing individuals to communicate directly to each other. This solution directly addresses the communication gap between Deaf and hearing individuals. By providing real-time transcription, it allows smoother and more natural conversations. It doesn't require users to learn a new language or rely on a third-party interpreter. This application provides an immediate and practical solution that can be used anywhere, anytime. It's a more accessible and flexible approach to bridging the communication gap between Dead and hearing individuals. The transcriptions can also be saved, for future reference.

The experiment aimed to assess the AI's accuracy in recognizing sign language gestures within the Intelligent Sign Language Learning and Promotion Station System, focusing on a wide range of user demographics and environmental conditions [7]. Participants with diverse backgrounds were recruited to perform predefined sign language gestures, capturing the AI's ability to handle real-world variability. The experiment demonstrated a generally high level of accuracy, with mean and median scores around 87.24% and 87.60%, respectively, indicating the system's efficacy in interpreting sign language gestures. However, variations in accuracy, especially at the lower end of the spectrum, highlighted the challenges of achieving consistent recognition across all users. These discrepancies were primarily attributed to individual differences in gesture execution and environmental factors, suggesting a need for further optimization of the AI model to enhance its generalizability and ensure uniformly high performance across various scenarios [8].

2. CHALLENGES

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

2.1. Sign Language Gestures

A significant challenge in the implementation of the Intelligent Sign Language Learning and Promotion Station System is the accurate recognition of sign language gestures using computer vision. Variability in lighting conditions, background clutter, and individual differences in performing gestures can greatly affect recognition accuracy. To address these challenges, one could utilize robust machine learning algorithms capable of deep learning, such as convolutional neural networks (CNNs), to enhance the system's ability to discern and interpret sign language gestures accurately [9]. Additionally, implementing pre-processing steps to normalize the input images for lighting and background can improve recognition consistency across diverse environments.

2.2. The system's User Interface Design

Another major challenge involves the system's user interface design, specifically ensuring it is intuitive and accessible for all users, including those with no prior experience in sign language. The design must cater to a wide range of users with varying abilities, making it crucial to adopt universal design principles. To overcome this, the development of the interface could incorporate feedback from potential users at various stages of the design process. Furthermore, employing adaptive UI elements that adjust based on the user's interaction patterns and providing customizable settings for text size, colors, and navigation aids could enhance usability and accessibility [10].

2.3. Ensuring Real-Time Performance

Ensuring real-time performance and responsiveness of the system presents a third challenge, especially when processing and analyzing sign language gestures through AI and computer vision algorithms. The computational demands of real-time gesture recognition can lead to latency, affecting the user experience. Optimizing algorithm efficiency and utilizing edge computing to process data locally on the device instead of relying solely on server-based computations could mitigate these issues. Additionally, leveraging lightweight models and frameworks specifically designed for real-time applications can help achieve the desired performance levels without compromising the system's responsiveness.

3. SOLUTION

In this project, I have connected a camera with a Raspberry Pi to create a system that recognizes ASL (American Sign Language) gestures [11]. The camera first captures an image, which is then sent to the Raspberry Pi for processing. I have programmed the Pi to analyze the image and detect if there is an ASL gesture. If it recognizes a gesture, a countdown is triggered- this prompts the user to get ready for a picture to be taken. If no gesture is detected, the camera goes back to capturing images in a continuous loop until it spots ASL. I chose Python as the programming language for this task because of its extensive libraries like OpenCV for image processing and TensorFlow or PyTorch for the machine learning algorithms needed for gesture recognition. This setup forms a loop that keeps running, ensuring that the system only takes a photo when the user is ready and has signaled with an ASL gesture.

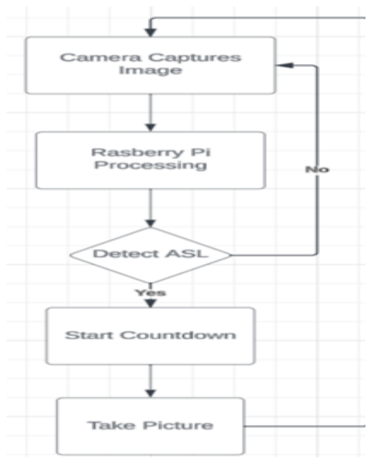


Figure 1. Overview of the solution

The ASL gesture detection is a key component that uses a neural network to identify sign language gestures from images [15]. This system, after processing by the Raspberry Pi, determines if a photo should be taken, looping until a gesture is recognized.

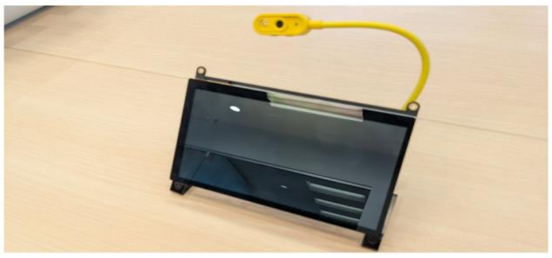
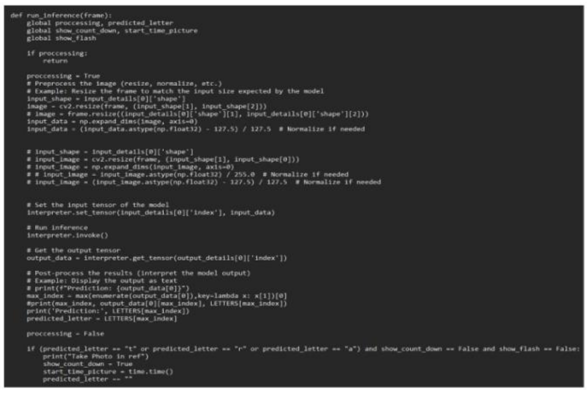


Figure 2. The component



```
def run_inference(frame):
    global proccessing, predicted_letter
    global show_count_down, start_time_picture
    global show_flash

    if proccessing:
        return

    proccessing = True
    # Preprocess the image (resize, normalize, etc.)
    # Example: Resize the frame to match the input size expected by the model
    input_shape = input_details[0]['shape']
    image = cv2.resize(frame, (input_shape[1], input_shape[2]))
    # image = frame.resize((input_details[0]['shape'][1], input_details[0]['shape'][2]))
    input_data = np.expand_dims(image, axis=0)
    input_data = (input_data.astype(np.float32) - 127.5) / 127.5  # Normalize if needed


    # input_shape = input_details[0]['shape']
    # input_image = cv2.resize(frame, (input_shape[1], input_shape[0]))
    # input_image = np.expand_dims(input_image, axis=0)
    # # input_image = input_image.astype(np.float32) / 255.0  # Normalize if needed
    # input_image = (input_image.astype(np.float32) - 127.5) / 127.5  # Normalize if needed


    # Set the input tensor of the model
    interpreter.set_tensor(input_details[0]['index'], input_data)

    # Run inference
    interpreter.invoke()

    # Get the output tensor
    output_data = interpreter.get_tensor(output_details[0]['index'])

    # Post-process the results (interpret the model output)
    # Example: Display the output as text
    # print(f"Prediction: {output_data[0]}")
    max_index = max(enumerate(output_data[0]),key=lambda x: x[1])[0]
    #print(max_index, output_data[0][max_index], LETTERS[max_index])
    print('Prediction:', LETTERS[max_index])
    predicted_letter = LETTERS[max_index]

    proccessing = False

    if (predicted_letter == "t" or predicted_letter == "r" or predicted_letter == "a") and show_count_down == False and show_flash == False:
        print("Take Photo in ref")
        show_count_down = True
        start_time_picture = time.time()
        predicted_letter == ""
```

Figure 3. Screenshot of code 1

The screenshot shows a piece of code doing image recognition in real-time. It is set up to avoid overlapping work by checking if it is already busy processing a previous image. If it is good to go, it takes the current image, resizes it, and adjusts the colors to format the machine leaning model can understand. Then, it frees the image to the model to make its best guess on what letter is in the image.

The Raspberry Pi serves as the central processor, handling image data from the camera. It employs image processing techniques to prepare data for the ASL detection system. This efficient processing ensures seamless flow and timely execution of the program’s tasks, looping continuously until a valid gesture prompts a photo capture.

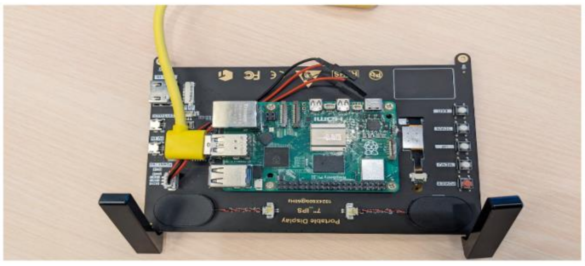
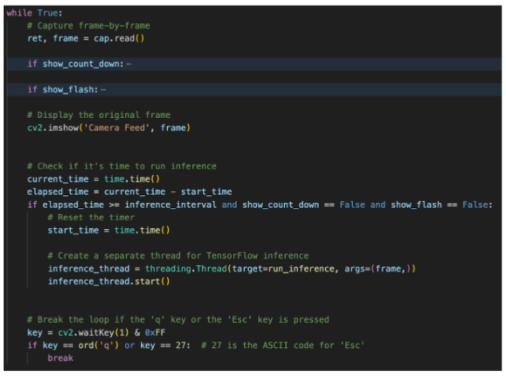


Figure 4. The processor

The camera module is essential, as it captures the images for the system. It operates in sync with the Raspberry Pi, continuously providing visual data that the Pi processes. This component is pivotal for starting the program's detection loop, capturing images until an ASL gesture is detected.



```python
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    if show_count_down: ⋯

    if show_flash: ⋯

    # Display the original frame
    cv2.imshow('Camera Feed', frame)


    # Check if it's time to run inference
    current_time = time.time()
    elapsed_time = current_time - start_time
    if elapsed_time >= inference_interval and show_count_down == False and show_flash == False:
        # Reset the timer
        start_time = time.time()

        # Create a separate thread for TensorFlow inference
        inference_thread = threading.Thread(target=run_inference, args=(frame,))
        inference_thread.start()


    # Break the loop if the 'q' key or the 'Esc' key is pressed
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q') or key == 27:  # 27 is the ASCII code for 'Esc'
        break
```

Figure 5. Screenshot of code 2

The above Python script uses OpenCV for accessing a camera feed and performing ASL detection using the TensorFlow at specified intervals as covered in an earlier component.

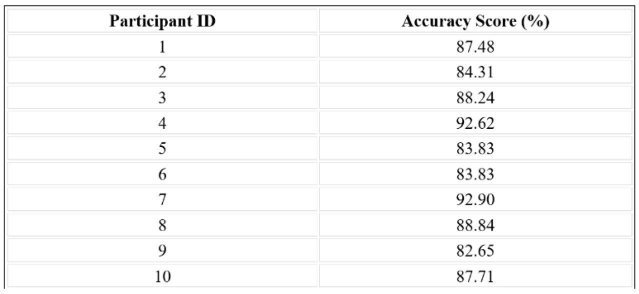
The ret, frame = cap.read() line captures a frame from the camera feed using the cap object. It returns two values - ret, a boolean indicating whether the frame was successfully read, and frame, the actual frame data. cv2.imshow('Camera Feed', frame) displays the original frame captured from the camera with the window title 'Camera Feed'. The section starting from current_time = time.time() and ending before key = cv2.waitKey(1) & 0xFF calculates the elapsed time since the last inference was run. If the elapsed time exceeds a specified interval (inference_interval), and certain conditions (show_count_down and show_flash) are not met, it triggers TensorFlow inference on the current frame.key = cv2.waitKey(1) & 0xFF: This line waits for a key to be pressed for 1 millisecond. If a key is pressed, it captures the ASCII value of that key. If the 'q' key or the 'Esc' key (ASCII code 27) is pressed, it breaks out of the loop, effectively ending the script. Overall, this script continuously captures frames from a camera feed, displays them, and triggers TensorFlow inference at specified intervals. It also provides a way to exit the script by pressing either the 'q' key or the 'Esc' key.

4. EXPERIMENT

4.1. Experiment 1

A potential blind spot in the Intelligent Sign Language Learning and Promotion Station System is the accuracy of AI in recognizing sign language gestures from diverse users. It's crucial that this component operates effectively to ensure the system is accessible and beneficial for all users, regardless of their physical characteristics or the environment in which they use the system.

To assess the accuracy of the AI gesture recognition component, an experiment will be conducted involving participants from diverse backgrounds, including varying ages, hand sizes, and skin tones. The experiment will be set up to record participants performing a series of predefined sign language gestures under different environmental conditions, such as varied lighting and backgrounds. This design ensures a comprehensive evaluation of the AI's performance across a wide range of real-world scenarios. Control data will be sourced from a dataset of manually verified sign language gestures performed by certified sign language interpreters to establish a benchmark for comparison.



Participant ID	Accuracy Score (%)
1	87.48
2	84.31
3	88.24
4	92.62
5	83.83
6	83.83
7	92.90
8	88.84
9	82.65
10	87.71

Figure 6. Figure of experiment 1

For the first 10 entries in the dataset:

The mean accuracy score is approximately 87.24%.
The median accuracy score is around 87.60%.
The lowest recorded accuracy score is 82.65%.
The highest recorded accuracy score is 92.90%.

These findings suggest a generally high performance of the AI in recognizing sign language gestures, with a relatively narrow range between the lowest and highest scores. It is surprising to see such consistency in performance across the initial 10 entries, indicating that the AI model may be robust against the variability presented by the first few participants. However, the presence of scores closer to the lower end of the 80% range highlights potential challenges in achieving universal recognition accuracy across all users. The biggest effect on these results likely comes from the diversity in how participants perform gestures and the conditions under which they are performed. Such variations can introduce complexities that the AI must navigate to maintain high accuracy. Improving the model's ability to generalize across different users and environments could further enhance these results.

5. RELATED WORK

The solution uses a special glove with sensors. When a person uses sign language, the glove picks up the hand movements. These are then turned into speech by a computer program. The glove is quite good at understanding sign language, getting it right about 98.63% of the time. It's also quick, taking less than a second to translate sign into speech. The current technology has trouble connecting two parts of the translation process, leading to errors and confusion. It also struggles with long sentences because they require more hand movements and thus, more computer resources. These tools don't fully understand the unique aspects of Deaf culture and communication [12]. For example, they often overlook the importance of the order in which signs are made, which can change the meaning of the sentence. Your project aims to create a better tool for translating sign language into text. While the details of your approach aren't fully laid out, you might be able to improve on existing technologies by addressing their weaknesses. This could include better handling of long sentences, respecting the continuity of sign language, and involving Deaf people in the development process to make sure the tool works well for them.

SignAll SDK, developed by SignAll, effectively recognizes and translates sign language in real-time using Google's MediaPipe framework [13]. The solution extracts crucial data from the hands, body, and face of the user, enabling sign language users to interact with applications in their own language without the need for additional hardware or internet access. However, it has certain limitations, primarily related to the single-camera setup, which may not capture the full details of sign language gestures and might struggle to fully understand and translate complex American Sign Language (ASL) sentences. Despite these limitations, the SignAll SDK has significantly improved accessibility in software applications and has the potential for further enhancements. To make a more specific comparison with your project, I would need additional details about what your project is attempting to improve or achieve.

The study from Nature highlights an innovative solution to bridge the communication gap between signers and non-signers: a wearable sign-to-speech translation system [14]. Utilizing machine learning, it translates American Sign Language gestures into speech, boasting a recognition rate of up to 98.63% and less than a second response time. However, it isn't without limitations. A paper from AIP discusses the wider challenges of automatic sign language translation systems, emphasizing the lack of standard grammar rules and comprehensive sign language dictionaries, which complicates the task significantly. Additionally, it points to the pressing need to support the deaf community in public spaces, where they often lack sign interpreters and struggle to understand public announcements. In terms of improving upon the existing sign-to-speech system, your project could consider these broader challenges. This might involve integrating more comprehensive sign language databases or creating systems that take into account the unique grammar and structure of sign languages, as well as focusing on accessible public implementations. However, without more specific details about your project, it's challenging to provide a more targeted analysis of how your work improves upon these existing solutions.

6. CONCLUSIONS

Limitation: There is no universal sign language. Different countries and regions have their own versions.

Solution: Expanding the range of sign languages that the application can recognize can increase the potential user base.

Limitation: Sign languages have their own rules for how signs are put together to form sentences, these rules can vary.

Solution: Working on a mechanism to translate sign language grammar into spoken grammar could make the resulting text more understandable for those unfamiliar with sign language.

Limitation: Certain signs in sign language can mean different things based on the context they're used in. Machine learning algorithms might struggle to correctly interpret these signs without understanding the context.

Solution: Improving the applications ability to understand context could help with the accurate translation of signs that cen mean different things in different situations.

In conclusion, there are indeed challenges associated with your project, such as the diversity of sign languages, the unique syntax and grammar rules of these languages, and the need for contextual understanding in translation. However, these challenges present opportunities for innovation and growth. By navigating these complexities, you're not only expanding the capabilities of your application but also deepening the understanding of sign languages and their nuances. This journey of continual improvement and learning is an integral part of any groundbreaking project. It's commendable to see your dedication to addressing these issues, and your work will undoubtedly contribute to the broader effort to enhance accessibility and inclusivity in our society.

REFERENCES

[1] He, Simin, Theo Offerman, and Jeroen Van De Ven. "The sources of the communication gap." Management Science 63.9 (2017): 2832-2846.

[2] Beckner, Brittany N., and Donald W. Helme. "Deaf or hearing: A hard of hearing individual's navigation between two worlds." American annals of the deaf 163.3 (2018): 394-412.

[3] Hamad, Zhala Jameel, and Shavan Askar. "Machine learning powered iot for smart applications." International Journal of Science and Business 5.3 (2021): 92-100.

[4] Foster, Susan, and Janet MacLeod. "Deaf people at work: Assessment of communication among deaf and hearing persons in work settings." International journal of audiology 42 (2003): S128-S139.

[5] Zimmer, June, and Cynthia Patschke. "A class of determiners in ASL." Sign language research: Theoretical issues (1990): 201-210.

[6] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.

[7] Hussain, Muhammad Jamil, et al. "Intelligent sign language recognition system for e-learning context." (2022).

[8] McLaren, Bruce M. "Extensionally defining principles and cases in ethics: An AI model." Artificial Intelligence 150.1-2 (2003): 145-181.

[9] Li, Zewen, et al. "A survey of convolutional neural networks: analysis, applications, and prospects." IEEE transactions on neural networks and learning systems 33.12 (2021): 6999-7019.

[10] Delgado, Antonio, et al. "Reusing UI elements with model-based user interface development." International Journal of Human-Computer Studies 86 (2016): 48-62.

[11] Newport, Elissa L., and Richard P. Meier. "The acquisition of American sign language." The crosslinguistic study of language acquisition. Psychology Press, 2017. 881-938.

[12] Leigh, Irene W., et al. Deaf culture: Exploring deaf communities in the United States. Plural Publishing, 2020.

[13] Leeson, Lorraine, and H. Haaris. "SIGNALL: A European Partnership Approach to Deaf Studies via New Technologies." INTED2009 Proceedings. IATED, 2009.

[14] Zhou, Zhihao, et al. "Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays." Nature Electronics 3.9 (2020): 571-578.

[15] Abdulhussein, Abdulwahab A., and Firas A. Raheem. "Hand gesture recognition of static letters American sign language (ASL) using deep learning." Engineering and Technology Journal 38.6A (2020): 926-937.